ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση ενός Μαζικού Ανοικτού Διαδικτυακού Μαθήματος (MOOC) για τη "Γλώσσα Προγραμματισμού C"

Δημοπούλου Γερασιμούλα

Επιβλέπων: Κουτσονίκος Ιωάννης

ΠΑΤΡΑ, 2015

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία χρησιμοποιεί την προτεινόμενη μεθοδολογία ADDIE που είναι ένα διαδεδομένο πρότυπο για το σχεδιασμό εκπαιδευτικού υλικού και με βάση την μεθοδολογία αυτή σχεδιάζεται ένα μαζικό ανοιχτό διαδικτυακό μάθημα (Massive Open Online Course, MOOC) με αντικείμενο τον προγραμματισμό ηλεκτρονικών υπολογιστών με τη γλώσσα προγραμματισμού C. Για το σκοπό αυτό δημιουργήθηκαν ομάδες όπως η ομάδα Διαχείρισης του Μαθήματος, η ομάδα Υποστήριξης Εκπαιδευτικών, η ομάδα Τεχνικής υποστήριξης και άλλοι συνεργάτες. Έτσι υλοποιήθηκαν σταδιακά οι πέντε φάσεις που υποστηρίζει το μοντέλο μεθοδολογίας ADDIE, ενώ μέχρι και την ολοκλήρωση της συγγραφής της παρούσας εργασίας, τα τελευταία στάδια των φάσεων του μοντέλου βρίσκονταν σε εξέλιξη. Για το λόγο αυτό αποφασίστηκε να γίνει εκτενής παρουσίαση των αποτελεσμάτων της φάσης σχεδιασμού με βάση τα οποία υλοποιήθηκαν και οι επόμενες φάσεις ως το κύριο μέρος της πτυχιακής εργασίας. Άρα στο τρίτο και κύριο μέρος της εργασίας, μετά την εισαγωγική παρουσίαση των MOOC και την περιγραφή της μεθοδολογίας, παρουσιάζονται οι έξι ενότητες του σχεδιαζόμενου μαθήματος, οι οποίες αντιστοιχούν στο χρονικό διάστημα των έξι εβδομάδων διάρκειας του MOOC με τίτλο «Βασικές αρχές Προγραμματισμού χρησιμοποιώντας την ANSI - C». Κάθε ενότητα αποτελείται από τρεις το πολύ δραστηριότητες και κάθε δραστηριότητα από ένα έως τρία μαθησιακά αντικείμενα υποστήριξης της αντίστοιχης θεωρίας και ένα έως πέντε μαθησιακά αντικείμενα αντίστοιχων ασκήσεων αξιολόγησης για κάθε θεωρία. Έτσι περιγράφονται συνολικά 18 περίπου δραστηριότητες και 170 περίπου μαθησιακά αντικείμενα τα οποία όλα υλοποιούνται ξεχωριστά με συγκεκριμένο μαθησιακό τύπο (βιντεοδιάλεξη, υπερκείμενο, παρουσιάσεις, ερωτήσεις αντικειμενικού τύπου, εργασίες κλπ.) και ανεβαίνουν σταδιακά στην πλατφόρμα EAP-MOOC που μας διέθεσε το Ελληνικό Ανοιχτό Πανεπιστήμιο.

1. ΕΙΣΑΓΩΓΗ

1.1 Τι είναι ένα MOOC

Ο όρος MOOC (Massive Open Online Courses) ή «Μαζικά Ελεύθερα Διαδικτυακά Μαθήματα», όπως αποδίδεται στα ελληνικά, εμφανίστηκε για πρώτη φορά στο χώρο της εκπαίδευσης το 2008 στο Πανεπιστήμιο της Manitoba του Καναδά και από τότε έχει προσελκύσει την προσοχή των ΜΜΕ, διεθνώς αναγνωρισμένων πανεπιστημιακών ιδρυμάτων, ερευνητικών οργανισμών αλλά και σημαντικών επενδυτών. Το ακρωνύμιο MOOC δείχνει και την ουσιαστική σημασία των μαθημάτων αυτών, αφού «massive» σημαίνει «μαζικά» διότι, έχουν σχεδιαστεί ώστε να επιτρέπουν την εγγραφή δεκάδων χιλιάδων εκπαιδευομένων, «open» σημαίνει «ελεύθερα» διότι, ο καθένας με μια σύνδεση στο διαδίκτυο μπορεί να εγγραφεί στην ελεύθερη σειρά μαθημάτων, «online» σημαίνει «διαδικτυακά» διότι, η αλληλεπίδραση λαμβάνει χώρα μέσω διαδικτύου, ομάδων συζητήσεων ή/και παρακολούθησης βιντεοδιαλέξεων και τέλος «courses» σημαίνει «μαθήματα» διότι, έχουν συγκεκριμένες ημερομηνίες έναρξης και λήξης και αξιολόγηση των εκπαιδευομένων.

Η διδασκαλία μέσω ενός MOOC γίνεται με διαφόρους τύπους εκπαιδευτικού υλικού, με τα εκπαιδευτικά βίντεο να αξιοποιούνται σε πολύ υψηλότερο ποσοστό. Παράλληλα, οι εκπαιδευόμενοι έχουν τη δυνατότητα αλληλεπίδρασης με τους υπόλοιπους μέσα από διαδικτυακές ομάδες συζητήσεων (forum, social media, webcasts, συνεδρίες ή ακόμα και δια ζώσης συναντήσεις). Τα περισσότερα MOOC δεν υποστηρίζουν την επικοινωνία μεταξύ των διδασκόντων και των εκπαιδευομένων κυρίως αυτά που προσελκύουν πολύ μεγάλο αριθμό συμμετεχόντων.

Παρακάτω παρουσιάζονται οι περισσότερο κοινοί τύποι εκπαιδευτικού υλικού που μπορούν να χρησιμοποιηθούν από την πλατφόρμα ενός MOOC:

1. Εκπαιδευτικά Βίντεο (Educational Video): Τα εκπαιδευτικά βίντεο παρουσιάζουν συγκεκριμένες ενότητες εκπαιδευτικού περιεχομένου με οπτικοακουστικό υλικό από τον εκπαιδευτή. Διατίθενται σε μορφή η οποία μπορεί να αναπαραχθεί είτε σε CD είτε μέσω φυλλομετρητών στο διαδίκτυο.

2. Παρουσίαση (Presentation): Μια παρουσίαση μπορεί να είναι οποιοσδήποτε τύπος μορφής διαφανειών για τους εκπαιδευομένους.

3. Υπερκείμενο (Hypertext): Αυτός ο τύπος εκπαιδευτικού υλικού αποτελεί ηλεκτρονικό κείμενο που έχει διασυνδεθεί με άλλα κείμενα.

4. Δοκίμιο (Document): Ένα δοκίμιο μπορεί να είναι οποιοσδήποτε τύπος εγγράφου PDF ή φυλλάδιο για εκπαιδευομένους.

5. Ήχος (Audio): Ένας ήχος μπορεί να είναι οποιοσδήποτε τύπος ήχου για τους εκπαιδευομένους.

6. Κουίζ: Αποτελεί οποιονδήποτε τύπο κουίζ που υλοποιεί ο εκπαιδευόμενος.

7. Wiki: Ένα Wiki είναι συνήθως μία ιστοσελίδα που επιτρέπει στους χρήστες της να προσθέσουν, να αφαιρέσουν, ή να επεξεργαστούν το περιεχόμενό της, γρήγορα και εύκολα.

8. Εργασίες: Αποτελεί οποιαδήποτε εργασία (Project) που δίνεται στον εκπαιδευόμενο.

Τα MOOCs γίνονται όλο και πιο δημοφιλή σήμερα εντός των κύκλων της τριτοβάθμιας εκπαίδευσης και σε μεγάλα αμερικανικά πανεπιστήμια, όπως το Στάνφορντ, το Πρίνστον, το

πανεπιστήμιο του Μίσιγκαν και το Μπέρκλεϊ στην Καλιφόρνια, παρέχουν δωρεάν κύκλους σπουδών σε διάφορα γνωστικά αντικείμενα, αν και παρέχουν ένα απλό πιστοποιητικό παρακολούθησης και όχι κανονικό πτυχίο. Βέβαια, μπορεί να χρεωθεί ένα ποσό για πιστοποίηση της παρακολούθησης αλλά όχι για το εκπαιδευτικό υλικό ή για τη χρήση της εκπαιδευτικής πλατφόρμας. Παρότι είναι μια σύγχρονη υπηρεσία, έχουν επενδύσει σε αυτή αρκετές εταιρείες και οργανισμοί. Η ευρεία διάδοση και αποδοχή των MOOCs έχει ενθαρρύνει κορυφαία διεθνή πανεπιστήμια να προσφέρουν τα μαθήματά τους στο διαδίκτυο μέσω της δημιουργίας ανοικτών πλατφορμών μάθησης, όπως η edX (Yuan & Powell, 2013). Η πλατφόρμα edX δημιουργήθηκε και στο Ελληνικό Ανοιχτό Πανεπιστήμιο ΕΑΠ και χρησιμοποιήθηκε για την υλοποίηση του μαθήματος της παρούσας πτυχιακής εργασίας. Η πλατφόρμα αυτή θα αναφέρεται παρακάτω σαν πλατφόρμα EAP-MOOC.

Πρωτοπόρος και ηγέτης στην ευρεία διάδοση και αποδοχή των MOOC έχει αναδειχθεί επίσης η Αμερικάνικη εταιρία Coursera με σχεδόν 5,5 εκατομμύρια εγγεγραμμένους χρήστες/μαθητές το έτος 2013 δηλαδή 540 μαθήματα σε 18 διαφορετικά αντικείμενα από 107 συνεργάτες. Επίσης, η Udacity έχει ξεκινήσει σε συνεργασία με έγκριτα πανεπιστήμια να προσφέρει δωρεάν διαδικτυακά μαθήματα ή χρεώνοντας ένα μικρό ποσό στην περίπτωση που κάποιος εκπαιδευόμενος επιθυμεί να λάβει μέρος σε μια διαδικασία πιστοποίησης των γνώσεών του. Οι Breslow et al., (2013) επισημαίνουν ότι εταιρείες κολοσσοί, όπως η Google σχεδιάζουν να διεισδύσουν στο χώρο της τριτοβάθμιας εκπαίδευσης ως «παγκόσμιοι παίκτες» παροχής εκπαιδευτικών υπηρεσιών και είναι πιθανό να υιοθετήσουν μια MOOC προσέγγιση ως μέρος των σχεδίων τους. Στην Ευρώπη, μια νέα εταιρεία, η Futurelearn, δημιουργήθηκε στο Ηνωμένο Βασίλειο από το Ανοικτό Βρετανικό Πανεπιστήμιο, παρέχοντας MOOC σε συνεργασία με κορυφαία πανεπιστήμια του Ηνωμένου Βασιλείου. Από Ευρωπαϊκής πλευράς, η Ευρωπαϊκή Ένωση έχει επενδύσει σε αυτό τον τομέα δημιουργώντας το OpenupEd.

1.2 Πλεονεκτήματα - Οφέλη των MOOCs

Ο κύριος στόχος των MOOCs ήταν να παρέχουν ανοιχτή μάθηση και ελεύθερη πρόσβαση πανεπιστημιακού επιπέδου σε όσο περισσότερους φοιτητές (Yuan και Powell, 2013). Οι Chen, Barnett και Stephens στο άρθρο τους Fad of Future: The Advantages and Challenges of Massive Open Online Courses (2013) διακρίνουν τρία βασικά πλεονεκτήματα των MOOCs:

Μαζικότητα: Μπορούν να υποστηρίξουν μεγάλο πλήθος συμμετεχόντων καθώς παρέχουν πρόσβαση σε ανθρώπους οι οποίοι διαφορετικά θα είχαν αποκλειστεί για ποικίλους λόγους όπως: ο χρόνος και ο τόπος διεξαγωγής των μαθημάτων, οι τυπικές προϋποθέσεις και οι οικονομικές δυσκολίες. Τα MOOCs αναδύουν μια αίσθηση δημοκρατικής και ελεύθερης πρόσβασης στην εκπαίδευση, μια αναπτυξιακή ατζέντα, καθώς και τη φιλοσοφία ότι οι δυνατότητες εκπαίδευσης δεν θα πρέπει να περιορίζονται πλέον σε λίγους προνομιούχους. Οι συμμετέχοντες και οι εκπαιδευτές διαπιστώνουν σημαντικά οφέλη από τη βελτιωμένη προσβασιμότητα που προσφέρουν τα MOOCs. Άλλοι ερευνητές επισημαίνουν τα πλεονεκτήματα που απορρέουν από τη δημιουργία κοινοτήτων μάθησης και στην ανάπτυξη δεξιοτήτων μάθησης του 21ου αιώνα. Οι υποστηρικτές των MOOCs υποστηρίζουν ότι ηλεκτρονικές πλατφόρμες μάθησης όπως η Coursera, edX και Udacity συμβάλλουν στον εκδημοκρατισμό της τριτοβάθμιας εκπαίδευσης σε χώρες που πριν είχαν μικρή πρόσβαση σε αυτήν. Η UNESCO (2012) αναγνωρίζει τη συμβολή τους για την καθολική πρόσβαση του πληθυσμού στη γνώση, καθώς και για την προαγωγή του εκδημοκρατισμού της εκπαίδευσης.

Ανοικτότητα: Στα MOOCs όλα είναι ανοικτά, από το λογισμικό που χρησιμοποιείται μέχρι την εγγραφή στα μαθήματα και τη διδακτέα ύλη, η οποία δεν είναι αυστηρά καθορισμένη

αλλά μπορεί να μεταβάλλεται καθώς το μάθημα εξελίσσεται στην πορεία. Οι πηγές πληροφόρησης είναι ανοικτές και τέλος τα μαθησιακά περιβάλλοντα που θα επιλέξει ο χρήστης είναι και αυτά ανοικτά. Έτσι τα MOOCs μέχρι σήμερα έχουν εγγράψει εκατομμύρια εκπαιδευόμενους από όλο τον κόσμο, με 5.5 εκατομμύρια χρήστες να εγγράφονται μόνο στα μαθήματα που παρέχει η Coursera. Άλλωστε, για τα ίδια τα πανεπιστημιακά ιδρύματα η παροχή MOOC μπορεί να επεκτείνει διεθνώς την πρόσβαση και τη φήμη τους. Επιπρόσθετα, τα πανεπιστήμια MIT και Harvard χρησιμοποιούν την πλατφόρμα edX προκειμένου να κατανοήσουν αφενός πώς μαθαίνουν οι φοιτητές τους και αφετέρου να εισάγουν τις καινοτομίες των MOOC όσον αφορά τη διδασκαλία και τη μάθηση εντός των πανεπιστημιακών αιθουσών.

Φιλοσοφία του Κονεκτιβισμού: Τα MOOCs βασίζονται στις αρχές του Κονεκτιβισμού όπως: η αυτονομία, η ποικιλομορφία, η ανοικτότητα και η αλληλεπίδραση. Δηλαδή ο σύγχρονος εκπαιδευόμενος μαθαίνει πλοηγούμενος σε μια πληθώρα διαδικτυακών πηγών γνώσης, επιλέγοντας μόνος του το περιεχόμενο, τον τρόπο, το χρόνο και τα εργαλεία που θα χρησιμοποιήσει για να επιτύχει τους προσωπικούς του εκπαιδευτικούς στόχους. Οι στρατηγικές διδασκαλίας που βασίζονται στον Κονεκτιβισμό επιτρέπουν στο διδάσκοντα να αναλάβει ρόλο διευκολυντή προκειμένου οι εκπαιδευόμενοι να συμμετέχουν και να αλληλεπιδρούν ενεργητικά με βάση τους μαθησιακούς στόχους, την προϋπάρχουσα γνώση, τις δεξιότητες και τα κοινά τους ενδιαφέροντα. Τα MOOCs χρησιμοποιούν στρατηγικές παρόμοιες με εκείνες των κοινωνικών δικτύων προκειμένου να συνδέσουν τις μάζες με το επιπρόσθετο όφελος ότι υπάρχουν ειδικοί οι οποίοι οργανώνουν το περιεχόμενο (McAuley, Stewart, Siemens & Cormier, 2010). Το San Jose State University στις ΗΠΑ προσπαθεί να εισάγει τα χαρακτηριστικά των MOOCs σε παραδοσιακές τάξεις αξιοποιώντας την εμπειρία του στην παροχή των MOOCs, προκειμένου οι παραδοσιακοί φοιτητές του να συμμετέχουν ενεργητικότερα στη μαθησιακή διαδικασία.

1.3 Μειονεκτήματα - Προκλήσεις των MOOCs

Η αλματώδης ανάπτυξη των MOOCs, παρ' όλα αυτά, προκάλεσε και αρνητικές κριτικές με αποτέλεσμα να έχουν αρχίσει ήδη να αμφισβητούνται για το κατά πόσο προσφέρουν ουσιαστική εκπαίδευση.

Πνευματικά Δικαιώματα και πιστοποίηση: Πως διασφαλίζονται τα δικαιώματα πνευματικής ιδιοκτησίας; Είναι γεγονός ότι η πρόσβαση σε τεράστιες πηγές πληροφορίας είναι ένα από τα πλεονεκτήματα των MOOCs. Ως εκ τούτου πρέπει να βρεθούν μηχανισμοί ώστε το υλικό που διατίθεται να είναι νόμιμο και να εξασφαλίζει τα πνευματικά δικαιώματα των συγγραφέων.

Επίσης το απλό πιστοποιητικό που παρέχεται είναι αρκετό για τους συμμετέχοντες; Αναγνωρίζεται; Είναι επίσης πολύ σημαντική η σωστή επιλογή της ενότητας, μια που ολοένα και περισσότεροι οργανισμοί (αλλά και επιχειρήσεις) συμμετέχουν στη διαδικασία, η εγκυρότητα των οποίων δεν είναι πάντα διασφαλισμένη.

Σε όσα MOOCs παρέχουν σήμερα κάποια μορφή πιστοποίησης, δεν λείπουν οι ανησυχίες που σχετίζονται με προβλήματα λογοκλοπής και δικαιώματα πνευματικής ιδιοκτησίας. Προκειμένου να αντιμετωπίσει το πρόβλημα αυτό η Coursera και η Udacity αναπτύσσουν πολιτικές χρηματοδότησης των δράσεών τους οι οποίες περιλαμβάνουν την πώληση στοιχείων των εκπαιδευομένων σε πιθανούς εργοδότες ή διαφημιστές, τακτική η οποία όπως επισημαίνουν οι Liyanagunawardena, Adams & Williams (2013) εγείρει σημαντικά ηθικά ζητήματα που σχετίζονται με την αυθαίρετη συχνά χρήση δεδομένων των χρηστών. Άλλες

πρακτικές αφορούν τη συνεργασία με εκδοτικούς οίκους για τη διάθεση ολόκληρου ή μέρους του ψηφιακού υλικού, την αναζήτηση χορηγών καθώς και την παροχή τελών για την απόκτηση πιστοποιητικών (Marshall, 2013). Η Coursera προσφέρει σήμερα στο 14% των μαθημάτων της τη δυνατότητα απόκτησης πιστοποιητικού επιτυχούς παρακολούθησης με αυθεντικοποίηση της ταυτότητας του χρήστη (Signature Track) με κόστος εγγραφής μεταξύ 30 και 100 δολαρίων.

Καθοδήγηση: Τα περισσότερα MOOCs βασίζονται στη θεωρία του κονεκτιβισμού, δημιουργώντας πολλούς προβληματισμούς ως προς τη διδακτική τους εφαρμογή: «Μήπως ο κονεκτιβισμός στην πράξη δημιουργεί ένα χάος για τον μαθητή, χωρίς ιδιαίτερη καθοδήγηση; Απαιτούνται έμπειροι χρήστες; Τι συμβαίνει με τον ιδιαίτερα μεγάλο αριθμό των μαθητών που εγκαταλείπουν λίγο μετά την εγγραφή τους;». Υπάρχουν βεβαίως οι βοηθοί στα περισσότερα MOOCs οι οποίοι πολλές φορές επιδρούν καταλυτικά, μια που η προσέγγιση του διδάσκοντα είναι δύσκολη. Αυτό συμβαίνει γιατί ο διδάσκων δεν μπορεί να είναι διαθέσιμος για τους δεκάδες χιλιάδες μαθητές του. Η κοινωνικότητα των συμμετεχόντων σε ένα ανοιχτό μάθημα πρέπει να υποστηρίζεται έντονα και αυτό είναι σπουδαία πρόκληση για τους υπευθύνους του μαθήματος, ειδικά αφού απευθύνονται σε ενήλικες που συχνά είναι πιο επιφυλακτικοί ως προς τις συναναστροφές τους σε σχέση με εκπαιδευόμενους νεαρής ηλικίας (MacKay, 2013). Η έλλειψη της αλληλεπίδρασης με ομότιμους είναι σημείο προβληματισμού.

Έτσι μια μεγάλη πρόκληση για τα MOOCs που σχετίζεται με τα χαμηλά ποσοστά επιτυχίας είναι τα υψηλά ποσοστά εγκατάλειψης των εκπαιδευομένων. Τα ποσοστά εγκατάλειψης στα MOOC τα οποία προσφέρονται από το Stanford, MIT και UC Berkley κυμαίνονται μεταξύ 80% και 95% (Yuan & Powell, 2013). Δίχως τα εμπόδια της καταβολής διδάκτρων ή κάποιας διαδικασίας προεπιλογής κάποιοι εκπαιδευόμενοι ενδεχόμενα να εγγραφούν σ' ένα MOOC απλώς από περιέργεια και όχι διότι στην πραγματικότητα θέλουν να εμπλακούν στη μαθησιακή διαδικασία. Άλλοι εκπαιδευόμενοι ενδεχόμενα έχουν στόχους διαφορετικούς από τους προβλεπόμενους από τους διοργανωτές των εκπαιδευτικών προγραμμάτων. Παρ' όλα αυτά, έστω και αν αφαιρεθούν τα υψηλά ποσοστά ατόμων που δεν έχουν ολοκληρώσει τα μαθήματα, εκατοντάδες χιλιάδες άνθρωποι αρχίζουν να συμμετέχουν σε MOOCs. Για την αντιμετώπιση της διαρροής, η Coursera έχει αθόρυβα μετακινηθεί σε βραχύτερης διάρκειας μαθήματα, μια αλλαγή η οποία δεν αναμένεται να έχει σημαντική επίπτωση στην ποιότητα των παρεχόμενων υπηρεσιών της. Η edX διερευνά πιο ριζικές αλλαγές στο υπάρχον MOOC μοντέλο. Συγκεκριμένα, έχει αρχίσει να προσφέρει SPOC (Small, Private Online Courses), όπως θα μεταφράζαμε στα Ελληνικά, σύντομα ιδιωτικά διαδικτυακά μαθήματα. Μια άλλη πειραματική ακόμα πρακτική περιλαμβάνει τον προγραμματισμό κάποιων ζωντανών τηλεδιασκέψεων, στις οποίες μπορούν να συμμετάσχουν όσοι εκπαιδευόμενοι ενδιαφέρονται.

Εκπαιδευτική διαδικασία: Η εκπαιδευτική διαδικασία στα MOOCs συντελείται αυτόνομα, πολλές φορές δε, μοιάζει να εξελίσσεται ανεξέλεγκτα από τον διδάσκοντα, γεγονός που οδηγεί σε υποβάθμιση του ρόλου του εκπαιδευτικού. Ο εκπαιδευτικός έρχεται αντιμέτωπος με προβλήματα που συνδέονται είτε άμεσα, είτε έμμεσα με το διδακτικό του έργο (Παγγέ, 2009) μια που οφείλει να διαχειρίζεται αποτελεσματικά τόσο τη διαδικασία μετάδοσης της γνώσης όσο και τη διαχείριση της τάξης, την κινητοποίηση των μαθητών και τη διαμόρφωση της προσωπικότητάς τους (Παγγέ 2009, Villegas et al., 2002). Συχνά, τα κριτήρια επιλογής των καθηγητών είναι πολύ διαφορετικά από αυτά στη συμβατική εκπαίδευση. Το ερευνητικό έργο λαμβάνεται ελάχιστα υπ' όψιν και εκτιμώνται περισσότερο οι διδακτικές ικανότητες του καθηγητή. Αρκετοί ακαδημαϊκοί ανησυχούν ότι τα MOOC και τα συνδεδεμένα μ' αυτά διαδικτυακά ιδρύματα εν τέλει θα αντικαταστήσουν την παραδοσιακή πρόσωπο με πρόσωπο επικοινωνία μεταξύ εκπαιδευτικού και εκπαιδευόμενου. Ανησυχούν, επίσης, ότι η ακαδημαϊκή κοινότητα μελλοντικά θα έχει λιγότερα μέλη, καθώς οι 'παραδοσιακοί'

καθηγητές σταδιακά θα αντικατασταθούν από χαμηλόμισθους βοηθούς και τεχνικό προσωπικό. Ορισμένοι προβληματίζονται ότι, καθώς τα MOOC θα παρέχουν τη δυνατότητα και στους νεαρούς εκπαιδευόμενους να παρακολουθούν μαθήματα σε τομείς όπως τη λογοτεχνία, φιλοσοφία κ.ά., τα πανεπιστήμια αναπόφευκτα θα πιεστούν προς την παροχή ενός αναλυτικού προγράμματος σπουδών βασισμένο κυρίως στην ανάπτυξη δεξιοτήτων (Littlefield, 2013).

2. ΜΕΘΟΔΟΛΟΓΙΑ

Η προτεινόμενη μεθοδολογία για την παρούσα πτυχιακή εργασία αφορά σε μια συνεργατική ανάπτυξη ενός μαθήματος MOOC για τον προγραμματισμό με τη γλώσσα C. Η μεθοδολογία αυτή υιοθετεί τα βασικά στοιχεία του μοντέλου ADDIE (Analysis Design Development Implementation Evaluation). Ο όρος ADDIE αποτελεί ακρωνύμιο των διαδοχικών διαδικασιών που απαιτεί κάθε εκπαιδευτικός σχεδιασμός, τα οποία είναι: η Ανάλυση, ο Σχεδιασμός, η Ανάπτυξη, η Εφαρμογή και η Αξιολόγηση. Το ADDIE χαρακτηρίζεται ως ένα σχεδιαστικό πρότυπο και πολλά από τα σύγχρονα μοντέλα διδασκαλίας αποτελούν παραλλαγές αυτού του μοντέλου. Για παράδειγμα, το ερευνητικό εργαστήριο CELSTEC του Ανοιχτού Πανεπιστημίου της Ολλανδίας χρησιμοποίησε το μοντέλο ADDIE για το σχεδιασμό ενός συστήματος ηλεκτρονικής μάθησης, το σύστημα OpenU για τη δια βίου μάθηση ενηλίκων εξ αποστάσεως, το οποίο παρέχει ανοιχτά μαθήματα και βασίζεται στην εξατομικευμένη εκπαίδευση. Επιπροσθέτως, ενώ τα βασικά στοιχεία του μοντέλου ADDIE παραμένουν σταθερά, οι δραστηριότητες συνήθως δεν είναι οργανωμένες με ένα γραμμικό τρόπο. Αντιθέτως, η εκπαιδευτική διαδικασία σχεδιασμού είναι επαναληπτική και αυτοδιορθωτική. Επομένως, μια τέτοια διαδικασία σχεδιασμού, η οποία παρέχει συνεχή αξιολόγηση σε όλες τις φάσεις της, δίνει τη δυνατότητα συνεχών βελτιώσεων σε καινοτόμες και νέες εφαρμογές όπως είναι τα μαθήματα υπό μορφή ανοιχτών μαζικών διαδικτυακών μαθημάτων (MOOC). Οι πλατφόρμες που χρησιμοποιούν τα πιο δημοφιλή MOOCs όπως Coursera, FutureLearn, Udemy, Udacity, edX και Iversity διερευνήθηκαν ως παράδειγμα προς μίμηση.

2.1 Μεθοδολογία - Βήματα

Όσον αφορά το περιεχόμενο της προαναφερθείσας μεθοδολογίας, αυτή ακολουθεί πέντε φάσεις:

Φάση Ανάλυσης:

Στη φάση αυτή διευκρινίζεται το διδακτικό πρόβλημα, οι διδακτικοί στόχοι και σκοποί και περιγράφεται το μαθησιακό περιβάλλον. Επίσης, αναγνωρίζονται τα χαρακτηριστικά των εκπαιδευομένων, η προϋπάρχουσα γνώση τους και οι δεξιότητες που έχουν αποκτήσει. Κατά τη διάρκεια της φάσης ανάλυσης επομένως, το εκπαιδευτικό θέμα του μαθήματος αναλύεται, προκειμένου να προσδιοριστεί ο σκοπός της μάθησης, το γνωστικό πεδίο και οι κύριοι στόχοι μάθησης.

Οι ομάδες που συμμετέχουν στη φάση της ανάλυσης είναι η ομάδα διαχείρισης του μαθήματος, η οποία είναι υπεύθυνη για την παραγωγή των αποτελεσμάτων της φάσης και η Ομάδα Υποστήριξης Εκπαιδευτικών η οποία θα βοηθήσει και θα στηρίξει την πρώτη ομάδα.

Χρήσιμες ερωτήσεις είναι: ποιο το ακροατήριο και τα χαρακτηριστικά του, ποιο το επιθυμητό αποτέλεσμα της γνώσης, ποιοι οι περιορισμοί της μάθησης και το είδος τους, με ποιους τρόπους και μορφές θα παραδοθεί η γνώση, ποιο το παιδαγωγικό πλαίσιο της παρέμβασης και ποια τα χρονικά πλαίσια της παρέμβασης.

Για την καλύτερη οργάνωση των αποτελεσμάτων της φάσης Ανάλυσης προτείνεται η συμπλήρωση του πρότυπου πίνακα «Πίνακας 1: Πίνακας Περιγραφής Μαθήματος (Φάση Ανάλυσης)», ο οποίος δίνεται στο παράρτημα αυτής της εργασίας.

Φάση σχεδιασμού:

Στη φάση αυτή σχεδιάζονται οι ρόλοι, τα εργαλεία αξιολόγησης, οι ασκήσεις, το περιεχόμενο που θα χρησιμοποιηθεί για να πετύχει τους μαθησιακούς στόχους, η ροή των δραστηριοτήτων και η επιλογή των μέσων με τρόπο λεπτομερή και συστηματικό (οργανωμένο με βάση συγκεκριμένες στρατηγικές). Συγκεκριμένα:

- Καταγράφεται η στρατηγική που θα εφαρμοστεί στη διδασκαλία
- Εφαρμόζονται στρατηγικές διδασκαλίας ανάλογα με τα επιθυμητά αποτελέσματα στο γνωστικό, συναισθηματικό και ψυχοκινητικό τομέα
- Σχεδιάζεται η μαθησιακή εμπειρία

Κατά τη διάρκεια της φάσης σχεδιασμού, εκπαίδευσης, οι μαθησιακές δραστηριότητες και τα μαθησιακά αντικείμενα (NS) σχεδιάζονται για να δείξουν τον τρόπο με τον οποίο η γνώση θα πρέπει να προσφέρεται στους εκπαιδευόμενους. Ο εκπαιδευτικός σχεδιασμός δεν έχει σκοπό να δημιουργήσει μια τυποποιημένη μορφή, αλλά κάθε μάθημα θα πρέπει να σχεδιαστεί με βάση τις ανάγκες που αποκαλύφτηκαν στη φάση της ανάλυσης. Σε αυτή τη φάση, ο Διευθυντής της ομάδας του μαθήματος ασχολείται με τη συνεργασία της Ομάδας Στήριξης.

Για την καλύτερη οργάνωση των αποτελεσμάτων της φάσης Σχεδιασμού προτείνεται η συμπλήρωση των Πρότυπων Πινάκων (Πίνακας 2α: Πίνακας Περιγραφής Ενοτήτων, Πίνακας 2β: Πίνακας Περιγραφής Εκπαιδευτικών Δραστηριοτήτων και Πίνακας 2γ: Πίνακας Περιγραφής Μαθησιακών Αντικειμένων), οι οποίοι δίνονται στο παράρτημα αυτής της εργασίας.

Φάση Ανάπτυξης:

Στη φάση αυτή οι σχεδιαστές δημιουργούν και συγκεντρώνουν το διδακτικό περιεχόμενο που έχουν αποφασίσει στην προηγούμενη φάση. Εφαρμόζονται τεχνολογικά μέσα υποστήριξης της διδασκαλίας και γίνεται δοκιμή της λειτουργίας τους ώστε να υπάρξει ανατροφοδότηση και τροποποιήσεις πριν να χρησιμοποιηθούν στη διδασκαλία.

Στη φάση ανάπτυξης, τα αντικείμενα μάθησης έχουν δημιουργηθεί. Ο Υπεύθυνος Ομάδας Μαθημάτων συνεργάζεται με τους ειδικούς προγραμματιστές πολυμέσων (επιπλέον προσωπικό). Η Ομάδα Υποστήριξης Εκπαιδευτικών βοηθά τη διαδικασία, ενώ η ομάδα τεχνικής υποστήριξης διαμορφώνει την πλατφόρμα MOOC και ανεβάζει το εκπαιδευτικό υλικό στην πλατφόρμα.

Φάση υλοποίησης:

Στη φάση αυτή υλοποιείται η διδακτική παρέμβαση όπως έχει σχεδιαστεί. Εφόσον χρειάζεται γίνεται εκπαίδευση του μαθητευόμενου και του διδάσκοντα και παραδίδονται τα μαθησιακά υλικά και το περιεχόμενο στους μαθητευόμενους. Η εκπαιδευτική διαδικασία υλοποιείται με τη χρήση του καθορισμένου χρονικού πλαισίου.

Πριν τη διεξαγωγή των ανεπτυγμένων μαθημάτων θα πραγματοποιηθεί πιλοτική εφαρμογή προκειμένου να δοκιμαστεί και να αξιολογηθεί το μάθημα και η πλατφόρμα για τυχόν βελτιώσεις και αντιμετώπιση προβλημάτων. Η πιλοτική εφαρμογή του μαθήματος θα διεξαχθεί σε ένα μικρό αριθμό εκπαιδευομένων και σε έμπειρο επιστημονικό προσωπικό. Κατά την αξιολόγηση και τη δοκιμή του μαθήματος και της πλατφόρμας θα χρειαστεί η βοήθεια και άλλων συνεργατών προκειμένου να αξιολογηθούν και βελτιωθούν όσο το δυνατόν περισσότερο.

Στη συνέχεια, τα μαθήματα MOOC θα διεξαχθούν και με την ολοκλήρωσή τους θα πραγματοποιηθεί διάχυση των αποτελεσμάτων. Κατά τη διάρκεια αυτής της φάσης, ο διαχειριστής του μαθήματος συνεργάζεται με την εκπαιδευτική ομάδα υποστήριξης κατά τη

διάρκεια της εκπαιδευτικής διαδικασίας. Η Ομάδα Τεχνικής Υποστήριξης παρέχει τεχνική υποστήριξη σε όλη την εκπαιδευτική διαδικασία.

Συνοψίζοντας, στη φάση της αξιολόγησης αποτιμάται **διαμορφωτικά** (κατά την διάρκεια της διαδικασίας) και **αθροιστικά** (στο τέλος της διαδικασίας) η επίτευξη των αρχικών διδακτικών στόχων και η επιτυχία της όλης διαδικασίας και γίνονται προτάσεις για βελτίωση και αναθεώρηση ορισμένων στοιχείων.

Δηλαδή η αξιολόγηση της προτεινόμενης μεθοδολογίας πραγματοποιείται σε δύο επίπεδα-κατευθύνσεις. Η μεθοδολογία αξιολογείται σε κάθε στάδιο της διαδικασίας και η τελική αξιολόγηση λαμβάνει χώρα στο τέλος της διαδικασίας μετά από όλες τις φάσεις. Ο σκοπός της αξιολόγησης είναι να εξετάσει τα στοιχεία που χρειάζονται βελτίωση, έτσι ώστε να γίνεται αποτίμηση της διαδικασίας με επιτυχία.

2.2 Ανάπτυξη MOOC: Βασικές αρχές Προγραμματισμού με την ANSI C

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην εφαρμογή της παραπάνω μεθοδολογίας για τη δημιουργία ενός μαζικού ανοιχτού διαδικτυακού μαθήματος MOOC με τίτλο «Βασικές αρχές Προγραμματισμού χρησιμοποιώντας την ANSI C». Για το σκοπό αυτό, δημιουργήθηκαν ομάδες οι οποίες συνεργάστηκαν για την ολοκλήρωση όλων των φάσεων του μοντέλου ADDIE για την ολοκλήρωση και την εφαρμογή του μαθήματος. Η ομάδα Διαχείρισης του διαδικτυακού μαθήματος συμπληρώθηκε, εκτός από την συγγραφέα της πτυχιακής εργασίας, με δύο ακαδημαϊκά μέλη ειδικούς σε θέματα προγραμματισμού. Επίσης διαμορφώθηκε ομάδα Υποστήριξης Εκπαιδευτών με δύο επιστημονικούς συνεργάτες από το Ελληνικό Ανοιχτό Πανεπιστήμιο, έναν εμπειρογνώμονα σε διαδικτυακές υπηρεσίες και έναν επιστημονικό συνεργάτη με επαγγελματικό υπόβαθρο στα παιδαγωγικά. Παρακάτω περιγράφονται συνοπτικά οι φάσεις του μοντέλου ADDIE ειδικά για το μάθημα «Βασικές αρχές Προγραμματισμού χρησιμοποιώντας την ANSI C» ενώ στο τρίτο μέρος της παρούσας εργασίας θα γίνει αναλυτική παρουσίαση των αποτελεσμάτων της φάσης σχεδιασμού, πάνω στα οποία στηρίχτηκαν και υλοποιούνται και στο επόμενο χρονικό διάστημα οι επόμενες φάσεις.

2.2.1 Φάση Ανάλυσης

Οι δύο ομάδες Διαχείρισης του μαθήματος και Εκπαιδευτικής Στήριξης συνεργάστηκαν στη φάση της ανάλυσης, προκειμένου να καθορίσουν την περιγραφή του μαθήματος. Κατευθυντήριες γραμμές για την ανάπτυξη του MOOC δόθηκαν στην ομάδα διαχείρισης, συμπεριλαμβανομένης και της διαδικασίας της μεθοδολογίας. Οι ομάδες επικοινωνούσαν κυρίως μέσω e-mail, συζητώντας τις ιδέες και τα θέματα που αφορούν τη διαδικασία εφαρμογής. Τα αποτελέσματα της φάσης ανάλυσης παρουσιάστηκαν σε πίνακα, πρότυπος του οποίου υπάρχει στο παράρτημα της παρούσας εργασίας (Πίνακας 1).

Ακολουθεί η κύρια περιγραφή του μαθήματος.

Τίτλος Μαθήματος	Βασικές αρχές Προγραμματισμού χρησιμοποιώντας την ANSI C
Περιγραφή	Το μάθημα πραγματεύεται βασικές έννοιες στον προγραμματισμό υπολογιστών και εξετάζει την πρακτική εφαρμογή τους μέσα από την ανάπτυξη προγραμμάτων σε γλώσσα προγραμματισμού C. Το μάθημα παρέχει τα εφόδια για τη διαχείριση της διαδικασίας ανάπτυξης δομημένων προγραμμάτων, τόσο σε θεωρητικό επίπεδο με την παρουσίαση και ανάλυση της τεχνικής του δομημένου προγραμματισμού, όσο και σε πρακτικό επίπεδο με την ανάπτυξη προγραμμάτων χρησιμοποιώντας ένα περιβάλλον ανάπτυξης εφαρμογών. Ένα βασικό στοιχείο του δομημένου προγραμματισμού είναι να οικοδομήσουμε το πρόγραμμα χρησιμοποιώντας υποπρογράμματα, τα οποία είτε εκτελούν γενικές εργασίες είτε αντιμετωπίζουν ένα μέρος του συνολικού προβλήματος.
Γνωστικό Πεδίο	Το γνωστικό πεδίο του μαθήματος είναι ο διαδικασιακός ή δομημένος προγραμματισμός. Η επιλογή της C συνδυάζει μία θεμελιώδη γλώσσα διαδικασιακού προγραμματισμού που είναι προγραμματιστική εμπειρία ενώ ταυτόχρονα αποτελεί το καλύτερο εφαλτήριο για γρήγορη και σε βάθος κατανόηση όλων των υπόλοιπων γλωσσών προγραμματισμού. Ο στόχος είναι να κατανοήσουν οι εκπαιδευόμενοι τις βασικές αρχές του προγραμματισμού και την φιλοσοφίας του, έτσι ώστε να είναι σε θέση, χωρίς δυσκολία να περάσουν σε άλλες προσεγγίσεις προγραμματισμού, όπως ο αντικειμενοστρεφής προγραμματισμός.
Τύπος Μαθήματος	Το μάθημα ορίζεται ως ***Part time*** επειδή απευθύνεται σε φοιτητές, απόφοιτους ή εργαζόμενους.
Συνολικός Χρόνος Μαθήματος	Διάρκεια 6 εβδομάδες
Εκπαιδευτικό πρόβλημα	Ο διαδικασιακός ή προστακτικός προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα ή εργασίες. Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση. Ο διαδικασιακός προγραμματισμός και η εκμάθησή του έχει ως αποτέλεσμα την ανάπτυξη αναλυτικής και συνθετικής σκέψης στους εκπαιδευόμενους, την ικανότητα να επιλύουν απλά προβλήματα σε προγραμματιστικό περιβάλλον, την ανάπτυξη δεξιοτήτων αλγοριθμικής προσέγγισης (ανάλυση προβλήματος, σχεδίαση αλγορίθμου, δομημένη σκέψη, αυστηρότητα έκφρασης) και την ανάπτυξη δημιουργικότητας και φαντασίας.

Χαρακτηριστικά και ανάγκες Εκπαιδευομένων	Η ηλεκτρονική προσβασιμότητα καθορίζει τις πρωτοβουλίες με τις οποίες θα εξασφαλιστεί για όλους τους πολίτες πρόσβαση στις υπηρεσίες της κοινωνίας των πληροφοριών. Βέβαια το συγκεκριμένο μάθημα απευθύνεται σε φοιτητές που πρέπει να έρθουν κοντά στον προγραμματισμό για να υλοποιήσουν εργασίες του τμήματος στο οποίο φοιτούν, εργαζόμενους που χρειάζονται τον προγραμματισμό στην εργασία τους ή και πολίτες που θέλουν να επιμορφωθούν σε ένα αντικείμενο που πάντα ήταν το χόμπι τους. Άρα τα χαρακτηριστικά των εκπαιδευομένων για το μάθημα αυτό είναι το διαφορετικό μορφωτικό επίπεδο και η ανομοιογένεια στις ώρες που αφορούν στον ελεύθερο χρόνο τους.
Έννοιες	Οι βασικές έννοιες που πρόκειται να διδαχθούν είναι η έννοια της εντολής, του προγράμματος και της γλώσσας προγραμματισμού δηλαδή ο εκπαιδευόμενος θα πρέπει να κατανοήσει πώς θα μελετήσει το συντακτικό και τη γραμματική μιας γλώσσας με συγκεκριμένο λεξιλόγιο και σημασιολογία για να επικοινωνήσει αυστηρά με τον ηλεκτρονικό υπολογιστή μέσα από απλές οδηγίες, βήμα - βήμα για να εκτελέσει ο υπολογιστής μια συγκεκριμένη εργασία. Επίσης βασικές έννοιες για τον προγραμματισμό αποτελούν οι έννοιες της μεταβλητής και της σταθεράς, καθώς και τα μέρη που αποτελούν κάθε προγραμματιστικό περιβάλλον.
Ενότητες Μάθησης	Οι ενότητες που θα διδαχθούν είναι*:* 1. Προγραμματιστικά περιβάλλοντα - γλώσσα C 2. Μεταβλητές, σταθερές, τύποι δεδομένων 3. Τελεστές, εκφράσεις, προτάσεις 4. Πίνακες, δείκτες 5. Προτάσεις ελέγχου ροής 6. Αφαιρετικότητα στις διεργασίες
Εκπαιδευτικοί Στόχοι	Στόχοι του μαθήματος είναι: • Η ανάπτυξη ικανοτήτων και η απόκτηση δεξιοτήτων για τη διαχείριση της διαδικασίας επίλυσης προβλημάτων με προγραμματισμό υπολογιστών. • Η κατανόηση των σταδίων του κύκλου ζωής ενός προγράμματος: ανάλυση προβλήματος, σχεδίαση λύσης, κωδικοποίηση της λύσης σε πρόγραμμα, έλεγχος και διόρθωση λαθών, τεκμηρίωση και συντήρηση, απόσυρση του προγράμματος. • Η θεμελίωση αλγοριθμικής σκέψης στην επίλυση προβλημάτων. • Η εμπέδωση βασικών αρχών προγραμματισμού με έμφαση στον δομημένο προγραμματισμό. • Η εκμάθηση της γλώσσας προγραμματισμού C. • Εξοικείωση με εργαλεία ανάπτυξης λογισμικού.

Εργαλεία	Το μάθημα χρησιμοποιεί πλούσιο εκπαιδευτικό υλικό. Αναλυτικότερα θα δημιουργηθούν κυρίως εκπαιδευτικά βίντεο αλλά και επιπλέον εκπαιδευτικό υλικό, όπως παρουσιάσεις, υπερκείμενο, δοκίμια και εργασίες. Το Camtasia Studio είναι ένα από τα καλύτερα προγράμματα που υπάρχουν αυτή τη στιγμή για την παραγωγή και επεξεργασία βίντεο και χρησιμοποιήθηκε αρκετά για τη δημιουργία των βιντεοδιαλέξεων του μαθήματος. Το Office χρησιμοποιήθηκε ανάλογα για τη δημιουργία δοκιμίων και υπερκειμένων, αλλά και για τη δημιουργία παρουσιάσεων.
Απαιτήσεις του Περιβάλλοντος	Το μάθημα αυτό σαν μαζικό ανοιχτό διαδικτυακό μάθημα είναι σχεδιασμένο έτσι ώστε να παραδίνει γνώση σε εκατοντάδες ή χιλιάδες εκπαιδευόμενους μέσω ανοικτής πρόσβασης. Στόχος λοιπόν είναι να δημιουργηθεί ένα περιβάλλον συνεργατικής μάθησης με τη δημιουργία κοινοτήτων μάθησης και πρακτικής όπου ο εκπαιδευόμενος όχι μόνο θα αφομοιώνει αλλά και θα παράγει γνώση. Ένα τέτοιο περιβάλλον μάθησης απαιτεί πρόσθετο πολυμορφικό εκπαιδευτικό υλικό και μηχανισμό υποστήριξης για το ρόλο του διδάσκοντα με στόχο την ενεργητική εμπλοκή των διδασκόμενων στη διεργασία της μάθησης.
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας, των εκπαιδευτών και των εκπαιδευόμενων. Επίσης μπορεί να υπάρξει και ο ρόλος του τεχνικού, ο οποίος εκτός από την υλοποίηση του μαθήματος ηλεκτρονικά, θα απαντάει και στα τεχνικά προβλήματα που μπορεί να προκύψουν με τη χρήση της ηλεκτρονικής πλατφόρμας. Για τους παραπάνω ρόλους δημιουργήθηκαν οι ομάδες Διαχείρισης μαθήματος, Υποστήριξης εκπαιδευτών και Τεχνικής Υποστήριξης.
Χρονοπρογραμματισμός	Προτείνεται ο χρόνος ενασχόλησης των εκπαιδευομένων να είναι ανάμεσα σε 3-4 ώρες την πρώτη εβδομάδα του μαθήματος, 5-6 ώρες την δεύτερη και τρίτη εβδομάδα και περίπου 10 ώρες για τις υπόλοιπες εβδομάδες.

2.2.2 Φάση σχεδιασμού

Οι ίδιες ομάδες συνεργάζονται κατά τη φάση του σχεδιασμού. Το μάθημα αποτελείται από έξι βασικές ενότητες για τις οποίες θα πρέπει να σχεδιαστούν δραστηριότητες και μαθησιακά αντικείμενα. Έχοντας την περιγραφή του μαθήματος μπορούμε να πούμε αναλυτικότερα, ότι οι δραστηριότητες εκμάθησης έχουν σχεδιαστεί σε εβδομαδιαία βάση για τους μαθητές, με βάση τις ενότητες που αναφέρονται παραπάνω. Κάθε ενότητα ξεκινά με μια παρουσίαση, η οποία καθοδηγεί τους μαθητές στο θέμα της ενότητας. Κάθε ενότητα περιέχει τουλάχιστον δύο δραστηριότητες. Το τελικό βήμα της φάσης αυτής ήταν ο σχεδιασμός των μαθησιακών αντικειμένων, τα οποία ορίστηκαν στην περιγραφή δραστηριότητας μάθησης. Κάθε δραστηριότητα αποτελείται από ένα ή περισσότερα αντικείμενα μάθησης. Ανάλογα με το

μαθησιακό αντικείμενο, απαιτείται ένας χρόνος εκμάθησης ανάλογα και με το βαθμό δυσκολίας της άσκησης, ο οποίος ποικίλλει.

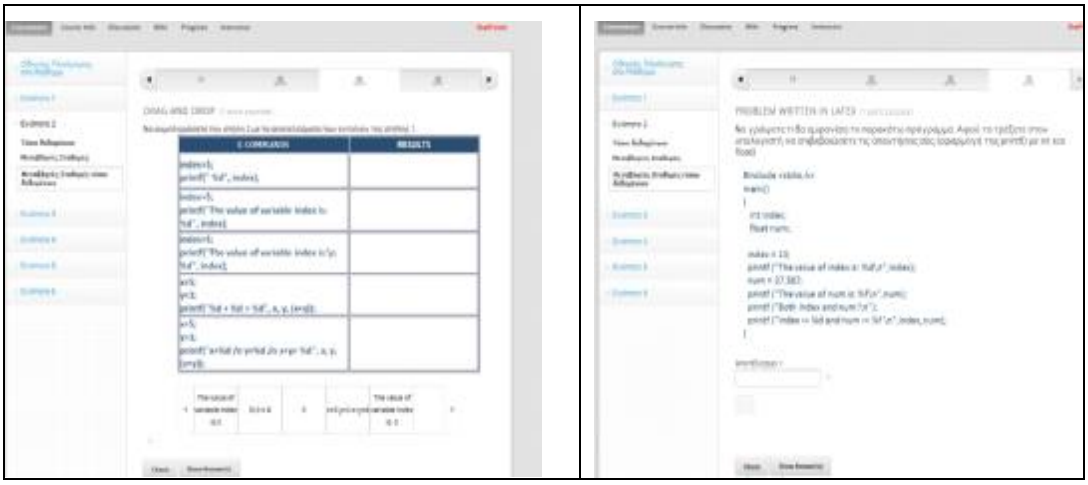
Η φάση του σχεδιασμού αποτελεί το κύριο αντικείμενο της παρούσας εργασίας γι αυτό και θα παρουσιαστεί αναλυτικά στο τρίτο μέρος της εργασίας. Τα αποτελέσματα της φάσης σχεδιασμού θα παρουσιαστούν σε πίνακες, πρότυποι των οποίων υπάρχουν στο παράρτημα της παρούσας εργασίας (Πίνακας 2α, 2β και 2γ).

2.2.3 Φάση Ανάπτυξης

Για το σκοπό αυτό, ένα σύνολο από 200 Μαθησιακά αντικείμενα σχεδιάστηκαν κατά τη διάρκεια της προηγούμενης φάσης. Αυτά τα Μαθησιακά Αντικείμενα αποτελούνταν από εκπαιδευτικά βίντεο, υπερκείμενα, παρουσιάσεις, ασκήσεις αυτοαξιολόγησης (π.χ. Κουίζ, ερωτήσεις πολλαπλής επιλογής, συμπληρώστε τα κενά, σύρε και άφησε) και εργασίες για συγγραφή και εκτέλεση προγραμμάτων. Όσον αφορά τα εκπαιδευτικά βίντεο, ένας έμπειρος προγραμματιστής πολυμέσων βοήθησε την ομάδα Διαχείρισης του μαθήματος για την ανάπτυξη εκπαιδευτικού βίντεο. Οι ασκήσεις αυτοαξιολόγησης αναπτύχθηκαν επίσης από την ομάδα Διαχείρισης του μαθήματος.

Μετά από αυτό, η ομάδα Τεχνικής Υποστήριξης σε συνεργασία με την ομάδα Εκπαιδευτικής Στήριξης ανέβασαν το εκπαιδευτικό υλικό στην πλατφόρμα. Η πλατφόρμα που χρησιμοποιείται είναι το EAP-MOOC, μια πλατφόρμα MOOC που αναπτύχθηκε από το ΕΑΠ με βάση την πλατφόρμα Open EDX.

Επειδή η φάση της ανάπτυξης δεν έχει ολοκληρωθεί μέχρι τη συγγραφή της παρούσας εργασίας θα παρουσιαστούν στιγμιότυπα υλοποίησης των μαθησιακών αντικειμένων στα παρακάτω σχήματα. Σε αυτή τη βάση στην εικόνα 1 παρουσιάζεται η υλοποίηση των μαθησιακών αντικειμένων LA3LO7, LA3LO7 που περιγράφονται αναλυτικά στο στάδιο του σχεδιασμού στο τρίτο κεφάλαιο της παρούσας εργασίας.



Εικόνα 1. Ενδεικτικά Μαθησιακά αντικείμενα στην πλατφόρμα EAP-MOOC

2.2.4 Φάση υλοποίησης

Αυτή η φάση MOOC πρόκειται να ξεκινήσει στο EAP-MOOC. Πριν από την παράδοση του μαθήματος, τρέξαμε ένα πιλοτικό πρόγραμμα, προκειμένου να αντιμετωπίσουμε και να αξιολογήσουμε την πορεία του μαθήματος και την πλατφόρμα. Οι συμμετέχοντες στο στάδιο της δοκιμής ήταν ένας μικρός αριθμός των φοιτητών και έμπειρο επιστημονικό προσωπικό. Η

διαφήμιση του μαθήματος μπορεί να γίνει μέσω των κοινωνικών δικτύων και των ενημερώσεων ηλεκτρονικού ταχυδρομείου. Κατά τη διάρκεια αυτής της φάσης η ομάδα διαχείρισης του μαθήματος συνεργάζεται με την Ομάδα Υποστήριξης Εκπαιδευτών κατά τη διάρκεια της εκπαιδευτικής διαδικασίας. Η Ομάδα Τεχνικής Υποστήριξης παρέχει τεχνική υποστήριξη σε όλη την εκπαιδευτική διαδικασία.

3. ΣΧΕΔΙΑΣΜΟΣ ΜΑΘΗΜΑΤΟΣ

3.1 Ενότητα 1 - Προγραμματιστικά περιβάλλοντα-γλώσσα C

Τίτλος Ενότητας	**S1 -** Προγραμματιστικά περιβάλλοντα-γλώσσα C
Τίτλος Μαθήματος	Βασικές αρχές Προγραμματισμού χρησιμοποιώντας την ANSI C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστούν βασικές έννοιες της τεχνολογίας λογισμικού. Σκοπός της ενότητας είναι να προσδιορίσει το ρόλο των γλωσσών προγραμματισμού στη διαδικασία ανάπτυξης λογισμικού και να κάνει μια σύντομη αναφορά, αφενός μεν στην εξέλιξη των γλωσσών προγραμματισμού, αφετέρου δε στις σημαντικότερες κατηγορίες στις οποίες αυτές κατατάσσονται. Στο τέλος της ενότητας γίνεται αναφορά στη γλώσσα προγραμματισμού C κάνοντας μια ιστορική αναδρομή για τη χρήση της και παρουσιάζοντας τους λόγους επιλογής της ως αντικείμενο του παρόντος μαθήματος.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να ξεχωρίζουν τα υπολογιστικά προβλήματα και τις φάσεις ανάπτυξης λογισμικού για κάθε ένα από αυτά, • να κατανοούν τα προγραμματιστικά περιβάλλοντα και τις διεργασίες που συμβαίνουν μέχρι να εκτελεστεί ένα πρόγραμμα από έναν υπολογιστή, • να κατανοούν τη διαφορά ανάμεσα στον πηγαίο κώδικα και στον εκτελέσιμο κώδικα, • να εξηγούν τη σημασία χρήσης των γλωσσών προγραμματισμού, • να κατηγοριοποιούν τις γλώσσες προγραμματισμού ανάλογα με την χρήση τους, • να εξηγούν τους λόγους επιλογής της γλώσσας C.
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει το κεφάλαιο αυτό, οι εκπαιδευόμενοι θα μπορούν: • να περιγράψουν με δικά τους λόγια τη διαδικασία ανάπτυξης λογισμικού αναφέροντας τις κυριότερες φάσεις της • να προσδιορίσουν το ρόλο που διαδραματίζει μια γλώσσα προγραμματισμού στη διαδικασία αυτή • να αιτιολογήσουν γιατί η φάση της υλοποίησης δεν είναι αρκετή για την ανάπτυξη ενός συστήματος λογισμικού • να χρησιμοποιήσουν τα κατάλληλα εργαλεία για τη συγγραφή πηγαίου κώδικα και την παραγωγή του αντίστοιχου εκτελέσιμου

Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: • Τεχνολογία λογισμικού • Η φάση της υλοποίησης • Μορφές προγραμματισμού και ιστορία γλωσσών προγραμματισμού
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας και των εκπαιδευόμενων.
Αξιολόγηση	Η ενότητα υποστηρίζεται από τρεις δραστηριότητες και καθεμία αξιολογείται με 4 έως 5 μαθησιακά αντικείμενα εκ των οποίων το πρώτο είναι η υποστηριζόμενη θεωρία και τα υπόλοιπα ασκήσεις αυτοαξιολόγησης για την αντίστοιχη θεωρία.
Συνολικός χρόνος Ενότητας	1 εβδομάδα

3.1.1 Δραστηριότητα 1 - Τεχνολογία λογισμικού

Τίτλος Δραστηριότητας	**S1_LA1** - Τεχνολογία λογισμικού
Τίτλος Ενότητας	Προγραμματιστικά περιβάλλοντα - γλώσσα C
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει τη Μηχανική λογισμικού ή τεχνολογία λογισμικού (software engineering), δηλαδή την τυποποιημένη και συστηματική προσέγγιση στην ανάλυση, σχεδίαση, υλοποίηση και συντήρηση λογισμικού ηλεκτρονικών υπολογιστών.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τις φάσεις της διαδικασίας ανάπτυξης λογισμικού, • να αναφέρουν ποιος είναι ο στόχος των μοντέλων του κύκλου ζωής λογισμικού, • να περιγράψουν με μια φράση τι είναι γλώσσα προγραμματισμού, • να δώσουν 6 τουλάχιστον παραδείγματα διεργασιών, • να δώσουν τον ορισμό της υπολογιστικής διεργασίας
Μαθησιακά Αντικείμενα	Φάσεις ανάπτυξης λογισμικού - Μεθοδολογία
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με τα 4 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στο πρώτο μαθησιακό αντικείμενο.
Λέξεις Κλειδιά	Κύκλος ζωής λογισμικού, δομημένη ανάλυση, αλγόριθμος, γλώσσα προγραμματισμού

Μαθησιακό Αντικείμενο	
Όνομα	**S1_LA1LO1**
Τίτλος	**Φάσεις ανάπτυξης λογισμικού**
Τίτλος Εκπαιδευτικής Δραστηριότητας	Τεχνολογία λογισμικού
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι έννοιες που σχετίζονται με την τεχνολογία λογισμικού. Έτσι περιγράφονται οι τρεις φάσεις ανάπτυξης λογισμικού δηλαδή τα στάδια από τα οποία διέρχεται μία εφαρμογή λογισμικού, από τη σύλληψη της ιδέας, τη διαδικασία κατασκευής / ανάπτυξης, τη λειτουργία & συντήρηση / αναβάθμιση, μέχρι την απόσυρσή της. Επίσης ορίζονται οι έννοιες πρόβλημα, αλγόριθμος, πρόγραμμα, διεργασία και γλώσσα προγραμματισμού.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	Αλγόριθμος, πρόγραμμα, διεργασία, κύκλος ζωής λογισμικού, τεχνολογία λογισμικού
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τις βασικές φάσεις ανάπτυξης λογισμικού σε σχέση με ένα δικό τους υπολογιστικό πρόβλημα • να κατανοήσουν τη διαφορά ανάμεσα σε έναν αλγόριθμο και ένα πρόγραμμα • να δώσουν 2 τουλάχιστον παραδείγματα διεργασιών

Καθημερινά, η μία μετά την άλλη οι ανθρώπινες δραστηριότητες αυτοματοποιούνται με τη χρήση υπολογιστών. Οι υπολογιστές αυτοί, γενικού ή ειδικού σκοπού, αυτοματοποιούν πλήρως ή κατά μέρος, όλο και περισσότερες εργασίες του ανθρώπου. Απλές εργασίες, όπως συγγραφή κειμένου, έκδοση κοινοχρήστων αλλά και πιο σύνθετες, όπως πρόβλεψη καιρού και έλεγχος τηλεπικοινωνιακών κέντρων, υποστηρίζονται πλέον από υπολογιστές με τα κατάλληλα συστήματα λογισμικού. Επιχειρήσεις, αλλά ακόμη και εθνικά αμυντικά συστήματα, βασίζουν την ύπαρξή τους σε ειδικά συστήματα λογισμικού, απαιτώντας από αυτά αυξημένη αξιοπιστία. Η ανάπτυξη συστημάτων λογισμικού έχει γίνει πλέον μια πολύ σύνθετη διαδικασία.

Η επιστήμη της Τεχνολογίας Λογισμικού είναι η εφαρμογή επιστημονικών αρχών για το μεθοδικό μετασχηματισμό ενός προβλήματος σε μια λειτουργούσα λύση λογισμικού και για τη μετέπειτα συντήρησή του για το διάστημα που αυτό είναι χρήσιμο. Η επιστήμη αυτή ορίζει αφενός μεν ένα σύνολο από μοντέλα κύκλου ζωής λογισμικού, αφετέρου δε ένα σύνολο από μεθοδολογίες.

Τα μοντέλα κύκλου ζωής λογισμικού περιγράφουν τον κύκλο ζωής λογισμικού, τη διαδικασία, δηλαδή, η οποία αρχίζει από τη σύλληψη της ιδέας και τελειώνει όταν το προϊόν δεν είναι πλέον διαθέσιμο για χρήση. Οι φάσεις, όπως τις ορίζει η πλειοψηφία των μοντέλων, είναι οι εξής:

Φάση Ανάλυσης (Analysis), η οποία περιλαμβάνει την κατανόηση του προβλήματος και καταγραφή των απαιτήσεων από το σύστημα.

Πρωταρχικός στόχος της ανάλυσης είναι να διαπιστωθεί κατά πόσον το πρόβλημα, που ζητείται να λυθεί, είναι επιλύσιμο με υπολογιστή. Στη φάση της ανάλυσης περιγράφονται οι απαιτήσεις της εφαρμογής, επισημαίνονται οι ιδιαιτερότητες, απομακρύνονται οι επουσιώδεις λεπτομέρειες που περιπλέκουν τη λύση, σκιαγραφείται ο τρόπος λύσης, υποδεικνύονται τα σημεία στα οποία απαιτείται ιδιαίτερη προσοχή και περιγράφονται οι συνθήκες κάτω από τις οποίες είναι δυνατή η υλοποίηση. Ακόμα αναλύονται στοιχεία που αφορούν το κόστος. Το προϊόν της φάσης αυτής είναι μία κατανοητή, πλήρης και σαφής περιγραφή χαρακτηριστικών και συμπεριφοράς της εφαρμογής ή αλλιώς προδιαγραφές της εφαρμογής.

Φάση Σχεδιασμού (Design), η οποία περιλαμβάνει το σχεδιασμό μιας λειτουργούσας λύσης.

Σ’ αυτή τη φάση το πρόβλημα χωρίζεται σε επιμέρους προβλήματα, που είναι δυνατόν να επιλυθούν πιο εύκολα και καθορίζεται η μεταξύ τους συσχέτιση. Δημιουργείται ένα πλάνο λύσης. Στο πλάνο λύσης για κάθε πρόβλημα επινοείται, αν δεν υπάρχει, ή επιλέγεται ένας από τους γνωστούς αλγόριθμους που το επιλύει. Επίσης, οργανώνονται τα δεδομένα σε δομές δεδομένων και σχεδιάζονται οι φόρμες - διεπαφές της εφαρμογής. Το προϊόν της φάσης αυτής είναι το πλήρες σχέδιο υλοποίησης της εφαρμογής.

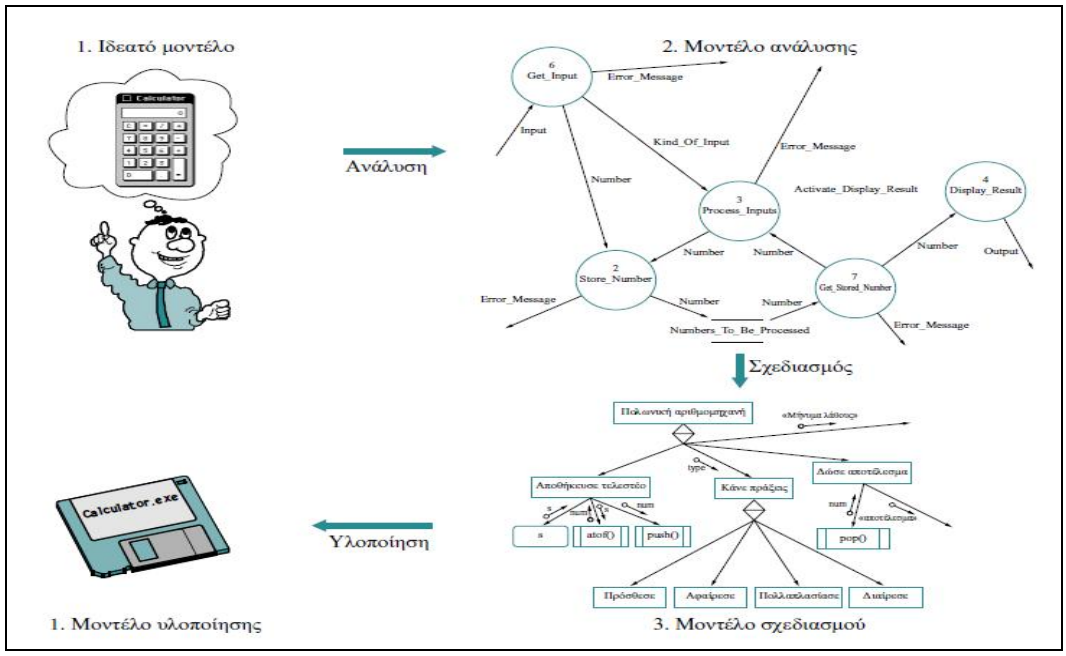
Φάση Υλοποίησης (Implementation), η οποία περιλαμβάνει την ανάπτυξη του εκτελέσιμου κώδικα.

Αντίθετα, μια μεθοδολογία ορίζει τα επιμέρους βήματα που πρέπει να εκτελέσει ο δημιουργός λογισμικού σε μια συγκεκριμένη συνήθως φάση της διαδικασίας ανάπτυξης. Για παράδειγμα, η μεθοδολογία της Δομημένης Ανάλυσης (Structured Analysis/SA) ορίζει τον τρόπο με τον οποίο προσεγγίζουμε τη φάση της κατανόησης και καταγραφής απαιτήσεων του συστήματος.

Έλεγχος προδιαγραφών και διόρθωση λαθών

Η φάση αυτή ονομάζεται και φάση δοκιμών. Στη φάση αυτή γίνονται οι έλεγχοι κατά πόσο το λογισμικό που έχει παραχθεί συμφωνεί με τις προδιαγραφές της ανάλυσης και ανιχνεύονται και διορθώνονται κατά το δυνατόν τα υπάρχοντα λάθη.

Τα παραπάνω οδηγούν στην αναπαράσταση της διαδικασίας ανάπτυξης λογισμικού όπως αυτή δίνεται στην παρακάτω εικόνα. Σύμφωνα με την αναπαράσταση αυτή, η διαδικασία ξεκινά από το ονομαζόμενο ιδεατό μοντέλο (conceptual model) και ολοκληρώνεται με τη δημιουργία του μοντέλου υλοποίησης (implementation model). Το ιδεατό μοντέλο είναι αυτό που έχει στο μυαλό του ο χρήστης πριν αρχίσει τη διαδικασία ανάπτυξης του συστήματος, ενώ το μοντέλο υλοποίησης είναι σε μορφή που μπορεί να εκτελέσει ο υπολογιστής και αυτή δεν είναι άλλη από τη γλώσσα μηχανής (machine language). Η μετάβαση από το ιδεατό μοντέλο στο μοντέλο υλοποίησης γίνεται σταδιακά και με την παρεμβολή δύο άλλων μοντέλων: του μοντέλου ανάλυσης και του μοντέλου σχεδιασμού.



Εικόνα 2. Διαδικασία ανάπτυξης λογισμικού

Το υπολογιστικό πρόβλημα που καλούμαστε να επιλύσουμε μπορεί να είναι π.χ., ένας αριθμητικός υπολογισμός, η διαχείριση μιας βάσης δεδομένων, ο έλεγχος ενός ρομπότ, μια ιατρική διάγνωση, η πρόγνωση του καιρού. Υπάρχουν προβλήματα που δεν είναι υπολογιστικά όπως π.χ. το πρόβλημα του καθαρισμού των ρούχων που δεν μπορεί να ανατεθεί σε έναν υπολογιστή.

Πρόβλημα - παράδειγμα

• Έχουμε ένα αρχείο με τα ονόματα των φοιτητών.

• Θέλουμε να βρούμε τους φοιτητές του 1ου έτους

Σχεδιασμός μιας λύσης (δηλαδή αλγόριθμος)

Αλγόριθμος είναι μια σαφώς καθορισμένη διαδικασία αποτελούμενη από ένα σύνολο εκτελέσιμων βημάτων που θα τερματίσει με ένα αποτέλεσμα μετά από πεπερασμένο πλήθος βημάτων που θα εκτελεστούν. Με βάση τον ορισμό του αλγορίθμου φαίνεται ότι ένας αλγόριθμος έχει τα εξής χαρακτηριστικά: είσοδο, έξοδο, καθοριστικότητα, αποτελεσματικότητα και περατότητα. Τέλος όσον αφορά στο παραπάνω πρόβλημα τα βήματα μπορεί να είναι:

1. Απόκτησε πρόσβαση στο αρχείο των φοιτητών.

2. Διάβασε την πρώτη γραμμή του αρχείου που περιέχει τα στοιχεία του πρώτου φοιτητή.

3. Αν είναι πρωτοετής φοιτητής τύπωσε το όνομά του.

4. Αν υπάρχει επόμενη γραμμή στο αρχείο διάβασέ την και επανάλαβε τα βήματα 3 έως 4.

• Πρόγραμμα (Program)

Ορίζεται μια κωδικοποίηση του αλγόριθμου σε μια γλώσσα προγραμματισμού.

Γλώσσα προγραμματισμού λέγεται μια τεχνητή γλώσσα που μπορεί να χρησιμοποιηθεί για τον έλεγχο μιας μηχανής, συνήθως ενός υπολογιστή. Οι γλώσσες προγραμματισμού (όπως άλλωστε και οι ανθρώπινες γλώσσες) ορίζονται από ένα σύνολο συντακτικών και εννοιολογικών κανόνων, που ορίζουν τη δομή και το νόημα των προτάσεων της γλώσσας.

• Διεργασία (Process)

Διεργασία (process) είναι ένας όρος της πληροφορικής ο οποίος περιγράφει το στιγμιότυπο ενός προγράμματος που εκτελείται σε έναν υπολογιστή. Σε αντιδιαστολή με την έννοια του προγράμματος, το οποίο είναι ένα στατικό σύνολο εντολών, μια διεργασία συνιστά την εκτέλεση αυτών των εντολών. Επομένως ένα πρόγραμμα γενικώς συσχετίζεται με περισσότερες από μία διεργασίες, μία για κάθε φορά που εκτελείται.

Όταν η διεργασία πρόκειται να εκτελεστεί από έναν υπολογιστή, η περιγραφή θα πρέπει να γίνει σε μία σημειογραφία που άμεσα ή έμμεσα κατανοεί ο υπολογιστής. Άμεσα κατανοεί μόνο τη γλώσσα μηχανής, αλλά η περιγραφή μιας διεργασίας με τη γλώσσα αυτή είναι σχεδόν αδύνατη για τα σημερινά, πολύπλοκα και αυξημένων απαιτήσεων, συστήματα. Επομένως, η μόνη επιλογή είναι αυτή της έμμεσης κατανόησης. Λέμε ότι ο υπολογιστής κατανοεί έμμεσα τις γλώσσες προγραμματισμού γιατί μια περιγραφή με αυτές μπορεί αυτόματα να μεταγλωττιστεί σε γλώσσα μηχανής και, άρα, να γίνει κατανοητή από τον υπολογιστή. Μπορούμε να πούμε λοιπόν, ότι μια **γλώσσα προγραμματισμού** είναι μία συστηματική σημειογραφία (notation) με την οποία περιγράφουμε υπολογιστικές διεργασίες.

Για την ανάπτυξη των προγραμμάτων που χρησιμοποιούνται ως παραδείγματα, θα ξεκινάμε αμέσως τη συγγραφή του πηγαίου κώδικα χωρίς να ασχολούμαστε καθόλου, στις περισσότερες περιπτώσεις, με τα μοντέλα ανάλυσης και σχεδιασμού. Και αυτό γιατί αφενός μεν στόχος αυτού του μαθήματος είναι ο προγραμματισμός, αφετέρου δε δεν έχετε ακόμη τις απαραίτητες γνώσεις για να δημιουργήσετε τα μοντέλα ανάλυσης και σχεδιασμού. Επιπλέον, τα παραδείγματά μας είναι συνήθως απλά και δεν απαιτούν από τη φύση τους την επισταμένη δημιουργία των μοντέλων αυτών.

Ερωτήσεις αντικειμενικού τύπου - S1_LA1LO2

Επιλέξτε την σωστή απάντηση για τις παρακάτω ερωτήσεις:

1. Όλα τα προβλήματα είναι υπολογιστικά.
2. Η φάση Σχεδιασμού περιλαμβάνει την ανάπτυξη του εκτελέσιμου κώδικα.
3. Το μοντέλο υλοποίησης είναι σε μορφή που μπορεί να εκτελέσει ο υπολογιστής και αυτή δεν είναι άλλη από τη γλώσσα μηχανής.
4. Η φάση Ανάλυσης περιλαμβάνει την κατανόηση του προβλήματος και καταγραφή των απαιτήσεων από το σύστημα.
5. Η γλώσσα προγραμματισμού χρησιμοποιείται για να μετατρέψουμε ένα πρόγραμμα σε έναν αλγόριθμο.
6. Μια **γλώσσα προγραμματισμού** είναι μία συστηματική σημειογραφία με την οποία περιγράφουμε υπολογιστικές διεργασίες.
7. Ένα από τα χαρακτηριστικά του αλγορίθμου είναι η περατότητα.
8. Ο υπολογιστής άμεσα κατανοεί μόνο τη γλώσσα μηχανής.
9. Ο προγραμματισμός σε γλώσσα μηχανής είναι μια εξαιρετικά δύσκολη δουλειά που ελάχιστοι μπορούν να πραγματοποιήσουν.
10. Διεργασία (process) είναι ένας όρος της πληροφορικής, ο οποίος περιγράφει το στιγμιότυπο ενός προγράμματος που εκτελείται σε έναν υπολογιστή.

Άσκηση αξιολόγησης - S1_LA1LO3

Βάλτε στη σωστή σειρά τα παρακάτω μοντέλα έτσι όπως χρησιμοποιούνται κατά την ανάπτυξη λογισμικού:

Μοντέλο υλοποίησης, μοντέλο ανάλυσης, ιδεατό μοντέλο, μοντέλο σχεδιασμού.

Άσκηση αξιολόγησης - S1_LA1LO4

Βάλτε στη σωστή σειρά τα παρακάτω βήματα έτσι όπως χρησιμοποιούνται κατά την ανάπτυξη ενός προγράμματος:

1. Ανάλυση του προβλήματος και προσδιορισμός της λύσης. Ένα πρόβλημα μπορεί να έχει πολλές λύσεις.

2. Μετάφραση και εκτέλεση του προγράμματος στον υπολογιστή.

3. Διατύπωση της λύσης υπό τη μορφή αλγόριθμου.

4. Διατύπωση του προβλήματος με σαφήνεια. Ένα πρόβλημα δεν μπορεί να επιλυθεί ορθά αν δεν γίνει πρώτα κατανοητό.

5. Έλεγχος και διόρθωση συντακτικών και λογικών λαθών.

6. Κωδικοποίηση του αλγόριθμου. Επιλογή γλώσσας προγραμματισμού.

Άσκηση συμπλήρωσης κενών - S1_LA1LO5

Αλγόριθμος είναι μια σαφώς ……. διαδικασία αποτελούμενη από ένα σύνολο ……. βημάτων που θα τερματίσει με ένα αποτέλεσμα μετά από ……..…. πλήθος βημάτων που θα εκτελεστούν.

Μπορούμε να πούμε λοιπόν, ότι μια ……. ……. είναι μία συστηματική σημειογραφία (notation) με την οποία περιγράφουμε υπολογιστικές ……..

3.1.2 Δραστηριότητα 2 - Η φάση υλοποίησης

Τίτλος Δραστηριότητας	**S1_LA2 -** Η φάση υλοποίησης
Τίτλος Ενότητας	Προγραμματιστικά περιβάλλοντα-γλώσσα C
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει ένα προγραμματιστικό περιβάλλον. Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού, για να εκτελεστεί σε έναν υπολογιστή πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής. Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Επίσης χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το αποτέλεσμα θα είναι ένα πρόγραμμα που θα μπορεί τελικά να εκτελέσει ο υπολογιστής.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τα βασικά βήματα για τη δημιουργία του εκτελέσιμου κώδικα, • να αναφέρουν ποια εργαλεία χρησιμοποιούνται στη διαδικασία αυτή, • να εξηγήσουν το ρόλο της βιβλιοθήκης στη διαδικασία αυτή.
Μαθησιακά Αντικείμενα	Τα βήματα της υλοποίησης
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με τα 5 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στο πρώτο μαθησιακό αντικείμενο.
Λέξεις Κλειδιά	Προγραμματιστικό περιβάλλον, μεταγλωττιστής, συνδέτης, πηγαίος κώδικας, εκτελέσιμος κώδικας

Μαθησιακό Αντικείμενο	
Όνομα	**S1_LA2LO6**
Τίτλος	**Τα βήματα της υλοποίησης**
Τίτλος Εκπαιδευτικής Δραστηριότητας	S1_LA2 - Η φάση υλοποίησης
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται όλες οι διεργασίες που εκτελούνται σ' ένα προγραμματιστικό περιβάλλον, από τη στιγμή που ο προγραμματιστής θα γράψει τον κώδικα σε έναν editor του λογισμικού της γλώσσας που χρησιμοποιεί, μέχρι την εκτέλεση του προγράμματος.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	Συντάκτης, μεταγλωττιστής, συνδέτης, βιβλιοθήκες, συντακτικά λάθη, λογικά λάθη, πηγαίο πρόγραμμα, εκτελέσιμο πρόγραμμα
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τα βασικά βήματα για τη δημιουργία του εκτελέσιμου κώδικα, • να αναφέρουν ποια εργαλεία χρησιμοποιούνται στη διαδικασία αυτή, • να εξηγήσουν το ρόλο της βιβλιοθήκης στη διαδικασία αυτή.

Η φάση της υλοποίησης που περιγράψαμε παραπάνω, μπορεί να θεωρηθεί σαν μία σύνθετη διεργασία, η οποία λαμβάνει σαν είσοδο το μοντέλο σχεδιασμού και έχει σαν έξοδο τον εκτελέσιμο κώδικα. Η διεργασία αυτή μπορεί να ανατεθεί στον προγραμματιστή, ο οποίος χρησιμοποιεί τη γνώση της γλώσσας προγραμματισμού, τις γενικότερες γνώσεις του για προγραμματισμό και τον υπολογιστή ως εργαλείο για να τον υποστηρίξει στο δύσκολο αυτό έργο.

Τα επιμέρους βήματα που πρέπει να ακολουθήσει ο προγραμματιστής, για τη δημιουργία του εκτελέσιμου κώδικα είναι:

1. συγγραφή πηγαίου κώδικα

2. μεταγλώττιση πηγαίου κώδικα

3. σύνδεση τμημάτων που αποτελούν το μοντέλο υλοποίησης (εκτελέσιμος κώδικας)

4. έλεγχος καλής λειτουργίας

Συγγραφή πηγαίου κώδικα

Στο πρώτο βήμα, ο προγραμματιστής επιλέγοντας μία από τις διαθέσιμες γλώσσες προγραμματισμού και χρησιμοποιώντας έναν επεξεργαστή κειμένου (text editor), δημιουργεί ένα αρχείο που περιέχει τον **πηγαίο κώδικα** του προγράμματός του. Η C επιβάλει την αποθήκευση του πηγαίου κώδικα σε ένα αρχείο με επέκταση .c. Στα πλαίσια του παραδείγματός μας, γράψτε τον παρακάτω πηγαίο κώδικα χωρίς να ασχοληθείτε με τη σημασία του. Αποθηκεύστε τον στο αρχείο example.c.

```
/* A Simple C Program */
#include <stdio.h>
main()
{
    printf("hello, world");
}
```

Μεταγλώττιση

Μεταγλώττιση είναι η διαδικασία της μετάφρασης μιας υψηλού επιπέδου γλώσσας προγραμματισμού, όπως Basic, Pascal, C, σε γλώσσα μηχανής που αποτελεί μορφή κατανοητή από τον υπολογιστή. Η μεταγλώττιση αποτελεί μια σύνθετη και χρονοβόρο για τον άνθρωπο διεργασία και για το λόγο αυτό την ανέθεσε στον υπολογιστή.

Η διεργασία της μεταγλώττισης είναι πλήρως αυτοματοποιημένη. Ο προγραμματιστής αρκεί να ενεργοποιήσει τον μεταγλωττιστή ενημερώνοντάς τον με το όνομα του αρχείου που θέλει να μεταγλωττίσει και με ένα σύνολο από παραμέτρους που καθορίζουν τις απαιτήσεις του από τον μεταγλωττιστή. Η ενεργοποίηση του μεταγλωττιστή γίνεται απλά εκτελώντας το πρόγραμμα του μεταγλωττιστή, όπως το λειτουργικό σας σύστημα ορίζει. Σε ένα γραφικό περιβάλλον, όλα εκτελούνται συνήθως μέσα από τις επιλογές ενός μενού που παρέχει το περιβάλλον ανάπτυξης. Αντίθετα, σε ένα παραδοσιακό περιβάλλον με γραμμή εντολής (command line) η ενεργοποίηση του μεταγλωττιστή έχει τη μορφή:

<όνομα μεταγλωττιστή> [<λίστα παραμέτρων>] <όνομα αρχείου>

Για παράδειγμα, για το μεταγλωττιστή της WATCOM στο MSDOS, η εντολή wcc example ενεργοποιεί το μεταγλωττιστή, ο οποίος μεταγλωττίζει το αρχείο example.c και, σε περίπτωση μη ανεύρεσης λαθών, τυπώνει στην οθόνη τα παρακάτω:

```
WATCOM C Optimizing Compiler Version 8.5e
Copyright by WATCOM Systems Inc. 1984, 1991. All rights reserved.
WATCOM is a trademark of WATCOM Systems Inc.
example.c: 4 lines, 0 warnings, 0 errors
Code size: 17
```

και παράγει το αρχείο example.obj.

Αντίθετα, στην περίπτωση ανεύρεσης λαθών, το αποτέλεσμα της διεργασίας της μεταγλώττισης είναι μια λίστα με τα λάθη που εντόπισε ο μεταγλωττιστής. Τα λάθη αυτά έχουν σχέση με το συντακτικό της γλώσσας και είναι γνωστά σαν **συντακτικά λάθη** (syntax errors). Παρακάτω, δίνεται η αναφορά του μεταγλωττιστή για την περίπτωση που λείπει η αγκύλη στο τέλος του πηγαίου κώδικα.

example.c (4): Error! E1077: Missing '}'

example.c: 4 lines, 0 warnings, 1 errors

Στην περίπτωση αυτή, αφού εντοπίσετε τα λάθη, με τη βοήθεια της πληροφορίας που αναφέρει ο μεταγλωττιστής, θα πρέπει χρησιμοποιώντας τον text editor να τα διορθώσετε. Στη συνέχεια, επαναλαμβάνετε τη διαδικασία της μεταγλώττισης μέχρι την επιτυχή έκβασή της.

Εκτός από τα συντακτικά λάθη θα πρέπει να εντοπιστούν και τα **λογικά λάθη**. Αυτά οφείλονται στην κακή απόδοση της λύσης του προβλήματος και ανιχνεύονται μετά το χρόνο εκτέλεσης βλέποντας τα λανθασμένα αποτελέσματα. Σ' αυτή την περίπτωση ο προγραμματιστής θα πρέπει να διορθώσει τα προβληματικά σημεία του αλγορίθμου.

Διεργασία της σύνδεσης

Κατά τη συγγραφή προγραμμάτων ο προγραμματιστής έχει τη δυνατότητα να επαναχρησιμοποιεί (reuse) έτοιμα κομμάτια κώδικα, τα οποία είτε έχει ο ίδιος αναπτύξει και χρησιμοποιήσει στο παρελθόν είτε προσφέρονται από άλλους δημιουργούς λογισμικού. Τα έτοιμα αυτά τμήματα κώδικα είναι συνήθως με τη μορφή μονάδων (συναρτήσεων, κλάσεων) με την κάθε μονάδα να εκτελεί μια συγκεκριμένη διεργασία, η οποία όσο γενικότερη είναι τόσο μεγαλύτερες είναι οι πιθανότητες επαναχρησιμοποίησής της. Οι μονάδες αυτές είναι οργανωμένες κατά λογικές κατηγορίες, με κάθε λογική κατηγορία να συνθέτει μια βιβλιοθήκη που χαρακτηρίζεται από ένα όνομα. Κάθε μεταγλωττιστής της C συνοδεύεται από τη βασική **βιβλιοθήκη** της γλώσσας (standard C library), η οποία περιέχει ένα σύνολο από συναρτήσεις που υλοποιούν πολύ βασικές διεργασίες. Μια τέτοια συνάρτηση είναι η printf, η οποία υλοποιεί τη διεργασία της εξόδου μορφοποιημένης πληροφορίας στην κύρια έξοδο του υπολογιστή (συνήθως οθόνη). Στη C, για να χρησιμοποιήσετε μια συνάρτηση της βασικής βιβλιοθήκης είναι απαραίτητο να συμπεριλάβετε στον πηγαίο κώδικα ένα ή περισσότερα αρχεία επικεφαλίδας (header files), όπως ορίζει το εγχειρίδιο χρήσης της βιβλιοθήκης για κάθε συνάρτησή της. Για παράδειγμα, για τη χρήση της printf πρέπει να περιλάβετε στην αρχή του πηγαίου κώδικα την πρόταση #include <stdio.h>.

Η διεργασία της σύνδεσης με τις βιβλιοθήκες της γλώσσας που ακολουθεί είναι πλήρως αυτοματοποιημένη και εκτελείται από τον υπολογιστή με τη βοήθεια του προγράμματος του **συνδέτη** (Linker). Ο προγραμματιστής θα πρέπει να ενεργοποιήσει το συνδέτη και να ορίσει τις παραμέτρους της διεργασίας της σύνδεσης. Στα περισσότερα περιβάλλοντα ανάπτυξης, ο συνδέτης καλείται αυτόματα μετά από επιτυχή μεταγλώττιση χωρίς τη μεσολάβηση του προγραμματιστή.

Το αποτέλεσμα της σύνδεσης είναι η δημιουργία του εκτελέσιμου κώδικα ή η αναφορά τυχόν προβλημάτων, όπως για παράδειγμα αδυναμία εντοπισμού μιας συνάρτησης ή μιας εξωτερικής μεταβλητής. Ο εκτελέσιμος κώδικας αποθηκεύεται σε αρχείο που έχει το όνομα του αρχείου πηγαίου κώδικα και επέκταση .exe (για Dos και Windows).

Έτσι, για το παραπάνω πρόγραμμα ο συνδέτης παράγει το αρχείο example.exe, το οποίο αποτελεί το μοντέλο υλοποίησης, ή αλλιώς, τον εκτελέσιμο κώδικα του προγράμματος, τον οποίο μπορείτε να εκτελέσετε για να διαπιστώσετε την ορθή λειτουργία του. Η πρόταση

example στη γραμμή διαταγών του DOS δίνει εντολή στο λειτουργικό να φορτώσει και εκτελέσει το αρχείο example.exe. Στην οθόνη θα έχετε το παρακάτω αποτέλεσμα:

hello, world

Τα πιθανά λάθη που θα εμφανιστούν, εντοπίζονται, διορθώνονται και η διεργασία μεταγλώττισης και σύνδεσης επαναλαμβάνεται μέχρι την επιτυχή λειτουργία του προγράμματος.

Ερωτήσεις αντικειμενικού τύπου - S1_LA2LO7

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Βιβλιοθήκη της γλώσσας (standard C library) λέγεται ένα σύνολο από συναρτήσεις που υλοποιούν πολύ βασικές διεργασίες.
2. Ο μεταγλωττιστής μας επιτρέπει να συντάσσουμε ένα πρόγραμμα.
3. Το εκτελέσιμο πρόγραμμα είναι ουσιαστικά γλώσσα μηχανής.
4. Ο συνδέτης είναι ένα πρόγραμμα ελέγχου των συντακτικών λαθών του πηγαίου προγράμματος.
5. Ο μεταγλωττιστής έχει το μειονέκτημα ότι ελέγχει όλο το πρόγραμμα και πραγματοποιεί και την διαδικασία της σύνδεσης πολλές φορές μέχρι να επιδιορθωθούν όλα τα λάθη.
6. Ο μεταγλωττιστής σ' ένα σύγχρονο προγραμματιστικό περιβάλλον καθιστά την ύπαρξη του συνδέτη προαιρετική.
7. Ένα ορθογραφικό λάθος σε μια εντολή είναι ένα λογικό λάθος.
8. Για την επιδιόρθωση των λογικών λαθών πολλές φορές ο προγραμματιστής καλείται να εκτελέσει το πρόγραμμά του επανειλημμένα.
9. Τα λογικά λάθη ενός προγράμματος εμφανίζονται κατά τη μεταγλώττιση.
10. Το εκτελέσιμο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή.
11. Ένας αλγόριθμος θα πρέπει να είναι γενικός δηλαδή να μην τελειώνει ποτέ.
12. Ο μεταγλωττιστής εξάγει το πηγαίο πρόγραμμα.
13. Ο εντοπισμός των συντακτικών λαθών σε ένα πρόγραμμα γίνεται από τον μεταγλωττιστή (compiler).

Ερωτήσεις συμπλήρωσης κενού - S1_LA2LO8

Συμπληρώστε τα κενά στις παρακάτω προτάσεις:

1. Κάθε πρόγραμμα γλώσσας υψηλού επιπέδου μεταφράζεται σε γλώσσα μηχανής από ένα ειδικό πρόγραμμα που ονομάζεται _____.
2. Η λανθασμένη γραφή των δεσμευμένων λέξεων της γλώσσας προγραμματισμού είναι _____ λάθος.
3. Τα _____ λάθη δεν είναι δυνατόν να εντοπίζονται από το προγραμματιστικό περιβάλλον.

Ερωτήσεις πολλαπλής επιλογής - S1_LA2LO9

1. Από τον συντάκτη παράγεται:

   α) τα αντικείμενα

   β) ο εκτελέσιμος κώδικας

   γ) ο πηγαίος κώδικας

   δ) τίποτα από τα παραπάνω

Άσκηση αυτοαξιολόγησης - S1_LA2LO10

Τοποθετήστε τις παρακάτω διεργασίες που έχουν σχέση με την ανάπτυξη

και εκτέλεση ενός προγράμματος σε σωστή χρονική σειρά.

1. αποθήκευση πηγαίου κώδικα
2. σύνδεση
3. αναφορά λαθών από μεταγλωττιστή
4. εκτέλεση προγράμματος
5. συγγραφή πηγαίου κώδικα
6. μεταγλώττιση
7. εμφάνιση αποτελεσμάτων προγράμματος
8. αναφορά λαθών από συνδέτη

Άσκηση αυτοαξιολόγησης - S1_LA2LO11

Συμπλήρωση κενών

Αν σε κάποια οδηγία έχουμε κάνει λάθος στο αλφάβητο, στο λεξιλόγιο ή στο …… τότε το πρόγραμμα που μετατρέπει τις οδηγίες μας σε σειρά από 0 και 1 θα μας δώσει το κατάλληλο μήνυμα λάθους, ώστε να μας βοηθήσει να διορθώσουμε το λάθος μας. Τα λάθη αυτά ονομάζονται ……. λάθη. Αν το αποτέλεσμα που τελικά προκύπτει από την εκτέλεση του προγράμματος δεν είναι το αναμενόμενο, τότε το πρόβλημα δεν βρίσκεται στον τρόπο εκτέλεσης, αλλά στον …….. που κατασκευάσαμε για την επίλυση του προβλήματός μας. Στην περίπτωση αυτή λέμε ότι έχουμε κάνει ένα …….. λάθος.

3.1.3 Δραστηριότητα 3 - Μορφές προγραμματισμού και ιστορία γλωσσών

Τίτλος Δραστηριότητας	**S1_LA3 -** Μορφές προγραμματισμού και ιστορία γλωσσών
Τίτλος Ενότητας	Προγραμματιστικά περιβάλλοντα - γλώσσα
Εκπαιδευτική Στρατηγική	*Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης*.
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει τις μορφές προγραμματισμού κατηγοριοποιώντας τις γλώσσες προγραμματισμού σε προστακτικές, οντοκεντρικές, συναρτησιακές και λογικοκεντρικές γλώσσες. Στη συνέχεια γίνεται μια ιστορική αναδρομή των γλωσσών προγραμματισμού αναφέροντας τις τέσσερις γενιές και τις γλώσσες που ανήκουν σε κάθε μια από αυτές, αναφέροντας και τις εφαρμογές χρήσης τους. Τέλος γίνεται αναφορά στην ιστορία της γλώσσας C, συγκεκριμένα αναφέροντας τους λόγους επιλογής της ως αντικείμενο του συγκεκριμένου μαθήματος.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν 4 τουλάχιστον μορφές προγραμματισμού, • να περιγράψουν τη διαφορά μεταξύ προστακτικού και δηλωτικού προγραμματισμού, • να αναφέρουν τουλάχιστον 4 γλώσσες που υποστηρίζουν την προστακτική μορφή προγραμματισμού, • να αναφέρουν τη βασική διαφορά μεταξύ των γλωσσών Pascal και Prolog, • να αναφέρουν 6 τουλάχιστον λόγους για τους οποίους επιλέχθηκε να χρησιμοποιηθεί η C στην εργασία αυτή
Μαθησιακά Αντικείμενα	Μορφές προγραμματισμού Ιστορία των γλωσσών προγραμματισμού Ιστορία της C
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 8 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για την θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα LA3LO12, LA3LO15 και LA3LO19.
Λέξεις Κλειδιά	Προστακτικός ή διαδικασιακός προγραμματισμός, αντικειμενοστραφής ή οντοκεντρικός προγραμματισμός, συναρτησιακός προγραμματισμός

Μαθησιακό Αντικείμενο	
Όνομα	**S1_LA3LO12**
Τίτλος	**Μορφές προγραμματισμού**
Τίτλος Εκπαιδευτικής Δραστηριότητας	S1_LA3 - Μορφές προγραμματισμού και ιστορία γλωσσών
Περιγραφή	Το συγκεκριμένο ΜΑ παρουσιάζει τις μορφές προγραμματισμού κατηγοριοποιώντας τις γλώσσες προγραμματισμού σε προστακτικές, οντοκεντρικές, συναρτησιακές και λογικοκεντρικές γλώσσες. Στην πληροφορική καλούμε προστακτικό προγραμματισμό (Αγγλικά: Imperative programming), ένα προγραμματιστικό υπόδειγμα όπου το ζητούμενο κατασκευάζεται αλλάζοντας την κατάσταση του υπολογιστή μέσω εντολών. Το υπόδειγμα αυτό ακολουθούν οι διαδικαστικές γλώσσες προγραμματισμού, όπως η Pascal, η C, η Fortran, κ.ά., αλλά και πολλές αντικειμενοστρεφείς γλώσσες όπως η Java, η C++, η C#, κ.ά. Επίσης καλούμε δηλωτικό προγραμματισμό όταν το ζητούμενο αποτέλεσμα υπολογίζεται περιγράφοντας απλώς τις επιθυμητές ιδιότητές του. Παραδείγματα γλωσσών δηλωτικού προγραμματισμού είναι η Haskell, η Prolog, η SQL και η CSS.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Τεχνικός Τύπος	Υπερκείμενο
Λέξεις Κλειδιά	Προστακτικός ή διαδικασιακός προγραμματισμός, αντικειμενοστρεφής ή οντοκεντρικός προγραμματισμός, συναρτησιακός προγραμματισμός
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν 4 τουλάχιστον μορφές προγραμματισμού • να περιγράψουν τη διαφορά μεταξύ προστακτικού και δηλωτικού προγραμματισμού • να αναφέρουν τουλάχιστον 4 γλώσσες που υποστηρίζουν την προστακτική μορφή προγραμματισμού

Μορφή ή στυλ προγραμματισμού (programming paradigm/programming style) είναι μια συλλογή από έννοιες, οι οποίες προσδιορίζοντας έναν ορισμένο τρόπο σκέψης και, άρα,

έκφρασης της λύσης, επηρεάζουν το σχεδιασμό των προγραμμάτων. Εάν μπορούμε να δομήσουμε τη λύση ενός προβλήματος με τις βασικές έννοιες μιας μορφής προγραμματισμού, τότε μόνο μπορούμε να χρησιμοποιήσουμε μια γλώσσα προγραμματισμού που υποστηρίζει τη συγκεκριμένη μορφή προγραμματισμού για να υλοποιήσουμε τη λύση. Ορισμένες γλώσσες εισήγαγαν νέα στυλ προγραμματισμού, νέους δηλαδή τρόπους σκέψης στον προγραμματισμό.

Ας δούμε κάποιες κατηγορίες γλωσσών προγραμματισμού έτσι όπως παρουσιάζονται στο ηλεκτρονικό βιβλίο «ΣΥΓΧΡΟΝΕΣ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ - Μια πρακτική εισαγωγή», Μετάφραση − επιστημονική επιμέλεια: Γεώργιος Φρ. Γεωργακόπουλος, Ιωάννης Παπαδόγγονας.

Προστακτικές γλώσσες

Ας δούμε ένα παράδειγμα προστακτικής γλώσσας, της γλώσσας C. Πρόκειται για μια συνάρτηση factorial(n), που υπολογίζει το παραγοντικό (factorial) ενός φυσικού αριθμού n:

```
int factorial(int n) {
    int sofar = 1;
    while (n > 0) sofar *= n--;
    return sofar;
}
```

Το παραπάνω παράδειγμα περιέχει τις δύο σφραγίδες γνησιότητας κάθε προστακτικής γλώσσας: την **τιμοδότηση** και την **επανάληψη**. Η εντολή sofar *= n--; της γλώσσας C τιμοδοτεί τη μεταβλητή sofar. Η μεταβλητή αυτή έχει κάποια τρέχουσα τιμή η οποία μεταβάλλεται κάθε φορά που πραγματοποιείται μια τιμοδότηση. Η εντολή επιδρά επίσης στη μεταβλητή n, μειώνοντας κάθε φορά την τιμή της κατά μία μονάδα. Ο βρόχος while (εντολή επανάληψης) επαναλαμβάνει διαρκώς την εντολή. Τελικά, σε κάποιο βήμα αυτής της επανάληψης, η τρέχουσα τιμή της μεταβλητής n θα γίνει μηδέν και η επανάληψη θα σταματήσει. Καθώς οι τιμές των μεταβλητών αλλάζουν σε κάθε βήμα, η σειρά εκτέλεσης των εντολών του προγράμματος έχει καίρια σημασία.

Οι έννοιες που μόλις αναφέραμε είναι τόσο στοιχειώδεις, που περνούν απαρατήρητες από τους περισσότερους προγραμματιστές της γλώσσας C. Για αυτούς είναι προφανές ότι η σειρά εκτέλεσης των εντολών παίζει καίριο ρόλο και εξίσου προφανές ότι οι τιμές των μεταβλητών αλλάζουν. Υπάρχουν όμως πολλές γλώσσες προγραμματισμού για τις οποίες όλα τα παραπάνω δεν έχουν κανένα νόημα. Γλώσσες στις οποίες δεν υπάρχουν ούτε τιμοδοτικές εντολές, ούτε επαναληπτικές εντολές, ούτε η έννοια της αλλαγής της τρέχουσας τιμής μιας μεταβλητής.

Οντοστρεφείς γλώσσες

Η συνάρτηση του παραγοντικού γραμμένη στη γλώσσα Java φαίνεται σχεδόν ίδια με την εκδοχή της στη γλώσσα C. Η Java όμως είναι μια οντοστρεφής γλώσσα, πράγμα που σημαίνει ότι αφ' ενός είναι προστακτικού τύπου και αφ' ετέρου έχει σχεδιαστεί ώστε να διευκολύνει την επίλυση των διαφόρων προβλημάτων μέσω οντοτήτων (ή αλλιώς αντικειμένων). Ονομάζουμε οντότητα μια (συνήθως μικρή) δέσμη δεδομένων η οποία γνωρίζει πώς να χειρίζεται τον εαυτό της. Για παράδειγμα, ας δούμε τον ορισμό σε Java μιας

οντότητας που φέρει έναν ακέραιο αριθμό και γνωρίζει πώς να αναφέρει τόσο τον αριθμό αυτό, όσο και το παραγοντικό του.

```
public class MyInteger {
    private int value;
    public MyInteger(int value) {
        this.value = value;
    }
    public int getValue() {
        return value;
    }
    public MyInteger getFactorial() {
        return new MyInteger(factorial(value));
    }
    private int factorial(int n) {
        int sofar = 1;
        while (n > 1) sofar *= n--;
        return sofar;
    }
}
```

Το παραπάνω παράδειγμα οντοστρεφούς προγραμματισμού φαίνεται φλύαρο σε σχέση με τα προηγούμενα, αλλά και πάλι η σύγκριση δεν είναι δίκαιη: ο οντοστρεφής προγραμματισμός έχει σχεδιαστεί ώστε να διευκολύνει την οργανωμένη σύνταξη προγραμμάτων πολύ μεγάλου μεγέθους και ως εκ τούτου δεν φανερώνει τα προτερήματά του σε παραδείγματα μικρού μεγέθους.

Συναρτησιακές γλώσσες

Ας δούμε την ίδια συνάρτηση (του παραγοντικού) υλοποιημένη στη γλώσσα ML:

```
fun factorial x =
if x <= 0 then 1 else x * factorial(x-1);
```

Το παραπάνω παράδειγμα περιλαμβάνει δύο από τις σφραγίδες γνησιότητας των συναρτησιακών γλωσσών: την **αναδρομή** και τις **μονότιμες μεταβλητές**. Η αναδρομή είναι μια προγραμματιστική τεχνική τόσο φυσική στους προγραμματιστές της ML, όσο φυσικές είναι οι επαναληπτικές εντολές στους προγραμματιστές της C.

Η ίδια συνάρτηση υλοποιημένη στη γλώσσα Lisp θα είχε ως εξής:

```
(defun factorial (x)
(if (<= x 0) 1 (* x (factorial (- x 1)))))
```

Όπως βλέπετε, η Lisp έχει ιδιόρρυθμη σύνταξη. Αυτή η συντακτική διαφορά είναι όμως επιφανειακή. Σε βαθύτερο επίπεδο, η συνάρτηση factorial γραμμένη στη Lisp και η

συνάρτηση factorial γραμμένη στην ML σχετίζονται μεταξύ τους πολύ περισσότερο απ' όσο σχετίζεται η καθεμία από αυτές με τη συνάρτηση factorial γραμμένη στη γλώσσα C. Και οι δύο είναι γραμμένες στο συναρτησιακό ύφος, χωρίς τιμοδοτικές ή επαναληπτικές εντολές.

Τα δύο παραπάνω παραδείγματα ίσως φαίνονται πιο κομψά από την εκδοχή της C, αλλά μια τέτοια σύγκριση δεν είναι δίκαιη. Το συναρτησιακό είδος προγραμματισμού ταιριάζει ιδιαίτερα σε συναρτήσεις όπως αυτή του παραγοντικού. Σε άλλα είδη προβλημάτων, όπως π.χ. ο πολλαπλασιασμός πινάκων, το πλεονέκτημα θα το είχαν οι προστακτικές γλώσσες προγραμματισμού.

Λογικοκεντρικές γλώσσες

Η συνάρτηση του παραγοντικού, ενώ είναι το καταλληλότερο παράδειγμα για τη γλώσσα ML, είναι ίσως το χειρότερο για την Prolog. Παρά ταύτα ας δούμε τι μορφή έχει στην Prolog:

```
factorial(X,1) :-
X =:= 1.
factorial(X,F) :-
X > 1,
NewX is X - 1,
factorial(NewX, NewF),
F is X * NewF.
```

Οι πρώτες δύο γραμμές εκφράζουν έναν κανόνα που επιτρέπει στο σύστημα της Prolog να συμπεράνει ότι όταν το X ισούται με 1 το παραγοντικό του X είναι 1. Οι υπόλοιπες πέντε γραμμές κώδικα καθορίζουν έναν γενικό τρόπο για να διαπιστώνει κανείς ότι το παραγοντικό του X ισούται με κάποια δεδομένη τιμή.

Συγκεκριμένα:

Για να αποδείξεις ότι το παραγοντικό του X ισούται με F, αρκεί να κάνεις τα εξής:

- να αποδείξεις ότι το X είναι μεγαλύτερο του 1.
- να αποδείξεις ότι το NewX είναι μικρότερο του X κατά 1.
- να αποδείξεις ότι το παραγοντικό του NewX ισούται με NewF και, τέλος,
- να αποδείξεις ότι το F ισούται με X επί NewF.

Η διατύπωση ενός προγράμματος μέσω κανόνων λογικού συμπερασμού είναι η σφραγίδα γνησιότητας του λογικού προγραμματισμού. Αν και αυτό το είδος προγραμματισμού δεν είναι το καταλληλότερο για τον υπολογισμό μαθηματικών συναρτήσεων, υπάρχουν κατηγορίες προβλημάτων στα οποία υπερέχει ξεκάθαρα.

Στην πράξη, μία γλώσσα προγραμματισμού σπάνια υποστηρίζει μια μόνο μορφή προγραμματισμού. Συνήθως, δανείζεται έννοιες από περισσότερες μορφές και σαν αποτέλεσμα, υποστηρίζει περισσότερα από ένα στυλ προγραμματισμού. Για παράδειγμα, η C++ υλοποιεί βασικές έννοιες από το προστακτικό και το αντικειμενοστρεφές παράδειγμα. Μία μορφή προγραμματισμού που, τα τελευταία χρόνια, γνωρίζει μεγάλη εξάπλωση γιατί υπόσχεται να δώσει (και ήδη δίνει) λύσεις σε πολλά από τα προβλήματα της διαδικασίας ανάπτυξης λογισμικού και όχι μόνο, είναι η Αντικειμενοστρεφής Προσέγγιση.

Ερωτήσεις αντικειμενικού τύπου - S1_LA3LO13

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Μια γλώσσα προγραμματισμού που χαρακτηρίζεται από την τιμοδότηση μεταβλητών και την επανάληψη είναι προστακτική γλώσσα.
2. Δεν υπάρχουν γλώσσες χωρίς την έννοια της αλλαγής της τρέχουσας τιμής μιας μεταβλητής.
3. Για τους περισσότερους από τους προγραμματιστές της γλώσσας C είναι προφανές ότι η σειρά εκτέλεσης των εντολών παίζει καίριο ρόλο.
4. Ο πολλαπλασιασμός πινάκων θα μπορούσε να δοθεί κομψότερα από μια συναρτησιακή γλώσσα προγραμματισμού.
5. Η συνάρτηση του παραγοντικού, ενώ είναι το καταλληλότερο παράδειγμα για τη γλώσσα ML, είναι ίσως το χειρότερο για την Prolog.
6. Δύο από τις σφραγίδες γνησιότητας των συναρτησιακών γλωσσών είναι η αναδρομή και τις τιμοδοτικές εντολές.
7. Η Java είναι μια οντοστρεφής γλώσσα όπως και η Prolog.
8. Στην πράξη, μία γλώσσα προγραμματισμού σπάνια υποστηρίζει μια μόνο μορφή προγραμματισμού.
9. Σε βαθύτερο επίπεδο, η γλώσσα Lisp και η ML σχετίζονται μεταξύ τους πολύ περισσότερο απ' όσο σχετίζεται η καθεμία από αυτές με τη γλώσσα C.
10. Ο οντοστρεφής προγραμματισμός έχει σχεδιαστεί ώστε να διευκολύνει την οργανωμένη σύνταξη προγραμμάτων πολύ μεγάλου μεγέθους.
11. Οι λογικοκεντρικές γλώσσες προγραμματισμού δεν προσφέρονται για τον υπολογισμό μαθηματικών συναρτήσεων.
12. Η Lisp είναι μια λογικοκεντρική γλώσσα προγραμματισμού.

Άσκηση αυτοαξιολόγησης - S1_LA3LO14

Να αντιστοιχίσετε τις παρακάτω γλώσσες προγραμματισμού με τις αντίστοιχες μορφές προγραμματισμού:

Γλώσσα προγραμματισμού	Είδος προγραμματισμού
Prolog	προστακτικός
Pascal, basic, C	οντοκεντρικός
Java, C++	συναρτησιακός
Lisp, ML	λογικοκεντρικός

Μαθησιακό Αντικείμενο	
Όνομα	**S1_LA3LO15**
Τίτλος	**Ιστορία των γλωσσών προγραμματισμού**
Τίτλος Εκπαιδευτικής Δραστηριότητας	S1_LA3 - Μορφές προγραμματισμού και ιστορία γλωσσών
Περιγραφή	Στο συγκεκριμένο ΜΑ, γίνεται αναφορά στις γλώσσες προγραμματισμού από τότε που δημιουργήθηκαν μέχρι σήμερα ενώ παράλληλα περιγράφεται το είδος χρήσης τους και ο τομέας εφαρμογής για τις πιο βασικές από αυτές.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	FORTRAN, COBOL, ALGOL, PL/1, LISP, PROLOG, BASIC, PASCAL, C, C++, JAVA
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τη βασική διαφορά μεταξύ των γλωσσών Pascal και Prolog • να αναφέρουν παραδείγματα εφαρμογών για τις γλώσσες προγραμματισμού που γνωρίζετε • να κατανοήσουν τις διαφορές μεταξύ των γλωσσών προγραμματισμού και πού πλεονεκτούν οι γλώσσες υψηλού επιπέδου σε σχέση με τις γλώσσες προηγούμενης γενιάς

Αν θέλαμε να κατηγοριοποιήσουμε τις γλώσσες προγραμματισμού θα λέγαμε ότι υπάρχουν οι παρακάτω 4 γενιές:

Γλώσσες μηχανής ή 1η γενιά

Ένα πρόγραμμα σε γλώσσα μηχανής είναι μια ακολουθία δυαδικών ψηφίων που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες. Οι εντολές αυτές είναι κατανοητές από τον υπολογιστή αλλά ακατανόητες από τον άνθρωπο καθώς απαιτούν βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Συμβολικές γλώσσες ή 2η γενιά

Μια συμβολική γλώσσα ενώ έχει έννοια για τον άνθρωπο μετατρέπεται εσωτερικά από τον υπολογιστή στις αντίστοιχες ακολουθίες από 0 και 1. Το έργο της μετάφρασης αναλαμβάνει ένα ειδικό πρόγραμμα ο συμβολομεταφραστής.

Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονόματα που αντιστοιχούν σε εντολές σε γλώσσα μηχανής

Τα μειονεκτήματα των συμβολικών γλωσσών είναι τα εξής:

- Παραμένουν συνδεδεμένες στενά με την αρχιτεκτονική του υπολογιστή.
- Δεν διαθέτουν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμματα που είναι δύσκολο να γραφούν και κυρίως να συντηρηθούν
- Δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολογιστή ακόμη και του ίδιου κατασκευαστή

Γλώσσες υψηλού επιπέδου ή 3η γενιά

Οι γλώσσες υψηλού επιπέδου χρησιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας ακολουθώντας αυστηρούς κανόνες σύνταξης. Οι εντολές αυτές μεταφράζονται από τον ίδιο τον υπολογιστή σε εντολές σε γλώσσα μηχανής. Οι κυριότερες γλώσσες υψηλού επιπέδου είναι οι εξής:

FORTRAN, COBOL, ALGOL, PL/1, LISP, PROLOG, BASIC, PASCAL, C, C++, JAVA

Η πρώτη γλώσσα υψηλού επιπέδου, η FORTRAN, αναπτύχθηκε το 1957 ως γλώσσα κατάλληλη για την επίλυση μαθηματικών προβλημάτων και χρησιμοποιείται ακόμη και σήμερα για επιστημονικές εφαρμογές. Η FORTRAN υστερεί στη διαχείριση αρχείων δεδομένων και γενικότερα αλφαριθμητικών πληροφοριών.

Μεταξύ των γλωσσών που υιοθετήθηκαν σε μεγάλο βαθμό θα πρέπει να αναφέρουμε και την COBOL (Common Business Oriented Language) με μεγάλη απήχηση στον επιχειρηματικό κόσμο που εξελίχθηκε σε γενικού σκοπού γλώσσα. Το 1960 αναπτύχθηκε ως γλώσσα κατάλληλη για ανάπτυξη εμπορικών συναλλαγών. Η COBOL καθιερώθηκε ως πρότυπο και χρησιμοποιήθηκε από πολλές επιχειρήσεις και από όλη τη δημόσια διοίκηση. Πολλές εφαρμογές βρίσκονται σε χρήση ακόμη και σήμερα.

Η διάδοχος της FORTRAN, η ALGOL, κυριάρχησε στη δεκαετία του ’60 σε βαθμό που η κατηγορία των προστακτικών γλωσσών να αναφέρεται σαν ALGOL οικογένεια, από όπου και ο όρος ALGOL-like. Στην πράξη, η γλώσσα περισσότερο θαυμάστηκε παρά υιοθετήθηκε, αντίθετα με τις απογόνους της, Pascal και C, που υιοθετήθηκαν ευρέως. Η γλώσσα ALGOL επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες. Αναπτύχθηκε με σκοπό τη δημιουργία προγραμμάτων γενικής φύσης που να μην συνδέονται με συγκεκριμένες εφαρμογές.

Η γλώσσα PL/1 προσπάθησε ανεπιτυχώς να αντικαταστήσει τη FORTRAN και την COBOL.

Στο χώρο της τεχνητής νοημοσύνης κυριαρχούν δυο διαφορετικές γλώσσες, η LISP και η PROLOG.

Οι επόμενες γλώσσες, Pascal και C, υιοθετήθηκαν ευρέως. Αμφότερες εμφανίστηκαν στις αρχές της δεκαετίας του ’70 και, η μεν Pascal σχεδιάστηκε σαν εκπαιδευτική γλώσσα από τον Nicklaus Wirth, η δε C σαν γλώσσα προγραμματισμού συστήματος (system programming) από τον Dennis Ritchie. Και οι δύο εξελίχθηκαν σε γενικού σκοπού (general purpose)

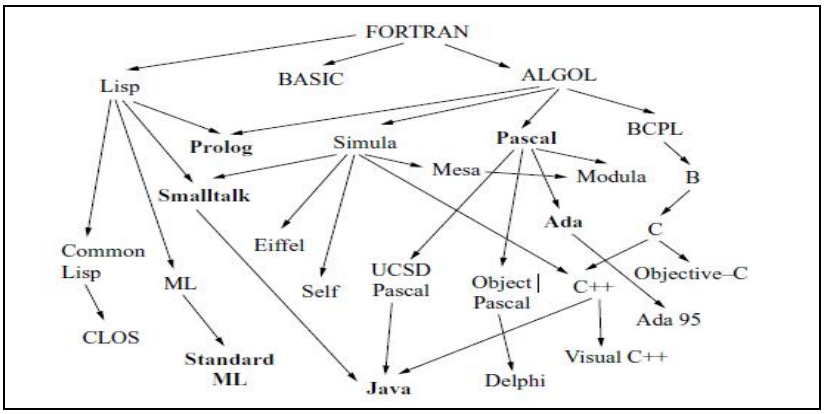
προστακτικές γλώσσες και επηρέασαν σε μεγάλο βαθμό επόμενες γλώσσες προστακτικού προγραμματισμού μεταξύ των οποίων οι Modula, Concurrent Pascal και Ada (από την Pascal) και C++ και Java (από την C).

Η γλώσσα προγραμματισμού BASIC αναπτύχθηκε ως γλώσσα για την εκπαίδευση αρχαρίων στον προγραμματισμό. Η ανάπτυξη των μικροϋπολογιστών και οι συνεχείς εκδόσεις της, την κατέστησαν την δημοφιλέστερη, ίσως γλώσσα στους προσωπικούς υπολογιστές.

Η γλώσσα PASCAL στηρίχτηκε πάνω στην ALGOL. Είναι γλώσσα γενικής χρήσης, κατάλληλη για εκπαίδευση αρχαρίων αλλά και για τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Είναι κατάλληλη για τη δημιουργία δομημένων προγραμμάτων. Γνωρίζει τεράστια εξάπλωση στο χώρο των μικροϋπολογιστών.

Η γλώσσα C γνώρισε μεγάλη διάδοση και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος UNIX. Είναι γλώσσα κατάλληλη για ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Έχει εξελιχθεί στη γλώσσα C++ που είναι αντικειμενοστρεφής.

Τα τελευταία χρόνια χρησιμοποιείται ειδικά για προγραμματισμό στο διαδίκτυο η γλώσσα JAVA. Η JAVA είναι αντικειμενοστρεφής γλώσσα που αναπτύχθηκε με σκοπό την ανάπτυξη εφαρμογών που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές που είναι συνδεδεμένοι στο διαδίκτυο. Τα προγράμματα αυτά μπορούν να εκτελούνται χωρίς αλλαγές από διαφορετικούς υπολογιστές προσωπικούς ή μεγάλα συστήματα με διαφορετικά λειτουργικά συστήματα.



Εικόνα 3. Γενεαλογικό δέντρο γλωσσών προγραμματισμού

Το παραπάνω σχήμα της εικόνας 3 παρουσιάζει ένα τμήμα του γενεαλογικού δένδρου των γλωσσών προγραμματισμού, όπου μπορείτε να δείτε τις σημαντικότερες γλώσσες προγραμματισμού και τον τρόπο με τον οποίο αυτές επηρέασαν την εξέλιξη των υπολοίπων.

Γλώσσες 4ης γενιάς

Οι γλώσσες 4ης γενιάς, αντίθετα από τις γλώσσες 3ης γενιάς, είναι γλώσσες εφοδιασμένες με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο χρήστης μπορεί να επιλύει μόνος του μικρά προβλήματα εφαρμογών. Στις γλώσσες αυτές μπορεί ο χρήστης ενός υπολογιστή σχετικά εύκολα να υποβάλει ερωτήσεις στο σύστημα ή να αναπτύσσει εφαρμογές που αποκτούν πληροφορίες

από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών.

Ερωτήσεις αντικειμενικού τύπου- S1_LA3LO16

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Ένα πρόγραμμα σε γλώσσα μηχανής χρειάζεται μετατροπή σε ακολουθία δυαδικών ψηφίων ώστε να εκτελεστεί από τον υπολογιστή
2. Ο προγραμματισμός σε γλώσσα μηχανής ήταν μια εξαιρετικά δύσκολη δουλειά που ελάχιστοι μπορούσαν να πραγματοποιήσουν
3. Ένα πρόγραμμα σε συμβολική γλώσσα ή γλώσσα χαμηλού επιπέδου τελικά μετατρέπεται σε γλώσσα μηχανής
4. Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονόματα που αντιστοιχούν σε εντολές της γλώσσας μηχανής
5. Τα προγράμματα σε γλώσσες υψηλού επιπέδου είναι ανεξάρτητα του υπολογιστή που αναπτύχθηκαν
6. Η γλώσσα προγραμματισμού Fortran είναι κατάλληλη για την επίλυση όλων των ειδικών προβλημάτων
7. Η Fortran είναι κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων
8. Οι γλώσσες 4ης γενιάς χρησιμοποιούνται σε εφαρμογές που χρησιμοποιούν βάσεις δεδομένων
9. Η επιλογή της καλύτερης γλώσσας προγραμματισμού εξαρτάται από το είδος της εφαρμογής
10. Δεν έχει αναπτυχθεί γλώσσα υψηλού επιπέδου που να επιλύει όλα τα είδη προβλημάτων
11. Η COBOL είναι γλώσσα προσανατολισμένη στην ανάπτυξη εφαρμογών τεχνητής νοημοσύνης
12. Η COBOL δεν μπορεί να επιλύσει μαθηματικά προβλήματα
13. Η Algol είναι μια γλώσσα γενικού σκοπού αλλά με ελάχιστη πρακτική εφαρμογή
14. Η Lisp και η Prolog είναι γλώσσες προγραμματισμού που χρησιμοποιούνται στον τομέα της Τεχνητής Νοημοσύνης
15. Η Basic είναι γλώσσα γενικού σκοπού με έμφαση στην εκπαίδευση αρχαρίων στον προγραμματισμό
16. Η C χρησιμοποιήθηκε για την ανάπτυξη συστημάτων και έχει πολλές δυνατότητες χαμηλού επιπέδου
17. Ένα πλεονέκτημα των συμβολικών γλωσσών προγραμματισμού είναι η μεταφερσιμότητα των προγραμμάτων
18. Η καλύτερη γλώσσα προγραμματισμού είναι η Pascal

Άσκηση επιλογής - S1_LA3LO17

Η Basic είναι:

Α. Κατάλληλη για εφαρμογές τεχνητής νοημοσύνης

Β. Υποστηρίζει την ανάπτυξη παράλληλου προγραμματισμού

Γ. Μία γλώσσα γενικής χρήσης

Δ. Κατάλληλη μόνο για εκπαίδευση

Άσκηση αντιστοίχησης - S1_LA3LO18

Να συνδέσετε τα στοιχεία της στήλης Α με τα στοιχεία της στήλης Β

ΟΝΟΜΑ ΓΛΩΣΣΑΣ	ΤΟΜΕΑΣ ΕΦΑΡΜΟΓΩΝ
1. Fortran	I. Επιστημονικός
2. Cobol	II. Εμπορικός
3. Algol	III. Επιστημονικός και Εμπορικός
4. Prolog	IV. Προγραμματισμός συστημάτων
5. Lisp	V. Προγραμματισμός στο διαδίκτυο
6. Pascal	VI. Γενικής χρήσης
7. Basic	VII. VII. Τεχνητής νοημοσύνης
8. C	
9. C++	
10. Java	
11. PL/1	

Μαθησιακό Αντικείμενο	
Όνομα	**S1_LA3LO19**
Τίτλος	**Ιστορία της C**
Τίτλος Εκπαιδευτικής Δραστηριότητας	S1_LA3 - Μορφές προγραμματισμού και ιστορία γλωσσών
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται χρονικά η διαδρομή της ανάπτυξης της γλώσσας προγραμματισμού C και η εξάπλωση σε όλο και περισσότερους τομείς εφαρμογών μέχρι σήμερα. Ιδιαίτερα, χρησιμοποιείται σήμερα η C, η οποία, αφενός μεν κυριάρχησε την περασμένη δεκαετία, αφ' ετέρου δε αποτέλεσε τη βάση για γλώσσες που κυριαρχούν (C++) ή διαφαίνεται ότι θα κυριαρχήσουν (Java) την επόμενη δεκαετία.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	Γλώσσα C, Αμερικανικό Εθνικό Ίδρυμα Προτυποποίησης (ANSI), ANSI Standard C
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν 3 τουλάχιστον λόγους επιλογής της γλώσσας C ως αντικείμενο αυτού του μαθήματος • να αναφέρουν παραδείγματα εφαρμογών για τα οποία θα επέλεγαν την γλώσσα C σαν την πιο κατάλληλη

Η C είναι από τις πλέον διαδεδομένες γλώσσες προγραμματισμού στον κόσμο. Γι' αυτό, είναι η καλύτερη επιλογή για όσους θέλουν να μάθουν πώς να προγραμματίζουν. Η γλώσσα C αποτελεί γερή βάση για την είσοδο στον κόσμο του προγραμματισμού. Αν μπορείτε να προγραμματίσετε σε C τότε μπορείτε εύκολα να μεταπηδήσετε σε άλλες παρεμφερείς γλώσσες όπως η C++, η Java και η C# με σχετική ευκολία.

Η γλώσσα C είναι γλώσσα προγραμματισμού συστημάτων, ενώ σχεδιάστηκε και αναπτύχθηκε το 1972 στα AT&T Bell Labs από τον Dennis Ritchie. Η C έγινε ιδιαίτερα δημοφιλής κυρίως για το λιτό τρόπο σύνταξης και την ευκολία εκμάθησής της. Βασίστηκε στην BCPL [M. Richard, 1967] αλλά και στην απόγονο της B [Thompson '72], την οποία ανέπτυξε ο Ken Thompson και το διάστημα εκείνο χρησιμοποιούσε στην ανάπτυξη της νέας έκδοσης του λειτουργικού συστήματος UNIX. Ο Dennis Ritchie, προσθέτοντας και

αφαιρώντας στοιχεία από την B με στόχο να την απομακρύνει από το υλικό, δημιούργησε μια νέα γλώσσα που την ονόμασε C (η γλώσσα μετά την B).

Η C, όντας ευέλικτη και αποδοτική, χρησιμοποιήθηκε το 1973 για να ξαναγραφεί το μεγαλύτερο τμήμα του UNIX σε έναν PDP-11. Στη συνέχεια, και για χρόνια, χρησιμοποιήθηκε αποκλειστικά για system programming στο UNIX. Η πρώτη επίσημη τεκμηρίωση της γλώσσας έκανε την εμφάνισή της μόλις το 1977, με τίτλο «The C Programming Language» από τους Brian Kernighan και Dennis Ritchie. Η τεκμηρίωση αυτή αποτέλεσε για χρόνια το «ευαγγέλιο» των προγραμματιστών της C και είναι γνωστό σαν «white book» ή K&R πρότυπο [Kernighan ’78].

Είναι γλώσσα μετρίου επιπέδου, ακολουθεί τη φιλοσοφία του διαδικαστικού προγραμματισμού και δεν προσφέρει δυνατότητες αντικειμενοστρέφειας. Παρ’ όλα αυτά, ο απόλυτος έλεγχος που δίδεται στον προγραμματιστή σε συνδυασμό με την ευκολία εκμάθησης την καθιέρωσαν ως μια πολύ καλή γλώσσα προγραμματισμού.

Με την πάροδο των χρόνων, η γλώσσα άρχισε να χρησιμοποιείται και σε άλλα πεδία εφαρμογών εκτός του system programming για το οποίο σχεδιάστηκε, κατακτώντας ένα πολύ μεγάλο μέρος της αγοράς, με αποτέλεσμα να θεωρείται στις αρχές τις δεκαετίας του ’90 μία από τις επικρατέστερες γλώσσες. Σε αυτό, συνέτειναν και τα πολλά πλεονεκτήματά της, όσον αφορά την ευελιξία, αποδοτικότητα, φορητότητα και ταχύτητα εκτέλεσης, σε σύγκριση με άλλες ανάλογες γλώσσες όπως Fortran, Basic, Pascal.

Μετά από αρκετές προσθήκες και τροποποιήσεις της αρχικής έκδοσης, η C τυποποιήθηκε από το Αμερικανικό Εθνικό Ίδρυμα Προτυποποίησης (ANSI) το 1983 και προέκυψε η ANSI Standard C. Με ελάχιστες εξαιρέσεις, κάθε σύγχρονη εκδοχή της C ακολουθεί τα πρότυπα και της προδιαγραφές της ANSI C.

Γιατί γλώσσα C;

Η επιλογή της C για τη διδασκαλία της προστακτικής μορφής προγραμματισμού, έγινε για ένα σύνολο από λόγους που παρουσιάζονται παρακάτω:

• Είναι σχετικά μικρή και εύκολη στην εκμάθηση.

• Υποστηρίζει top-down και modular σχεδιασμό αλλά και δομημένο (structured) προγραμματισμό.

• Είναι αποτελεσματική (efficient), παράγοντας συμπαγή και γρήγορα στην εκτέλεση προγράμματα.

• Είναι φορητή (portable), ευέλικτη (flexible) και ισχυρή (powerful).

• Δε βάζει περιορισμούς, γεγονός πάντως που πολλές φορές αποβαίνει σε βάρος της.

• Αποτελεί με την C++ την ευρύτερα χρησιμοποιούμενη γλώσσα σε ερευνητικά και αναπτυξιακά προγράμματα.

- Υπάρχει μια πολύ μεγάλη εγκατεστημένη βάση εφαρμογών που αναπτύχθηκαν με τη γλώσσα αυτή και πρέπει να συντηρούνται και να εξελίσσονται.

• Η C μπορεί να χρησιμοποιηθεί σαν χαμηλού επιπέδου γλώσσα προγραμματισμού επιτρέποντας άμεση πρόσβαση στους πόρους του υπολογιστή και άρα στην αποτελεσματική και χωρίς overhead αξιοποίησή τους. Ταυτόχρονα, μπορεί να χρησιμοποιηθεί και σαν γλώσσα υψηλού επιπέδου, καθώς η πληθώρα των διαθέσιμων βιβλιοθηκών υπερκαλύπτει τις απαιτήσεις ανάπτυξης λογισμικού εφαρμογής (Application Software).

• Η γνώση της C αποτελεί ένα πολύ καλό εφόδιο για την εκμάθηση της Java.

Ερωτήσεις συμπλήρωσης κενού - S1_LA3LO20

1. Η γλώσσα C σχεδιάστηκε και αναπτύχθηκε τη δεκαετία του ……. στα AT&T Bell Labs από τον Dennis Ritchie.
2. Η γλώσσα C υποστηρίζει top-down και modular σχεδιασμό αλλά και …… (structured) προγραμματισμό.
3. Η γλώσσα C μπορεί να χρησιμοποιηθεί σαν ……. επιπέδου γλώσσα προγραμματισμού αλλά ταυτόχρονα, μπορεί να χρησιμοποιηθεί και σαν γλώσσα ……. επιπέδου.
4. Η γλώσσα C είναι ισχυρή γλώσσα ανάπτυξης συστημάτων με δυνατότητες ……..… επιπέδου.
5. Μετά από αρκετές προσθήκες και τροποποιήσεις της αρχικής έκδοσης, η C τυποποιήθηκε από το Αμερικανικό Εθνικό Ίδρυμα Προτυποποίησης (…..….) το 1983 και προέκυψε η ANSI Standard C.

Άσκηση αξιολόγησης (dropdown) - S1_LA3LO21

Η γλώσσα C υποστηρίζει ……..…. προγραμματισμό.

Α. παράλληλο

Β. αντικειμενοστρεφή

Γ. προστακτικό

Δ. λογικό

Άσκηση αξιολόγησης (dropdown) - S1_LA3LO22

Η γλώσσα C μπορεί να χαρακτηριστεί σαν γλώσσα ……

Α. ειδικής χρήσης

Β. τεχνητής νοημοσύνης

Γ. εμπορικών εφαρμογών

Δ. προγραμματισμού συστημάτων

3.2 Ενότητα 2 - Μεταβλητές, Σταθερές, Τύποι δεδομένων

Τίτλος Ενότητας	**S2 -** Μεταβλητές, Σταθερές, Τύποι δεδομένων
Τίτλος Μαθήματος	Βασικές αρχές προγραμματισμού με τη Γλώσσα προγραμματισμού C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστούν τα βασικά στοιχεία της γλώσσας προγραμματισμού C. Θα ασχοληθούμε με τους τύπους δεδομένων που υποστηρίζει, τα είδη των μεταβλητών της, τον τρόπο που υπολογίζονται οι παραστάσεις καθώς και τη δομή που πρέπει να ακολουθεί κάθε πρόγραμμα. Επίσης θα παρουσιαστούν οι βασικές εντολές της γλώσσας προγραμματισμού C, η εντολή εκχώρησης τιμών σε μεταβλητές και οι εντολές εισόδου εξόδου, με τις οποίες το πρόγραμμα επικοινωνεί με το χρήστη.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να αναγνωρίζουν τους τύπους δεδομένων των μεταβλητών • να διακρίνουν τις σταθερές από τις μεταβλητές • να δίνουν τιμή σε μια μεταβλητή είτε με αρχικοποίηση είτε με τη χρήση μιας συνάρτησης εισόδου • να εμφανίζουν την τιμή μιας μεταβλητής με τη χρήση μιας συνάρτησης εξόδου • να διατυπώνουν τη δομή ενός προγράμματος • να συντάσσουν απλά προγράμματα, τα οποία εισάγουν δεδομένα, τα επεξεργάζονται και εμφανίζουν τα αποτελέσματα στην οθόνη
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει οι εκπαιδευόμενοι το κεφάλαιο αυτό θα μπορούν: • να περιγράψουν πώς μια γλώσσα προγραμματισμού υποστηρίζει την πρόσβαση στην κύρια μνήμη του υπολογιστή είτε για αποθήκευση είτε για ανάκληση πληροφορίας, • να αναφέρουν τουλάχιστον 4 βασικούς τύπους δεδομένων που συνήθως υποστηρίζουν οι διάφορες γλώσσες προγραμματισμού, • να δηλώσουν μεταβλητές διαφόρων τύπων και προαιρετικά να δώσουν αρχική τιμή σε αυτές, • να αναγνωρίσουν για το κάθε δεδομένο του προβλήματος αν μπορούν να το κατατάξουν σε μία από τις κατηγορίες δεδομένων που υποστηρίζει η γλώσσα.

Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: • Στην πρώτη, παρουσιάζουμε τους βασικούς τύπους δεδομένων της C και τον τρόπο χειρισμού των μεταβλητών του κάθε τύπου. • Στη δεύτερη, παρουσιάζουμε την έννοια της σταθεράς και της μεταβλητής. • Στην τρίτη, περιγράφουμε τις συναρτήσεις εισόδου και εξόδου της γλώσσας προγραμματισμού και δοκιμάζουμε απλά προγράμματα
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του εκπαιδευτή και των εκπαιδευόμενων.
Αξιολόγηση	Η αξιολόγηση αποτελεί αναπόσπαστο κομμάτι της μάθησης. Γι' αυτό και σε κάθε ενότητα χρησιμοποιείται και ως μέσο μάθησης. Οι ενότητες του μαθήματος αυτού χρησιμοποιούν σε κάθε δραστηριότητα και σε κάθε μαθησιακό αντικείμενο ασκήσεις αυτοαξιολόγησης. Τέτοιες ασκήσεις μπορεί να είναι εργασίες ανάπτυξης προγραμμάτων ή ερωτήσεις αντικειμενικού τύπου. Για τη διεκπεραίωση αυτών των ασκήσεων μπορεί να χρησιμοποιηθούν διάφορα εργαλεία λογισμικού όπως προγράμματα σχεδιασμού ερωτήσεων συμπλήρωσης κενού, σταυρόλεξα κλπ. ή λογισμικό σχεδιασμού ανάπτυξης προγραμμάτων στη γλώσσα C.
Συνολικός χρόνος Ενότητας	***1 εβδομάδα***

3.2.1 Δραστηριότητα 1 - Τύποι δεδομένων

Τίτλος Δραστηριότητας	***S2_LA4*** - Τύποι δεδομένων
Τίτλος Ενότητας	Μεταβλητές, σταθερές, τύποι δεδομένων
Εκπαιδευτική Στρατηγική	*Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.*
Περιγραφή	*Ο σκοπός της ενότητας είναι να παρουσιάσει τους βασικούς τύπους δεδομένων της C, καθώς και τον τρόπο που χρησιμοποιούνται στη δήλωση μεταβλητών. Οι βασικοί τύποι δεδομένων στη γλώσσα* C είναι μεταβλητές ακέραιου τύπου, κινητής υποδιαστολής και τύπου χαρακτήρα. Οι τύποι αυτοί σε συνδυασμό με τροποποιητές μπορούν να χαρακτηριστούν με πρόσημο ή χωρίς, ή σε σχέση με το εύρος τιμών που ορίζουν. Τέλος, οι τύποι δεδομένων της γλώσσας προγραμματισμού C μπορούν να κατηγοριοποιηθούν σαν βαθμωτοί η συναθροιστικοί όπως επίσης και σαν ενσωματωμένοι ή παραγόμενοι.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τις βασικές εντολές ορισμού τύπου δεδομένων της γλώσσας προγραμματισμού C • να εξηγήσουν τι είναι οι παραγόμενοι τύποι και γιατί είναι σημαντικό να υποστηρίζονται από μια γλώσσα προγραμματισμού • να εξηγήσουν τη διαφορά των τύπων float και double για τους πραγματικούς αριθμούς • να αναφέρουν τουλάχιστον 2 συναθροιστικούς τύπους δεδομένων • να κατασκευάσουν ένα παράδειγμα ορισμού και χρήσης φυσικών αριθμών σε ένα πρόγραμμα
Μαθησιακά Αντικείμενα	Τύποι δεδομένων - θεωρία Τύποι δεδομένων - ασκήσεις
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 5 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στο μαθησιακό αντικείμενο S2_LA4LO20.
Λέξεις Κλειδιά	Τύπος ακεραίου αριθμού, πραγματικού αριθμού, πραγματικού αριθμού διπλής ακρίβειας, χαρακτήρα

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA4LO20**
Τίτλος	**Τύποι δεδομένων της C**
Τίτλος Εκπαιδευτικής Δραστηριότητας	S2_LA4 - Τύποι δεδομένων
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι μεταβλητές ορισμένου τύπου δεδομένων και δίνονται παραδείγματα δήλωσης ορισμένων τύπων μεταβλητών με τροποποιητές ή όχι. Ένας τύπος δεδομένων είναι ένα σύνολο τιμών και ένα σύνολο λειτουργιών (πράξεων) που μπορούν να εφαρμοστούν σε αυτές τις τιμές. Χωρίζονται σε βασικούς τύπους και σε σύνθετους ή συναθροιστικούς.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία και ασκήσεις αυτοαξιολόγησης
Τεχνικός Τύπος	Υπερκείμενο
Λέξεις Κλειδιά	Τύπος ακεραίου αριθμού, πραγματικού αριθμού, πραγματικού αριθμού διπλής ακρίβειας, χαρακτήρα
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τις βασικές εντολές ορισμού τύπου δεδομένων της γλώσσας προγραμματισμού C • να εξηγήσουν τη διαφορά των τύπων float και double για τους πραγματικούς αριθμούς • να κατασκευάσουν ένα παράδειγμα ορισμού και χρήσης φυσικών αριθμών σε ένα πρόγραμμα

Η C προσφέρει ένα μικρό αλλά χρήσιμο σύνολο τύπων δεδομένων. Στην κατηγορία των **βαθμωτών** ή βασικών τύπων, δηλαδή αυτών που δεν μπορούν να διασπαστούν σε απλούστερα στοιχεία, έχει τους αριθμούς, ακέραιους (int) και πραγματικούς (float και double), το χαρακτήρα (char), τους δείκτες (pointers) και τον απαριθμητικό (enum) τύπο. Στην κατηγορία των **συναθροιστικών** έχει τους πίνακες, τις δομές (struct) και τις ενώσεις (union).

Μια άλλη κατηγοριοποίηση για τους τύπους δεδομένων στη γλώσσα C είναι οι ενσωματωμένοι τύποι δεδομένων, όπως οι παραπάνω βασικοί τύποι, και οι παραγόμενοι τύποι, δηλαδή νέοι τύποι δεδομένων που έχει τη δυνατότητα να ορίζει ο χρήστης.

Η C υποστηρίζει βασικούς τύπους δεδομένων, οι οποίοι παρουσιάζονται στον παρακάτω πίνακα μαζί με τις δεσμευμένες λέξεις που τους αντιπροσωπεύουν.

Τύπος	**Δεσμευμένη λέξη**
χαρακτήρας	char
προσημασμένος ακέραιος	int
αριθμός κινητής υποδιαστολής	float
αριθμός κινητής υποδιαστολής διπλής ακρίβειας	double

Μια **μεταβλητή τύπου char** έχει μέγεθος 8 bits και χρησιμοποιείται συνήθως για την αποθήκευση ενός μεμονωμένου χαρακτήρα. Η τιμή ενός χαρακτήρα εσωκλείεται με αποστρόφους (single quotes)

'C', '2', '*', ','

Παραδείγματα δήλωσης χαρακτήρων:

char choice= 'A';

char x, y;

Μέγεθος: 1 byte (εύρος τιμών -128 έως 127)

Σε μερικές υλοποιήσεις ο τύπος char σημαίνει signed char, ενώ σε άλλες σημαίνει unsigned char. Στη C κάθε χαρακτήρας αναπαρίσταται ως ακέραιος με τιμή τον αντίστοιχο κωδικό ASCII. Έτσι η C είναι πολύ ευέλικτη και μια μεταβλητή τύπου char μπορεί επίσης να χρησιμοποιηθεί σαν ένας ‘μικρός ακέραιος’ (short int).

Μερικά παραδείγματα ASCII κωδικών είναι:

'0': ASCII 48, 'A': ASCII 65, 'Z': ASCII 90, 'a': ASCII 97, 'z': ASCII 122, '*': ASCII 42, '=': ASCII 61, '~': ASCII 126 κλπ.

Στην παρακάτω εικόνα 4 βλέπουμε τον πίνακα ASCII κωδικών

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Εικόνα 4. American Standard Code for Information Interchange - ASCII

Ειδικές τιμές χαρακτήρων με βάση τον πίνακα είναι:

'\0' ASCII 0 NUL

'\a' ASCII 7 BEL

'\b' ASCII 8 BS (backspace)

'\f' ASCII 12 FF (form feed)

'\n' ASCII 10 LF (newline)

'\r' ASCII 13 CR (carriage return)

'\t' ASCII 9 HT (horizontal tab)

'\v' ASCII 11 VT (vertical tab)

'\\' ASCII 92 backslash

'\" ASCII 39 single quote

Οι μεταβλητές για την αποθήκευση ακέραιων τιμών (int) μπορούν να αποθηκεύουν προσημασμένους ακέραιους αριθμούς (αριθμούς χωρίς δεκαδικό ή κλασματικό μέρος). Για περιβάλλοντα τα οποία λειτουργούν στα 16-bits όπως το DOS ή τα Windows 3.1, οι ακέραιοι έχουν συνήθως μέγεθος 16 bits και μπορούν να αποθηκεύουν τιμές στη περιοχή τιμών -32768 έως 32767. Σε περιβάλλοντα τα οποία λειτουργούν στα 32-bits όπως τα Windows NT, το τυπικό μέγεθος των ακεραίων είναι συνήθως 32 bits. Σε αυτή την περίπτωση οι ακέραιες μεταβλητές μπορούν να αποθηκεύουν τιμές στη περιοχή τιμών -2.147.483.648 έως 2.147.483.647.

Ο ακέραιος τύπος δεδομένων μπορεί να συνοδεύεται από προσδιοριστικό μήκους, όπως: short ή long. Ο short int έχει μήκος 16 bit, ο int 32 bit και ο long int 64 bits.

Απρόσημος ακέραιος τύπος

Αν θέλουμε να αποθηκεύει μόνο μη αρνητικούς ακεραίους, τότε βάζουμε μπροστά στη δήλωση τον προσδιοριστή unsigned. Υποστηρίζονται τρεις διαφορετικοί τύποι:

- unsigned short
- unsigned int
- unsigned long

Αν και αποθηκεύουν το ίδιο πλήθος ακεραίων αριθμών (καθώς χρησιμοποιούν το ίδιο πλήθος bits), οι απρόσημοι ακέραιοι μπορούν να αποθηκεύσουν διπλάσιο πλήθος θετικών αριθμών από τους προσημασμένους.

Έτσι, ενώ ο int αποθηκεύει τους θετικούς ακεραίους 0 έως 2.147.483.647, ο unsigned int αποθηκεύει τους θετικούς ακεραίους 0 έως 4.294.967.295

Παρατηρήσεις

- Οι απρόσημοι ακέραιοι διευκολύνουν τις αριθμητικές λειτουργίες σε bits.
- Η μίξη προσημασμένων και απρόσημων ακεραίων στην ίδια έκφραση μπορεί να περιπλέξει τις μετατροπές τύπων.
- Η γενική οδηγία σε μεικτές εκφράσεις είναι να μετατρέπονται οι προσημασμένοι σε απρόσημους.

Αναπαράσταση ακέραιων σταθερών

Οι άνθρωποι χρησιμοποιούν αριθμούς σε δεκαδική μορφή, αλλά οι προγραμματιστές μπορεί να χρησιμοποιούν αριθμούς σε δυαδική, οκταδική ή δεκαεξαδική μορφή. Η αναπαράσταση των ακέραιων σταθερών περιλαμβάνει:

- Δεκαδική μορφή ως default αναπαράσταση
- Οκταδική μορφή με πρόθεμα 0 (μηδέν)
- Δεκαεξαδική μορφή με πρόθεμα 0x
- Δεν υποστηρίζεται δυαδική αναπαράσταση
- Η κατάληξη L υποδηλώνει σταθερές long
- Για τις σταθερές short δεν χρησιμοποιείται κατάληξη, αλλά πρέπει να γίνει μετατροπή τύπου (cast).

Παραδείγματα:

int: 2010, -2010, 03732, 0x7DA

long: 2010L, -2010L, 03732L, 0x7DAL

short: (short)2010, (short)-2010, (short)03732, (short)0x7DA

Οι μεταβλητές με τύπο float και double αποθηκεύουν προσημασμένες τιμές κινητής υποδιαστολής οι οποίες μπορούν να έχουν δεκαδικό μέρος. Μια διαφορά μεταξύ των τύπων δεδομένων float και double είναι το γεγονός ότι ο τύπος double προβλέπει περίπου διπλάσια ακρίβεια (πλήθος δεκαδικών ψηφίων) από τον τύπο float. Επίσης για τις περισσότερες χρήσεις της C μια μεταβλητή τύπου double έχει δυνατότητα να αποθηκεύει τιμές με απόλυτο πεδίο τιμών μεγαλύτερο από αυτό που μπορούν να αποθηκεύουν οι μεταβλητές τύπου float. Φυσικά σε όλες τις περιπτώσεις οι μεταβλητές τύπου float και double μπορούν να αποθηκεύουν πολύ μεγάλες τιμές. Ειδικότερα:

Τύπος float

Αναπαριστά πραγματικούς αριθμούς, θετικούς ή αρνητικούς (κινητής υποδιαστολής απλής ακρίβειας). Έχουν μέγεθος συνήθως 4 bytes ανάλογα με τον υπολογιστή.

Παράδειγμα: float x;

Αναπαράσταση σταθερής υποδιαστολής (ακρίβεια συγκεκριμένων δεκαδικών ψηφίων), π.χ. 0.012

Αναπαράσταση κινητής υποδιαστολής (αυξημένη ακρίβεια δεκαδικών ψηφιών με επιστημονικό συμβολισμό) π.χ. 6.3E-05

Τύπος double

Αναπαριστά πραγματικούς αριθμούς, θετικούς ή αρνητικούς (κινητής υποδιαστολής διπλής ακρίβειας). Έχουν διπλάσια ακρίβεια δεκαδικών ψηφίων σε σχέση με τύπους float. Έχουν μέγεθος συνήθως 8 bytes ανάλογα με τον υπολογιστή.

Παράδειγμα: double x;

Τύπος long double

Αναπαριστά πραγματικούς αριθμούς θετικούς ή αρνητικούς (κινητής υποδιαστολής εκτεταμένης ακρίβειας). Έχουν μέγεθος συνήθως 12 bytes ανάλογα με τον υπολογιστή. Έχουν διπλάσια ακρίβεια δεκαδικών ψηφίων σε σχέση με τύπους double.

Παράδειγμα: long double x;

Άσκηση αυτοαξιολόγησης - S2_LA4LO21

Αντιστοιχίστε τους παρακάτω τύπους δεδομένων με την κατηγορία στην οποία ανήκουν:

1. ακέραιος (int)	α. βαθμωτός τύπος δεδομένων
2. πραγματικός (float)	β. συναθροιστικός τύπος δεδομένων
3. πραγματικός (double)	
4. χαρακτήρας (char)	
5. απαριθμητικός (enum)	
6. πίνακας	
7. δομή (struct)	
8. ένωση (union)	

Άσκηση αυτοαξιολόγησης - S2_LA4LO22

Συμπληρώστε την τρίτη στήλη του πίνακα:

Α/Α	**Δεδομένο**	**Τύπος δεδομένου**
1	17	
2	“George”	
3	2.35	
4	0.0023	
5	-25	
6	‘m’	
7	4.32E-6	
8	“185.3”	
9	0	
10	1	

Παρατήρηση εφαρμογής:

Τα δεδομένα με α/α 9 και 10 παρουσιάζουν την εξής δυσκολία. Η C ουσιαστικά δεν υποστηρίζει λογικές μεταβλητές. Θεωρεί το 0 ως FALSE και κάθε μη μηδενική τιμή ως TRUE. Με αυτό το δεδομένο, εναπόκειται στον προγραμματιστή να χειριστεί ακέραιους αριθμούς για να αναπαραστήσει λογικές τιμές.

Στην ANSI C99, ένα λογικού τύπου δεδομένο υποστηρίζεται και καθορίζεται στο αρχείο βιβλιοθήκης stdbool.h. Στο stdbool.h η χρήση του τύπου TRUE και FALSE, υποκαθίστανται από τις ακέραιες σταθερές 0 και 1:

#define false 0

#define true 1

Άσκηση αυτοαξιολόγησης - S2_LA4LO23

Ανατρέξτε στο αρχείο limits.h που συνοδεύει τον μεταγλωττιστή σας για να δείτε τα όρια των διαφόρων τύπων ακεραίων: short, long, signed, unsigned. Δημιουργήστε ένα πρόγραμμα στο οποίο θα δηλώσετε τις μεταβλητές των παραπάνω τύπων, θα ζητήσετε αντίστοιχες τιμές από τον χρήστη και στη συνέχεια θα τυπώσετε τις μεταβλητές αυτές αυξημένες κατά ένα. Τι παρατηρείτε;

Άσκηση αυτοαξιολόγησης - S2_LA4LO24

Αντιστοιχίστε κάθε έναν από τους παρακάτω τύπους με την κατηγορία ή τις κατηγορίες στις οποίες ανήκει:

1. χαρακτήρας	α. ενσωματωμένος (για την C)
2. ακέραιος	β. παραγόμενος (για την C)
3. πραγματικός	
4. πίνακας	
5. ημερομηνία	

Άσκηση αυτοαξιολόγησης - S2_LA4LO225

Δηλώστε τις παρακάτω μεταβλητές:

Μια μεταβλητή τύπου ακεραίου με όνομα PORTA η οποία μπορεί να κρατάει μόνο θετικούς αριθμούς

Μια μεταβλητή με όνομα **up_down** η οποία μπορεί να παίρνει τιμές από το -20000 έως το 20000

Μια μεταβλητή με όνομα **heat** η οποία μπορεί να κρατάει όλους τους αριθμούς από το -20,000 έως το 350,000

Μια μεταβλητή με όνομα **max_read** η οποία χρειάζεται ακρίβεια 8 δεκαδικών ψηφίων

Για κάθε μεταβλητή που δηλώνετε κάντε ένα σχόλιο για το πόσα bytes μνήμης χρειάζονται και το εύρος δεδομένων που καλύπτουν

3.2.2 Δραστηριότητα 2 - Μεταβλητές Σταθερές

Τίτλος Δραστηριότητας	**S2_LA5** - Μεταβλητές, Σταθερές
Τίτλος Ενότητας	Μεταβλητές, Σταθερές, τύποι δεδομένων
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης
Περιγραφή	Η ενότητα περιγράφει τις πολύ βασικές έννοιες της μεταβλητής και της σταθεράς καθώς και τον τρόπο με τον οποίο αυτές χρησιμοποιούνται στη διαδικασία συγγραφής προγράμματος. Οι μεταβλητές περιέχουν μια τιμή, η οποία μπορεί να μεταβληθεί κατά τη διάρκεια εκτέλεσης του προγράμματος. Για να χρησιμοποιηθεί όμως μια μεταβλητή πρέπει πρώτα να δηλωθεί, να της δοθεί δηλαδή ένα όνομα και να επιλεγεί ο τύπος δεδομένων που θα αποθηκεύει. Οι σταθερές δηλώνονται είτε όπως οι μεταβλητές, με προσθήκη της δεσμευμένης λέξης const στην αρχή, είτε με την οδηγία define. Και οι δύο τρόποι δεν επιτρέπουν να αλλάξει η τιμή μετά την αρχικοποίηση.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη διαφορά μεταξύ μιας μεταβλητής και μιας σταθεράς • να αναφέρουν τις βασικές εντολές ορισμού τύπου δεδομένων της γλώσσας προγραμματισμού C • να κατασκευάσουν ένα παράδειγμα δήλωσης μιας μεταβλητής τύπου μη προσημασμένου ακεραίου αριθμού
Μαθησιακά Αντικείμενα	Δήλωση μεταβλητών - Αρχικοποίηση μεταβλητών Ορισμός σταθερών
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 4 και 2 μαθησιακά αντικείμενα με ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S2_LA5LO26 και S2_LA5LO33 αντίστοιχα.
Λέξεις Κλειδιά	Μεταβλητή, τύπος δεδομένων, σταθερά

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA5LO26**
Τίτλος	**Δήλωση μεταβλητών- Αρχικοποίηση μεταβλητών**
Τίτλος δραστηριότητας	Μεταβλητές, Σταθερές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται ο τρόπος δήλωσης μιας μεταβλητής ορισμένου τύπου δεδομένων, καθώς επίσης δίνονται παραδείγματα δήλωσης μεταβλητών των βασικών τύπων δεδομένων. *Η έννοια της μεταβλητής που περιγράφεται εδώ είναι από τις βασικές έννοιες για την εισαγωγή στον προγραμματισμό και τη γλώσσα C.*
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία - Ασκήσεις αυτοαξιολόγησης
Τεχνικός Τύπος	Κείμενο
Λέξεις Κλειδιά	Μεταβλητή, τύπος δεδομένων
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τις βασικές εντολές ορισμού τύπου δεδομένων της γλώσσας προγραμματισμού C • να δημιουργήσουν τουλάχιστον δύο παραδείγματα δήλωσης μεταβλητών • να κατασκευάσουν ένα παράδειγμα δήλωσης μιας μεταβλητής τύπου μη προσημασμένου ακεραίου αριθμού

Μια μεταβλητή είναι μια επώνυμη θέση στη μνήμη η οποία μπορεί να λαμβάνει διάφορες τιμές. Μόνο τα πολύ απλά προγράμματα στη C δεν περιλαμβάνουν μεταβλητές. Στη C όλες οι μεταβλητές πρέπει να δηλώνονται πριν μπορέσουν να χρησιμοποιηθούν. Η δήλωση μιας μεταβλητής εξυπηρετεί ένα σημαντικό σκοπό: λέει στον μεταγλωττιστή τον τύπο δεδομένων που χρησιμοποιεί η μεταβλητή ώστε να δεσμεύσει τον ανάλογο χώρο στη μνήμη. Μπορείτε να δηλώσετε μια μεταβλητή με την ακόλουθη γενική μορφή:

τύπος όνομα-μεταβλητής;

όπου τύπος είναι ένας τύπος δεδομένων της C και όνομα-μεταβλητής είναι το όνομα που δίνεται στη μεταβλητή. Για παράδειγμα η ακόλουθη γραμμή δηλώνει την μεταβλητή counter σαν τύπου int (ακέραιος):

int counter;

Στη C, μια δήλωση μεταβλητής είναι μια εντολή που πρέπει να τερματίζει με τον χαρακτήρα του ελληνικού ερωτηματικού.

Μπορείτε να δηλώσετε παραπάνω από δύο μεταβλητές του ίδιου τύπου χρησιμοποιώντας μια λίστα διαχωρισμένη με κόμματα. Δεν μπορείτε να δηλώσετε δύο μεταβλητές με το ίδιο όνομα στο ίδιο τμήμα του προγράμματος.

Όμοια με τα ονόματα των συναρτήσεων στη C, τα ονόματα των μεταβλητών μπορούν να αποτελούνται από τα γράμματα της αγγλικής αλφαβήτου, τα ψηφία 0 έως 9 και το χαρακτήρα της κάτω παύλας. Ωστόσο δεν μπορείτε να ξεκινάτε το όνομα μιας μεταβλητής με ένα αριθμητικό ψηφίο. Τέλος η C κάνει διάκριση μεταξύ κεφαλαίων και πεζών χαρακτήρων, π.χ. τα count και COUNT είναι διαφορετικά ονόματα μεταβλητών. Τα ονόματα των μεταβλητών δεν μπορεί να είναι δεσμευμένες λέξεις της γλώσσας, π.χ. απαγορεύεται μια μεταβλητή να έχει όνομα float.

Για να εκχωρήσετε μια τιμή σε μια μεταβλητή τοποθετείτε το όνομα της μεταβλητής στα αριστερά του συμβόλου ίσον και την τιμή που θέλετε να δώσετε στην μεταβλητή στα δεξιά του συμβόλου ίσον. Στη C, μια λειτουργία εκχώρησης τιμής είναι μια εντολή, συνεπώς πρέπει να τερματίζει με το χαρακτήρα του ελληνικού ερωτηματικού. Η γενική μορφή μιας εντολής εκχώρησης τιμής είναι:

όνομα-μεταβλητής = τιμή;

Μια μεταβλητή, όποιος κι αν είναι ο τύπος της, χαρακτηρίζεται από ένα όνομα, μια διεύθυνση και μία τιμή, όπως φαίνεται στην παρακάτω εικόνα. Το όνομα μίας μεταβλητής είναι άμεσα συνδεδεμένο με τη διεύθυνση στην οποία είναι αποθηκευμένο το δεδομένο. Με τον τρόπο αυτό ο προγραμματιστής μπορεί να χειριστεί δεδομένα χωρίς να γνωρίζει την ακριβή διεύθυνση της μνήμης όπου αυτά τοποθετούνται. Ο χώρος που καταλαμβάνει η μεταβλητή στη μνήμη εξαρτάται από τον τύπο της μεταβλητής. Έτσι, η μεταβλητή ch που είναι τύπου char καταλαμβάνει ένα byte, ενώ η μεταβλητή size που είναι τύπου int καταλαμβάνει τέσσερα bytes.

Εικόνα 5. Χαρακτηριστικά μεταβλητών

Άσκηση πολλαπλής επιλογής - S2_LA5LO27

1) Δεν υπάρχει πρόγραμμα στη C που να μην εμπεριέχει τουλάχιστον μια μεταβλητή.

1. Σωστό 2. Λάθος

2) Τι δηλώνει η ακόλουθη γραμμή:

int counter;

1. μεταβλητή με όνομα counter τύπου ακεραίου
2. μεταβλητή με όνομα int τύπου ακεραίου
3. μεταβλητή με όνομα counter τύπου χαρακτήρα
4. μεταβλητή με όνομα counter τύπου πραγματικού

3) Στη C, όλες οι μεταβλητές πρέπει να δηλώνονται πριν μπορέσουν να χρησιμοποιηθούν.

1. Σωστό 2. Λάθος

4) Δεν είναι απαραίτητη στη δήλωση μεταβλητών η χρήση του ελληνικού ερωτηματικού.

1. Σωστό 2. Λάθος

5) Η γενική μορφή μιας εντολής εκχώρησης τιμής είναι:

1. όνομα-μεταβλητής = τιμή;
2. τιμή = όνομα-μεταβλητής;
3. όνομα-μεταβλητής = τύπος δεδομένου;
3. τιμή = τύπος δεδομένου;

6) Μπορούμε να δηλώσουμε δύο μεταβλητές που έχουν το ίδιο όνομα, αλλά το πρώτο γράμμα στη μία είναι κεφαλαίο και στην άλλη πεζό.

1. Σωστό 2. Λάθος

7) Ποιο από τα παρακάτω ονόματα μεταβλητών δεν είναι αποδεκτό στη γλώσσα C.

1. counter 2. counter1 3. counter1 4. counter_1 5. 1_counter

8) Μπορείτε να δηλώσετε μια μεταβλητή χωρίς απαραίτητα να της εκχωρήσετε τιμή.

1. Σωστό 2. Λάθος

9) Μια μεταβλητή είναι μια επώνυμη θέση στη μνήμη η οποία μπορεί να λαμβάνει διάφορες τιμές.

1. Σωστό 2. Λάθος

10) Ποιο από τα παρακάτω ονόματα μεταβλητών δεν είναι έγκυρο

1. totalArea 2.max_amount
3. Counter1 _temp_in_F 4. total%

Άσκηση αυτοαξιολόγησης - S2_LA5LO28

Αν θέλετε να εκχωρήσετε στην ακέραια μεταβλητή με όνομα num την τιμή 100 θα πρέπει να χρησιμοποιήσετε την ακόλουθη εντολή:

1. num = 100;

2. num == 100;

3. int num = 100;

4. num_int = 100;

Άσκηση αυτοαξιολόγησης- S2_LA5LO29

Να αντιστοιχίσετε τα κελιά της πρώτης στήλης του πίνακα με τον σωστό τύπο δεδομένων που βρίσκεται στα κελιά της δεύτερης στήλης του πίνακα

Μεταβλητή	**Τύπος**
Μετρητής	float a
βαθμολογία φοιτητή	int s
βάρος, μάζα ή ταχύτητα	unsigned char
λογική κατάσταση	char g[] ή char *g
όνομα ή διεύθυνση	float d, ή double d

Παρατήρηση:

Δηλώνοντας το σωστό τύπο για τις παραπάνω μεταβλητές

- Επιτυγχάνουμε καλύτερη εκμετάλλευση της μνήμης

- Επιτυγχάνουμε καλύτερο έλεγχο κατά τη μεταγλώττιση του προγράμματος

Άσκηση αυτοαξιολόγησης - S2_LA5LO30

Να χαρακτηρίσετε τα παρακάτω ονόματα μεταβλητών, εάν κατά την γνώμη σας είναι ορισμένα σωστά ή όχι.

ΟΝΟΜΑΤΟΛΟΓΙΑ ΜΕΤΑΒΛΗΤΩΝ	
1. int main;	α. Σωστός ορισμός μεταβλητής
2. int Main;	β. Λάθος ορισμός μεταβλητής
3. int 3x;	
4. int x123456;	
5. int hello world;	
6. int hello_world;	

Ερωτήσεις αντικειμενικού τύπου - S2_LA5LO32

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Το όνομα μιας μεταβλητής είναι άμεσα συνδεμένο με την ακριβή διεύθυνση της μνήμης όπου είναι αποθηκευμένη η τιμή της μεταβλητής.
2. Η C απαιτεί από τον προγραμματιστή να προσδιορίζει τις ιδιότητες κάθε μεταβλητής στη διάρκεια της μεταγλώττισης.
3. Η C δεν απαιτεί να δηλώνεται μια μεταβλητή πριν από τη χρήση της.
4. Η δήλωση μεταβλητών στη γλώσσα C απαιτεί ταυτόχρονα την απόδοση τιμής.
5. Η δυνατότητα ορισμού νέων τύπων δεδομένων σε μια γλώσσα προγραμματισμού αυξάνει όχι μόνο την αναγνωσιμότητα του προγράμματος αλλά και την αξιοπιστία.
6. Οι τύποι δεδομένων διακρίνονται σε βαθμωτούς (scalar) και συναθροιστικούς (aggregate).
7. Ο προσδιοριστής **unsigned** χρησιμοποιείται στη δήλωση πριν από τη λέξη **int** για να χαρακτηρίσει τη μεταβλητή ως απρόσημη.
8. Για έναν υπολογιστή με λέξη 16 bit, η περιοχή τιμών του τύπου int είναι από -32767 έως +32768 για προσημασμένους ακεραίους
9. Ο προσδιοριστής **long** χρησιμοποιείται πριν από τη λέξη int για να προσδιορίσει ακέραιο με μεγαλύτερο εύρος τιμών.
10. Τα ονόματα των μεταβλητών μπορεί να είναι δεσμευμένες λέξεις της γλώσσας C.

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA5LO33**
Τίτλος	**Ορισμός σταθερών**
Τίτλος δραστηριότητας	Μεταβλητές, Σταθερές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται ο τρόπος ορισμού μιας σταθεράς στη γλώσσα προγραμματισμού C. Οι σταθερές δηλώνονται είτε όπως οι μεταβλητές, με προσθήκη της δεσμευμένης λέξης const στην αρχή, είτε με την οδηγία define. Και οι δύο τρόποι δεν επιτρέπουν να αλλάξει η τιμή μετά την αρχικοποίηση.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Σταθερά
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη διαφορά μεταξύ μιας μεταβλητής και μιας σταθεράς • να εξηγήσουν τη χρήση των σταθερών σε ένα πρόγραμμα C

Αν θέλετε να εκχωρήσετε στην ακέραια μεταβλητή με όνομα num την τιμή 100 θα πρέπει να χρησιμοποιήσετε την ακόλουθη εντολή:

num =100;

Στην παραπάνω εντολή εκχώρησης, η τιμή 100 είναι μια σταθερά. Ακριβώς όπως υπάρχουν διάφοροι τύποι μεταβλητών υπάρχουν και διάφοροι τύποι σταθερών. Μια σταθερά είναι μια σταθερή τιμή την οποία χρησιμοποιείτε στο πρόγραμμά σας.

Δήλωση σταθεράς

Μια δηλωμένη σταθερά είναι ένα όνομα που αντιπροσωπεύει μια τιμή η οποία δεν αλλάζει κατά την εκτέλεση του προγράμματος. Τα χαρακτηριστικά μιας σταθεράς είναι το όνομα και η τιμή της. Σε αντίθεση με τη μεταβλητή, η σταθερά δεν έχει διεύθυνση στη μνήμη (αναφορά). Ο μεταγλωττιστής αντικαθιστά το όνομα με την τιμή. Η χρήση των σταθερών συνεισφέρει:

- Στην αναγνωσιμότητα των προγραμμάτων

- Στην ευκολία τροποποίησης των προγραμμάτων

Η δήλωση σταθερών μπορεί να γίνει με 2 τρόπους:

1. Με τη δήλωση const

const <τύπος δεδομένων> <όνομα>= <τιμή>;

Ένας τρόπος είναι να ορίσουμε ένα όνομα και να το εξισώσουμε με την επιθυμητή σταθερά, ως εξής:

const float taxrate = 0.2;

Μ' αυτόν τον τρόπο η αντικατάσταση της τιμής του ονόματος taxrate θα γίνεται όταν το πρόγραμμα τρέχει.

Παραδείγματα:

const double PI = 3.14159;

const unsigned char TRUE = 1;

2. Με τη δήλωση #define

#define <όνομα> <τιμή>

Ο προεπεξεργαστής της C διαθέτει έναν καλύτερο τρόπο, όπου απλά προσθέτουμε μια γραμμή στην αρχή του αρχείου που περιέχει το πρόγραμμά μας, ως εξής:

#define TAXRATE 0.2

Όταν το πρόγραμμα μεταγλωττιστεί, η τιμή 0.2 θα αντικατασταθεί παντού όπου έχει χρησιμοποιηθεί το όνομα TAXRATE. Έτσι, όταν τρέξουμε το πρόγραμμα, όλες οι αντικαταστάσεις θα έχουν ήδη γίνει. Πρέπει να προσέξουμε τη σύνταξη της #define, στην οποία δεν χρησιμοποιείται το ελληνικό ερωτηματικό (;) στο τέλος, αφού δεν είναι μια πρόταση της C και ακόμη αποτελεί παράδοση (αλλά όχι κανόνα) στη C να γράφονται όλες οι σταθερές με κεφαλαία γράμματα για να τις αναγνωρίζουμε αμέσως και να τις ξεχωρίζουμε έτσι από τις μεταβλητές.

Παραδείγματα:

#define PI 3.14159

#define TRUE 1

Ερωτήσεις αντικειμενικού τύπου - S2_LA5LO34

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Με την εντολή προεπεξεργαστή define N 5, η τιμή 5 θα αντικατασταθεί παντού όπου έχει χρησιμοποιηθεί με το όνομα της σταθεράς N.
2. Με τη λέξη κλειδί const, μία σταθερά μπορεί να αλλάξει τιμή κατά τη διάρκεια της εκτέλεσης του προγράμματος, μετά την ανάθεση της αρχικής της τιμής.
3. Μια σταθερά τύπου χαρακτήρα προσδιορίζεται περικλείοντας το χαρακτήρα μέσα σε αποστρόφους.
4. Οι σταθερές χρησιμοποιούνται συχνά για την αρχικοποίηση των μεταβλητών με κάποια τιμή, στην αρχή της εκτέλεσης ενός προγράμματος.

5. Η πρόταση float plank = 6.63e-34; δηλώνει τη σταθερά plank σαν απλής ακρίβειας και της δίνει αρχική τιμή 6.63e-34.
6. Η χρήση των σταθερών συνεισφέρει στην αναγνωσιμότητα των προγραμμάτων.
7. Μια σταθερά μπορεί να αλλάξει τιμή κατά τη διάρκεια εκτέλεσης ενός προγράμματος.
8. Μια σταθερά μπορεί να αλλάζει τύπο δεδομένων κατά τη διάρκεια εκτέλεσης ενός προγράμματος.
9. Μια σταθερά μπορεί να αποθηκεύσει και αλφαριθμητικά δεδομένα.
10. Μια μεταβλητή μπορεί να αλλάζει τιμή και όνομα κατά τη διάρκεια εκτέλεσης ενός προγράμματος.
11. Σε μια εντολή εκχώρησης δεν επιτρέπεται η χρήση σταθερών.
12. Για την αναπαράσταση των δεδομένων εισόδου ενός προγράμματος χρησιμοποιούμε τις σταθερές.
13. Το αποτέλεσμα μια πράξης μπορεί να εκχωρηθεί σε μια σταθερά

Άσκηση αυτοαξιολόγησης - S2_LA5LO35

Να βάλετε στη σωστή σειρά τις παρακάτω εντολές και σχόλια προγράμματος:

```
// Δήλωση σταθερής τιμής με define
const float fpa_percent = 0.21;
#include<stdio.h>
int main()
{
#define FPA_PERCENT 0.21
// Δήλωση σταθερής τιμής με const
}
```

3.2.3 Δραστηριότητα 3 - Εντολές εισόδου/εξόδου

Τίτλος Δραστηριότητας	**S2_LA6** - Εντολές εισόδου/εξόδου
Τίτλος Ενότητας	***Μεταβλητές, Σταθερές, τύποι δεδομένων***
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αντιστοίχισης, ασκήσεις συμπλήρωσης κενών (μέθοδος σύρε και άφησε) και εργασίες.
Περιγραφή	Ο σκοπός της ενότητας είναι να παρουσιάσει τις συναρτήσεις εισόδου και εξόδου που χρησιμοποιούνται στη γλώσσα προγραμματισμού C. Αρχικά περιγράφεται η συνάρτηση printf, πώς συντάσσεται και ποιοι είναι οι προσδιοριστές που χρησιμοποιούνται για να εμφανιστούν οι μεταβλητές του προγράμματος με διαφορετική μορφή. Στη συνέχεια περιγράφεται η συνάρτηση scanf με το γενικό τύπο της και τα ορίσματά της και επίσης πώς χρησιμοποιούνται οι προσδιοριστές σ' αυτή τη συνάρτηση. Τέλος δίνονται οι βασικές οδηγίες για το σχηματισμό ενός προγράμματος στη γλώσσα C, περιγράφεται η δομή του και δίνονται κάποιες παρατηρήσεις.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη χρήση των εντολών εισόδου/εξόδου της γλώσσας προγραμματισμού C • να αναφέρουν τους βασικούς προσδιοριστές που χρησιμοποιούνται στις συναρτήσεις εισόδου/εξόδου της γλώσσας προγραμματισμού C • να δώσουν δύο τουλάχιστον παραδείγματα συναρτήσεων εισόδου/εξόδου με διαφορετικούς προσδιοριστές.
Μαθησιακά Αντικείμενα	Εντολές εξόδου - συνάρτηση printf Εντολές εισόδου - συνάρτηση scanf Το 1ο πρόγραμμα
Αξιολόγηση Εκπαιδευομένων	Σε κάθε μαθησιακό αντικείμενο θεωρίας υπάρχουν ασκήσεις αυτοαξιολόγησης για την εμπέδωση αυτής της θεωρίας, ενώ όπου χρειάζεται απαιτείται εργασία συνολικής αξιολόγησης για το θεωρητικό υπόβαθρο της δραστηριότητας ή και της ενότητας. Έτσι τα μαθησιακά αντικείμενα S2_LA6LO36 και S2_LA6LO43 που παρουσιάζουν τη θεωρία της δραστηριότητας υποστηρίζονται από 6 και 2 ασκήσεις αυτοαξιολόγησης αντίστοιχα. Το μαθησιακό αντικείμενο S2_LA6LO46 αποτελείται από 5 ασκήσεις αξιολόγησης όλης της ενότητας.
Λέξεις Κλειδιά	Έξοδος, εκτύπωση, συνάρτηση printf, προσδιοριστές, συνάρτηση scanf, είσοδος

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA6LO36**
Τίτλος	**Εντολές εξόδου - συνάρτηση printf**
Τίτλος δραστηριότητας	Εντολές εισόδου/εξόδου
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται η συνάρτηση εξόδου printf και πώς αυτή χρησιμοποιείται για την εκτύπωση μεταβλητών διαφόρων τύπων δεδομένων. Δίνεται ο γενικός τύπος της συνάρτησης και περιγράφονται αναλυτικά τα ορίσματά της. Τέλος δίνονται παραδείγματα εφαρμογής της συνάρτησης με χρήση διαφορετικών προσδιοριστών για μεταβλητές διαφορετικού τύπου.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία, Ασκήσεις αυτοαξιολόγησης
Τεχνικός Τύπος	Κείμενο
Λέξεις Κλειδιά	Έξοδος, εκτύπωση, συνάρτηση printf, προσδιοριστές
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τρία τουλάχιστον παραδείγματα εξόδου της γλώσσας προγραμματισμού C • να εξηγήσουν τη χρήση των εντολών εξόδου της γλώσσας προγραμματισμού C • να αναφέρουν τους βασικούς προσδιοριστές που χρησιμοποιούνται στη συνάρτηση εξόδου της γλώσσας προγραμματισμού C • να εξηγήσουν τη χρήση των προσδιοριστών • να κατασκευάσουν ένα παράδειγμα εξόδου ενός ακεραίου στην οθόνη του Η/Υ

Μπορείτε να χρησιμοποιήσετε την printf για να εμφανίζετε τιμές οι οποίες είναι χαρακτήρες, ακέραιοι και αριθμοί κινητής υποδιαστολής. Ωστόσο, για να το κάνετε αυτό, θα πρέπει να ξέρετε λίγα περισσότερα πράγματα για την συνάρτηση printf. Ας δούμε ένα παράδειγμα. Η εντολή printf (“This prints the number %d”, 99); εμφανίζει «This prints the number 99» στην οθόνη. Όπως βλέπετε, αυτή η κλήση της συνάρτησης printf δεν περιέχει ένα, αλλά δυο ορίσματα. Το πρώτο όρισμα είναι αλφαριθμητικό (περικλείεται σε εισαγωγικά) και το δεύτερο είναι η σταθερά 99. Το δεύτερο όρισμα μπορεί να είναι σταθερά, μεταβλητή ή

αριθμητική έκφραση. Στην περίπτωση της μεταβλητής εμφανίζεται η τιμή της, ενώ στην περίπτωση της έκφρασης υπολογίζεται η τιμή της και εμφανίζεται το αποτέλεσμα. Παρατηρήστε ότι τα ορίσματα χωρίζονται μεταξύ τους με ένα κόμμα. Γενικά, όταν υπάρχουν περισσότερα από ένα ορίσματα σε μια συνάρτηση, τα ορίσματα διαχωρίζονται μεταξύ τους με κόμματα. Ο τρόπος λειτουργίας της συνάρτησης printf είναι ο εξής: Το πρώτο όρισμα είναι αλφαριθμητικό, περικλεισμένο σε εισαγωγικά, το οποίο μπορεί να περιέχει είτε χαρακτήρες, είτε χαρακτηρισμούς μορφοποίησης (**προσδιοριστές**), οι οποίοι ξεκινούν με το σύμβολο επί τοις εκατό (%). Οι χαρακτήρες του αλφαριθμητικού εμφανίζονται ως έχουν στην οθόνη με τη σειρά με την οποία αναγράφονται στο αλφαριθμητικό (όπως διαβάζεται, από τα αριστερά προς τα δεξιά). Ένας προσδιοριστής πληροφορεί τη συνάρτηση printf ότι πρέπει να εμφανίσει έναν συγκεκριμένο τύπο δεδομένου. Στο παραπάνω παράδειγμα, το %d σημαίνει ότι πρέπει να εμφανιστεί ένας ακέραιος στο δεκαδικό σύστημα, ενώ η προς εμφάνιση τιμή είναι το δεύτερο όρισμα. Η τιμή αυτή εμφανίζεται κατόπιν στην οθόνη, στο σημείο στο οποίο βρίσκεται ο **προσδιοριστής** μέσα στο αλφαριθμητικό. Για να κατανοήσετε τη σχέση μεταξύ των χαρακτήρων και των **προσδιοριστών**, εξετάστε την ακόλουθη εντολή:

printf("This displays %d, too", 99);

Σε αυτήν την περίπτωση, η κλήση στην συνάρτηση printf εμφανίζει:

This displays 99, too

Το βασικό στοιχείο εδώ είναι ότι η τιμή που σχετίζεται με έναν **προσδιοριστή** εμφανίζεται στο σημείο στο οποίο βρίσκεται ο **προσδιοριστής** μέσα στο αλφαριθμητικό.

Αν με την printf θέλουμε να εμφανίσουμε στην οθόνη περισσότερες τιμές (σταθερών, μεταβλητών ή εκφράσεων), τότε τα ορίσματα θα είναι περισσότερα (ένα για κάθε τιμή) και θα πρέπει να υπάρχουν αντίστοιχοι προσδιοριστές στο αλφαριθμητικό. Για παράδειγμα:

printf(“The difference between %d and %d is %d”, a, b, a-b);

Στο παράδειγμα αυτό, αν τα a και b έχουν τιμές 12 και 7, αντίστοιχα, στην οθόνη θα εμφανιστεί:

The difference between 12 and 7 is 5

Εάν θέλετε να καθορίσετε μια τιμή χαρακτήρων, ο **προσδιοριστής** είναι %c. Για να καθορίσετε μια τιμή κινητής υποδιαστολής, θα χρησιμοποιήσετε τον **προσδιοριστή** %f, ο οποίος λειτουργεί και για τους δυο τύπους κινητής υποδιαστολής (float και double). Όπως θα δείτε, η συνάρτηση printf έχει πολλές δυνατότητες. Για κάθε τύπο δεδομένων έχουμε τους εξής προσδιοριστές:

%d ή **%i**	Ακέραιος / int
%c	Χαρακτήρας / char (ένας μόνο)
%s	Σειρά χαρακτήρων (συμβολοσειρά)
%f	Κινητής υποδιαστολής, δηλαδή float ή double
%e	Κινητής υποδιαστολής, δηλαδή float ή double σε εκθετική μορφή
%g	Κινητής υποδιαστολής σαν %e ή %f (όποιο είναι μικρότερο)
%u	Ακέραιος χωρίς πρόσημο (unsigned int)
%o	Ακέραιος σε οκταδικό σύστημα (χωρίς πρόσημο)
%x ή **%X**	Ακέραιος σε δεκαεξαδικό σύστημα (χωρίς πρόσημο)

Συχνά οι προσδιοριστές συνοδεύονται με αριθμούς που ορίζουν το εύρος της εμφάνισης του δεδομένου. Για παράδειγμα το **%6d**, σημαίνει ότι το δεδομένο είναι ακέραιος και ότι το εύρος του είναι 6 ψηφία (δηλαδή θα εμφανιστεί σε 6 θέσεις, αφήνοντας κενά στην αρχή αν χρειάζεται), ενώ το **%7.2f,** σημαίνει ότι το δεδομένο είναι κινητής υποδιαστολής και έχει εύρος 7 ψηφία με 2 ψηφία μετά την υποδιαστολή. Επίσης ο χαρακτήρας "+" μπροστά από τον προσδιοριστή (για παράδειγμα %+d) εμφανίζει το πρόσημο, ο χαρακτήρας "-" (πχ %-d) στοιχίζει αριστερά το αποτέλεσμα, ενώ το μηδέν (πχ %08d) γεμίζει με μηδέν τα κενά που είναι άδεια μπροστά από το αποτέλεσμα.

Άσκηση αυτοαξιολόγησης - S2_LA5LO37

Να γράψετε τι θα εμφανίσει το παρακάτω πρόγραμμα. Αφού το εκτελέσετε στον υπολογιστή σας, να επιβεβαιώσετε τις απαντήσεις σας.

```
#include <stdio.h>

main ()

{

    int i = 123;

    double f = 3.1415926535;

    printf ("i = %i\n", i);

    printf ("i = %o\n", i);

    printf ("i = %x\n", i);

    printf ("i = %X\n", i);

    printf ("i = %+i\n", i);

    printf ("i = %8i\n", i);

    printf ("i = %08i\n", i);

    printf ("i = %+08i\n", i);

    printf ("f = %f\n", f);

    printf ("f = %10.3f\n", f);

    printf ("f = %+10.3f\n", f);

    printf ("f = %g\n", f);

    printf ("f = %10.6g\n", f);

    printf ("f = %10.6e\n", f);

}
```

Άσκηση αυτοαξιολόγησης S2_LA5LO38

Αντιστοιχίστε τους παρακάτω προσδιοριστές με τους κατάλληλους τύπους:

1. %c	Οκταδικός ακέραιος
2. %f	Κινητής υποδιαστολής

3. %s	Χαρακτήρας
4. %o	Κινητής υποδιαστολής
5. %x	Δεκαεξαδικός ακέραιος
6. %d	Ακέραιος
7. %e	Χαρακτήρας
8. %g	Χαρακτήρας

Άσκηση αυτοαξιολόγησης - S2_LA5LO39

Να γράψετε τι θα εμφανίσει το παρακάτω πρόγραμμα. Αφού το εκτελέσετε στον υπολογιστή σας, να επιβεβαιώσετε τις απαντήσεις σας (εφαρμογή της printf() με int και float).

```
#include <stdio.h>
    main()
    {
    int index;
    float num;
    index = 13;
    printf ("The value of index is: %d\n", index);
    num = 27.567;
    printf ("The value of num is: %f\n", num);
    printf ("Both index and num:\n");
    printf ("index -> %d and num -> %f \n", index, num);
}
```

Άσκηση αυτοαξιολόγησης - S2_LA6LO40

Να συμπληρώσετε τη στήλη 3 με τα αποτελέσματα των εντολών της στήλης 2.

α/α	**ΕΝΤΟΛΕΣ ΤΗΣ C**	**ΑΠΟΤΕΛΕΣΜΑΤΑ**
1	index=5; printf(" %d", index);	
2	index=5; printf("The value of variable index is: %d", index);	
3	index=5; printf("The value of variable index is:\n %d", index);	

4	x=5;
	y=3;
	printf("%d + %d = %d", x, y, (x+y));	

5	x=5;
	y=3;
	printf("x=%d /n y=%d /n x+y= %d", x, y, (x+y));	

Άσκηση αυτοαξιολόγησης - S2_LA6LO41

Δημιουργήστε ένα πρόγραμμα που θα εμφανίζει στην οθόνη του υπολογιστή τα παρακάτω κατά σειρά:

• Τον ακέραιο αριθμό 5

• Τον πραγματικό αριθμό 5

• Τον πραγματικό αριθμό 5.01234567890123456789

• Τον πραγματικό αριθμό 5.01234567890123456789 σε απλή γραφή με δύο δεκαδικά ψηφία

• Τον χαρακτήρα N

Και η έξοδος να είναι η εξής:

O akeraios arithmos x exei timh: 5

O pragmatikos arithmos y exei timh: 5.000000

O pragmatikos arithmos z exei timh: 5.012346

O pragmatikos arithmos z exei timh: 5.01

O xaraktiras einai o N

Ερωτήσεις αντικειμενικού τύπου - S2_LA6LO42

1) Δίνονται οι κάτωθι εντολές. Τι τυπώνει το συγκεκριμένο κομμάτι του προγράμματος;

metavliti=5;

printf(" %d", metavliti);

i. 5
ii. metavliti 5
iii. 5;
iv. 5 metavliti

2) Δίνονται οι κάτωθι εντολές. Τι τυπώνει το συγκεκριμένο κομμάτι του προγράμματος;

metavliti=5;

printf("The value of variable metavliti is: %d", metavliti);

i. The value of variable metavliti is: metavliti;
ii. The value of variable metavliti is:5
iii. The value of variable metavliti is:5;
iv. The value of variable metavliti is: metavliti

3) Δίνονται οι κάτωθι εντολές. Τι τυπώνει το συγκεκριμένο κομμάτι του προγράμματος;

```
metavliti=5;
printf("The value of variable metavliti is:\n %d", metavliti);
```

i. The value of variable metavliti is: metavliti;
ii. The value of variable metavliti is:5
iii. The value of variable metavliti is:5;
iv. The value of variable metavliti is:

4) Δίνονται οι κάτωθι εντολές. Τι τυπώνει το συγκεκριμένο κομμάτι του προγράμματος;

```
x=5;
y=3;
printf("%d + %d = %d", x, y, (x+y));
```

i. 5+3 = 8
ii. 3+5=8
iii. 5,3,8
iv. 5,3,5+3

5) Δίνονται οι κάτωθι εντολές. Τι τυπώνει το συγκεκριμένο κομμάτι του προγράμματος;

```
x=5;
y=3;
printf("x=%d /n y=%d /n x+y= %d", x, y, (x+y));
```

i. x=5 y=3 x+y=8
ii. 5, 3, 8
iii. x=5 y=3 (x+y)=8
iv. x=5
y=3
x+y=8

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA6LO43**
Τίτλος	**Εντολές εισόδου - συνάρτηση scanf**
Τίτλος δραστηριότητας	Εντολές εισόδου/εξόδου
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται η συνάρτηση εισόδου scanf και πώς αυτή χρησιμοποιείται για την ανάγνωση από το χρήστη μεταβλητών διαφόρων τύπων δεδομένων. Δίνεται ο γενικός τύπος της συνάρτησης και περιγράφονται αναλυτικά τα ορίσματά της. Τέλος, δίνονται παραδείγματα εφαρμογής της συνάρτησης για μια ακέραια μεταβλητή και μια μεταβλητή κινητής υποδιαστολής.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία, Ασκήσεις αυτοαξιολόγησης
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	Συνάρτηση scanf, προσδιοριστής
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τουλάχιστον τρία παραδείγματα εισόδου της γλώσσας προγραμματισμού C • να εξηγήσουν τη χρήση της συνάρτησης εισόδου της γλώσσας προγραμματισμού C • να κατασκευάσουν ένα παράδειγμα εισόδου ενός ακεραίου από το χρήστη

Για να χρησιμοποιήσετε τη συνάρτηση **scanf** για το διάβασμα (την είσοδο) μιας ακέραιας τιμής από το πληκτρολόγιο, θα πρέπει να την καλέσετε με την ακόλουθη γενική μορφή:

scanf("%d", &όνομα-μεταβλητής);

όπου το όνομα-μεταβλητής είναι το όνομα της ακέραιας μεταβλητής (int) που θέλετε να λάβει την τιμή που διαβάζει η scanf. Το πρώτο όρισμα της scanf είναι ένα αλφαριθμητικό το οποίο καθορίζει πώς θα αντιμετωπιστεί το δεύτερο όρισμα. Σε αυτή την περίπτωση το **%d** καθορίζει ότι το δεύτερο όρισμα θα λάβει μια ακέραια τιμή στο δεκαδικό σύστημα. Για παράδειγμα, το ακόλουθο κομμάτι κώδικα διαβάζει έναν ακέραιο τον οποίο εισάγει ο χρήστης από το πληκτρολόγιο.

int num;

scanf(“%d”, &num);

Το σύμβολο & που προηγείται του ονόματος της μεταβλητής επιτρέπει στη συνάρτηση scanf() να καταχωρήσει την τιμή που θα εισαγάγει ο χρήστης σε ένα από τα ορίσματά της.

Στο σημείο αυτό θα πρέπει να κατανοήσετε ένα σημαντικό στοιχείο. Όταν εισάγετε έναν αριθμό από το πληκτρολόγιο, ουσιαστικά πληκτρολογείτε μια ακολουθία ψηφίων. Η συνάρτηση **scanf** περιμένει μέχρι να πατήσετε το πλήκτρο enter πριν μετατρέψει αυτή την ακολουθία στην εσωτερική δυαδική μορφή που χρησιμοποιεί ο υπολογιστής.

Για να διαβάσετε έναν αριθμό κινητής υποδιαστολής από το πληκτρολόγιο, θα πρέπει να καλείτε τη συνάρτηση με την ακόλουθη γενική μορφή:

scanf("%f", &όνομα-μεταβλητής);

όπου το **όνομα-μεταβλητής** είναι το όνομα της μεταβλητής που δηλώνεται ως float. Εάν θέλετε να εισάγετε έναν αριθμό από το πληκτρολόγιο σε μια μεταβλητή τύπου double θα πρέπει να χρησιμοποιήσετε τον προσδιοριστή μορφοποίησης **%lf.**

Παραδείγματα για τη συνάρτηση εισόδου scanf:

/* Διαβάζει από το πληκτρολόγιο την ακέραια μεταβλητή x */

printf("Enter first number: ");

scanf("%d", &x);

Καλό είναι κάθε κλήση της scanf() να συνοδεύεται από μία κλήση της printf(), όπως στο προηγούμενο παράδειγμα (η printf να μπαίνει πριν τη scanf), ώστε ο χρήστης να γνωρίζει τι δεδομένο να εισάγει με το πληκτρολόγιο.

/* Διαβάζει από το πληκτρολόγιο μία μεταβλητή κινητής υποδιαστολής (fnum)

και μία ακέραια μεταβλητή (inum) */

printf ("Give me 1 float and 1 integer number:");

scanf("%f %d", &fnum, &inum);

Η κλήση:

n = scanf("%d%f", &i, &x);

όταν η γραμμή εισαγωγής είναι **25 54.324** και οι δηλώσεις των μεταβλητών είναι: **int i, n; float x;** θα έχει ως αποτέλεσμα η μεταβλητή **n** να πάρει την τιμή 2, η μεταβλητή **i** να πάρει την τιμή 25 και η μεταβλητή **x** να πάρει την τιμή 54.324.

Το πιο συνηθισμένο λάθος: π.χ. να δώστε scanf("%d", x) αντί για **scanf("%d", &x)**, δηλαδή να ξεχάσετε τη διεύθυνση της μεταβλητής.

Άσκηση αυτοαξιολόγησης - S2_LA6LO44

Να γράψετε τι θα εμφανίσει το παρακάτω πρόγραμμα. Αφού το εκτελέσετε στον υπολογιστή σας, να επιβεβαιώσετε τις απαντήσεις σας.

```
#include <stdio.h>
main(){
int index;
float num;
printf ("Give me the integer index: ");
scanf("%d", &index);
printf (" Give me the integer index: ");
scanf("%d", &index);
printf("The value of index is %d\n", index);
num = 27.567;
printf("The value of num is %f\n", num);
}
```

Άσκηση αυτοαξιολόγησης - S2_LA6LO45

Γράψτε ένα πρόγραμμα το οποίο θα εισάγει δυο αριθμούς κινητής υποδιαστολής και κατόπιν θα εμφανίζει το άθροισμά τους.

Μαθησιακό Αντικείμενο	
Όνομα	**S2_LA6LO46**
Τίτλος	**Το 1ο πρόγραμμα**
Τίτλος δραστηριότητας	Αξιολόγηση ενότητας **S2**
Περιγραφή	Στην ενότητα αυτή, θα σας ζητηθεί να αναπτύξετε και το πρώτο δικό σας πρόγραμμα σε C. Αυτό σημαίνει ότι, πρέπει να εκτελέσετε τα βήματα της φάσης της υλοποίησης και να παράγετε τον εκτελέσιμο κώδικα, τον οποίο πρέπει να εκτελέσετε για να διαπιστώσετε αν πράγματι το πρόγραμμά σας κάνει τις λειτουργίες που εσείς θέλετε. Αν όχι, εντοπίστε τα λάθη, διορθώστε τα και επαναλάβετε τη διαδικασία έως ότου το πρόγραμμά σας ικανοποιήσει τις απαιτήσεις του προβλήματος.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Παρατηρήσεις, Ασκήσεις αυτοαξιολόγησης
Τεχνικός Τύπος	Κείμενο, παρουσίαση
Λέξεις Κλειδιά	Προεπεξεργαστής, επικεφαλίδα, σώμα προγράμματος, σχόλια, μεταγλώττιση
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν τα βασικά μέρη ενός προγράμματος στη γλώσσα προγραμματισμού C. • να εξηγήσουν το λόγο ύπαρξης των εντολών της επικεφαλίδας ενός προγράμματος στη γλώσσα C. • να εξηγήσουν το λόγο ύπαρξης των σχολίων σε ένα πρόγραμμα. • να μπορούν να δημιουργήσουν μικρά και ευανάγνωστα προγράμματα στη γλώσσα προγραμματισμού C.

Μια συνάρτηση στη C αποτελείται από μια επικεφαλίδα και ένα σώμα.

Η επικεφαλίδα περιέχει τις προτάσεις του προεπεξεργαστή, όπως είναι η #include, και το όνομα της συνάρτησης. Η γραμμή #include <stdio.h> είναι οδηγία προς τον προεπεξεργαστή της C να συμπεριλάβει σε εκείνο το σημείο τα περιεχόμενα του αρχείου επικεφαλίδας (header file) stdio.h, το οποίο περιέχει χρήσιμες δηλώσεις για τις συναρτήσεις εισόδου/εξόδου.

Το σώμα της συνάρτησης βρίσκεται μέσα στα άγκιστρα { … } και αποτελείται από μια σειρά προτάσεων που η κάθε μια τελειώνει μ' ένα ελληνικό ερωτηματικό «;». Μέσα στο σώμα της συνάρτησης μπορεί να υπάρχουν προτάσεις δήλωσης, καταχώρησης και κλήσης άλλων συναρτήσεων. Κάθε πρόγραμμα C πρέπει να περιέχει ακριβώς μία συνάρτηση με όνομα main, που είναι αυτή από την οποία θα αρχίσει να εκτελείται το εκτελέσιμο πρόγραμμα που θα προκύψει από τη μεταγλώττιση.

Εισαγωγικές Παρατηρήσεις

Προσοχή στη γλώσσα προγραμματισμού *C*, υπάρχει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων.

Χρησιμοποιήστε εκφραστικά ονόματα μεταβλητών και σταθερών, έτσι ώστε να βοηθήσουν στην αναγνωσιμότητα του προγράμματος

Σχολιάστε τον κώδικα με τη χρήση των /*…..*/

Άσκηση αυτοαξιολόγησης - S2_LA6LO47

Αντιστοιχίστε τις παρακάτω ενέργειες με τις σωστές προτάσεις ξαναδιαβάζοντας το πρόγραμμα που παρουσιάστηκε στη βασική παρουσίαση.

1. *Ζήτα από το χρήστη ένα χαρακτήρα.*	α. printf("ο ASCII κώδικας του χαρακτήρα είναι %c είναι %d\n", next_ch, next_ch);
2. *Τύπωσέ τον μαζί με τον κωδικό του.*	β. printf("Δώσε ένα χαρακτήρα:\t");
3. *Βρες τον επόμενο χαρακτήρα*	γ. scanf("%c", &ch);
4. *Τύπωσε τον επόμενο χαρακτήρα και τον ASCII κωδικό του*	δ. printf("ο ASCII κώδικας του χαρακτήρα %c είναι %d\n", ch, ch);
5. *Πάρε από το χρήστη το χαρακτήρα.*	ε. next_ch = ch + 1;

Άσκηση αυτοαξιολόγησης - S2_LA6LO48

Να φτιάξετε ένα πρόγραμμα που ο χρήστης θα δίνει έναν αριθμό ο οποίος κατόπιν εκτυπώνεται ως χαρακτήρας και ως αριθμός στον κώδικα ASCII.

Άσκηση αυτοαξιολόγησης - S2_LA6LO49

Να φτιάξετε ένα πρόγραμμα που θα τυπώνονται δύο ακέραιες μεταβλητές, οι οποίες κατόπιν θα παίρνουν τιμές από τον χρήστη και μετά θα ξανατυπώνονται.

Άσκηση αυτοαξιολόγησης - S2_LA6LO50

Να φτιάξετε ένα πρόγραμμα που θα τυπώνονται δύο πραγματικές μεταβλητές, οι οποίες κατόπιν θα παίρνουν τιμές από τον χρήστη και μετά θα ξανατυπώνονται.

Άσκηση αυτοαξιολόγησης - S2_LA6LO51

Αναπτύξτε ένα πρόγραμμα το οποίο θα ζητά από το χρήστη δυο ακεραίους, θα υπολογίζει το άθροισμά τους και στη συνέχεια θα τυπώνει: “οκταδική μορφή δεκαδική μορφή δεκαεξαδική μορφή” και στην αμέσως επόμενη γραμμή το άθροισμα που υπολόγισε στις αντίστοιχες μορφές.

Άσκηση αυτοαξιολόγησης - S2_LA6LO52

Αναπτύξτε ένα πρόγραμμα το οποίο θα ζητά από το χρήστη έναν αριθμό από το 66 ως το 90. Στη συνέχεια θα τυπώνει τρεις διαδοχικούς χαρακτήρες με τους αντίστοιχους ASCII κωδικούς τους, με τον ενδιάμεσο χαρακτήρα να είναι αυτός που έχει σαν ASCII κωδικό τον αριθμό που έδωσε ο χρήστης.

3.3 Ενότητα 3 - Τελεστές - Εκφράσεις - προτάσεις

Τίτλος Ενότητας	**S3** - Τελεστές - Εκφράσεις - Προτάσεις
Τίτλος Μαθήματος	Βασικές αρχές προγραμματισμού με τη γλώσσα C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστεί η βασική έννοια των τελεστών και του τρόπου που αυτοί χρησιμοποιούνται αφενός μεν για το σχηματισμό εκφράσεων, αφετέρου δε για τον υπολογισμό της τιμής αυτών των εκφράσεων. Η C διαθέτει πλούσιο σύνολο τελεστών, πλουσιότερο από άλλες γλώσσες προγραμματισμού, για τους οποίους, επιπλέον, επιτρέπει την ανάμειξη των τύπων της σχεδόν χωρίς περιορισμούς. Εισάγεται ακόμη η πρόταση ως βασική μονάδα δόμησης του διαδικασιακού προγραμματισμού.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να κατηγοριοποιούν τους τελεστές της γλώσσας, ως προς τον αριθμό των τελεστέων που επιδρούν • να κατηγοριοποιούν τους τελεστές της γλώσσας, ως προς τις διεργασίες που επιτελούν • να εξηγούν τη σημασία των επιπέδων προτεραιότητας των τελεστών • να εξηγούν με παραδείγματα τη σημασία της προσεταιριστικότητας των τελεστών • να κατανοούν τη διαφορά ανάμεσα σε μια έκφραση και σε μια πρόταση στη γλώσσα C
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν οι εκπαιδευόμενοι θα έχουν μελετήσει το κεφάλαιο αυτό, θα μπορούν: • να αναφέρουν τουλάχιστον 3 κατηγορίες τελεστών της γλώσσας προγραμματισμού C • να αναφέρουν τους αριθμητικούς τελεστές της γλώσσας προγραμματισμού C • να αναφέρουν τους 3 λογικούς τελεστές της γλώσσας προγραμματισμού C • να αναφέρουν τους τελεστές σύγκρισης της γλώσσας προγραμματισμού C • να αναφέρουν τους τελεστές διαχείρισης bits της γλώσσας προγραμματισμού C • να αναφέρουν τη διαφορά της εντολής ++i με την i++, όπου i μια μεταβλητή τύπου ακεραίου

	• να αναφέρουν τη χρήση του τριαδικού τελεστή • να αναφέρουν την έννοια της προτεραιότητας των τελεστών χρησιμοποιώντας παραδείγματα • να δημιουργούν εκφράσεις και να υπολογίζουν την τιμή τους • να δίνουν 3 τουλάχιστον παραδείγματα αυτόματης μετατροπής τύπων • να δίνουν από 2 τουλάχιστον εκφράσεις για κάθε έναν από τους τελεστές της C
Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: Στην πρώτη, παρουσιάζουμε τις κατηγορίες τελεστών της γλώσσας C ανάλογα με τη διεργασία την οποία εκτελούν. Στην δεύτερη, παρουσιάζουμε τις κατηγορίες τελεστών της γλώσσας C ανάλογα με τον αριθμό των τελεστέων στους οποίους δρουν. Στην τρίτη, παρουσιάζουμε την έννοια της έκφρασης και της πρότασης. Επίσης περιγράφουμε τις μετατροπές τύπων της γλώσσας προγραμματισμού και δοκιμάζουμε απλά προγράμματα
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας και των εκπαιδευόμενων.
Αξιολόγηση	Η ενότητα του μαθήματος αυτού χρησιμοποιεί σε κάθε δραστηριότητα και σε κάθε μαθησιακό αντικείμενο ασκήσεις αυτοαξιολόγησης. Τέτοιες ασκήσεις μπορεί να είναι ερωτήσεις επίλυσης προβλημάτων ή ερωτήσεις αντικειμενικού τύπου. Για τη διεκπεραίωση αυτών των ασκήσεων μπορεί να χρησιμοποιηθούν διάφορα εργαλεία λογισμικού όπως προγράμματα σχεδιασμού ερωτήσεων συμπλήρωσης κενού, σταυρόλεξα κλπ. ή λογισμικό σχεδιασμού ανάπτυξης προγραμμάτων στη γλώσσα C.
Συνολικός χρόνος Ενότητας	1 εβδομάδα

3.3.1 Δραστηριότητα 1 - Κατηγορίες τελεστών ανάλογα με τη λειτουργία τους

Τίτλος Δραστηριότητας	***S3_LA7*** - Κατηγορίες τελεστών ανάλογα με τη διεργασία που εκτελούν
Τίτλος Ενότητας	Τελεστές - Εκφράσεις - Προτάσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει τους τελεστές της γλώσσας C, η οποία έχει το προνόμιο να διαθέτει ένα πολύ πλούσιο σύνολο τελεστών πλουσιότερο από αυτό των άλλων γλωσσών προγραμματισμού. Οι τελεστές κατηγοριοποιούνται ανάλογα με τη διεργασία που εκτελούν και αναφέρονται με παραδείγματα ανά κατηγορία.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να διακρίνουν τους τελεστές από τους τελεστέους μέσα σε μια έκφραση • να εξηγήσουν τη χρήση των αριθμητικών και των λογικών τελεστών • να εξηγήσουν τη χρήση των συγκριτικών τελεστών ή τελεστών σύγκρισης • να εξηγήσουν τη χρήση τελεστών ανάθεσης ή καταχώρησης τιμής • να εξηγήσουν τη διαφορά μεταξύ του τελεστή ανάθεσης (=) και του τελεστή σύγκρισης (==) • να εξηγήσουν τη χρήση των τελεστών διαχείρισης bits • να εξηγήσουν τη χρήση του τελεστή υπολογισμού υπολοίπου (%)
Μαθησιακά Αντικείμενα	Τελεστής καταχώρησης τιμής Αριθμητικοί τελεστές Λογικοί τελεστές Συγκριτικοί τελεστές Τελεστές διαχείρισης των bits
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 1, 3, 4, 3 και 5 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για την θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S3_LA7LO53, S3_LA7LO55, S3_LA7LO59, S3_LA7LO64 και S3_LA7LO68, αντίστοιχα.
Λέξεις Κλειδιά	Τελεστής αριθμητικός, λογικός, συγκριτικός, διαχείρισης των bits, τελεστής ανάθεσης, τελεστής υπολογισμού υπολοίπου

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA7LO53**
Τίτλος	**Τελεστής καταχώρησης τιμής**
Τίτλος δραστηριότητας	*Κατηγορίες τελεστών* ανάλογα με τη διεργασία που εκτελούν
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι έννοιες του τελεστή και του τελεστέου όπως επίσης και ο τελεστής καταχώρησης τιμής. Σχηματίζουμε απλές εκφράσεις με συνδυασμό τελεστών και τελεστέων, το αποτέλεσμα των οποίων αναθέτουμε σε μια μεταβλητή χρησιμοποιώντας τον τελεστή καταχώρησης τιμής.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, παρουσίαση
Λέξεις Κλειδιά	Τελεστής καταχώρησης τιμής, τελεστέος, ανάθεση τιμής
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να διακρίνουν τους τελεστές από τους τελεστέους μέσα σε μια έκφραση • να εξηγήσουν τη χρήση των τελεστών ανάθεσης • να δημιουργήσουν τουλάχιστον 3 παραδείγματα προτάσεων με τον τελεστή καταχώρησης τιμής. • να εξηγήσουν τη διαφορά του τελεστή ανάθεσης (=) με τον τελεστή σύγκρισης (==)

Ένας τελεστής (operator) είναι ένα σύμβολο ή μία λέξη της γλώσσας προγραμματισμού, που αναπαριστά συγκεκριμένη διεργασία, η οποία εκτελείται πάνω σε ένα ή περισσότερα δεδομένα. Τα δεδομένα καλούνται τελεστέοι (operands) και μπορούν να είναι μεταβλητές, σταθερές ή ακόμη κλήσεις συναρτήσεων. Οι τελεστές χρησιμοποιούνται για το σχηματισμό εκφράσεων (expressions). Στην έκφραση num + 12 ο χαρακτήρας + αναπαριστά τη διεργασία της πρόσθεσης των δύο τελεστέων, της τιμής της μεταβλητής num και της σταθεράς 12.

Ο προγραμματιστής χρησιμοποιεί τους τελεστές για να επεξεργαστεί δεδομένα και να πάρει τα αποτελέσματα της επεξεργασίας. Αυτό το πετυχαίνει κατασκευάζοντας εκφράσεις. Για να εκτελέσει, για παράδειγμα, την πρόσθεση των μεταβλητών num1 και num2 κατασκευάζει την έκφραση num1 + num2 κάνοντας χρήση του αριθμητικού τελεστή +. Το = είναι ένας τελεστής που ήδη έχουμε δει και είναι ένας **τελεστής καταχώρησης τιμής**. Πρέπει να έχουμε υπόψη μας ότι δεν μπορούμε να καταχωρήσουμε τιμή σε μια σταθερά και το μέρος

στα αριστερά του συμβόλου = πρέπει να είναι το όνομα της μεταβλητής και αναφέρεται σε μία θέση μνήμης.

Εφαρμογές της εντολής αυτής είναι οι εξής:

```
year = 2006;
i = i + 1;
timi01 = timi02 = timi03 = 68;
```

Στην τελευταία εντολή οι καταχωρήσεις γίνονται από τα δεξιά προς τα αριστερά.

Άσκηση αυτοαξιολόγησης - S3_LA7LO54

Ποιο θα είναι το αποτέλεσμα της εκτέλεσης της παρακάτω εντολής ανάθεσης;

```
d = (a = 4) * (c = 8);
```

Τι θα συμβεί αν αναιρέσουμε όλες τις παρενθέσεις;

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA7LO55**
Τίτλος	**Αριθμητικοί τελεστές**
Τίτλος δραστηριότητας	*Κατηγορίες τελεστών* ανάλογα με τη διεργασία που εκτελούν
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι αριθμητικοί τελεστές με παραδείγματα εφαρμογής τους σε αριθμητικές εκφράσεις. Επίσης περιγράφονται οι διαιρέσεις στη γλώσσα C, τι συμβαίνει όταν έχουμε διαίρεση δεδομένων διαφορετικού τύπου καθώς και ο τελεστής του υπολοίπου με παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Τελεστής πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, διαίρεσης και ακέραιου υπολοίπου.
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη χρήση των αριθμητικών τελεστών • να εξηγήσουν τη χρήση του τελεστή υπολογισμού υπολοίπου (%) • να εξηγήσουν τη διαδικασία της αποκοπής • να εξηγήσουν τον τύπο δεδομένων που δίνει μια έκφραση

Εκτός από τον αριθμητικό τελεστή + για τη διεργασία της πρόσθεσης, έχουμε ακόμα τους αριθμητικούς τελεστές αφαίρεσης -, πολλαπλασιασμού *, διαίρεσης / και ακέραιου υπολοίπου %.

Ο τελεστής ακεραίου υπολοίπου (%) χρησιμοποιείται στην αριθμητική των ακεραίων και επιστρέφει το υπόλοιπο της ακέραιας διαίρεσης του ακεραίου στα αριστερά με τον ακέραιο στα δεξιά. Για παράδειγμα, η πράξη 13 % 5 δίνει ως αποτέλεσμα την τιμή 3, αφού το 5 χωράει δύο φορές στο 13 και έχει υπόλοιπο 3.

Επίσης, στη διαίρεση πρέπει να έχουμε υπόψη μας ότι η διαίρεση με αριθμούς πραγματικούς δίνει ως αποτέλεσμα πραγματικό αριθμό, ενώ η διαίρεση μεταξύ ακεραίων δίνει ως αποτέλεσμα ακέραιο αριθμό. Έτσι, αν η διαίρεση μεταξύ ακεραίων δεν είναι τέλεια, η C απορρίπτει το κλασματικό μέρος του πηλίκου χωρίς να το στρογγυλοποιεί. Αυτή η

διαδικασία λέγεται **αποκοπή**. ***Όταν σε μία έκφραση εμπλέκονται ακέραιοι και πραγματικοί αριθμοί, το αποτέλεσμα είναι πραγματικός αριθμός***.

Όταν διαιρούμε δύο ακεραίους, τότε με τον τελεστή / παίρνουμε το πηλίκο της ακέραιας διαίρεσης και με τον τελεστή % το υπόλοιπο.

p = a / 3;

ypol = a % 3;

Για να μην κάνουμε ακέραια διαίρεση, αλλά να προκύψει δεκαδικός αριθμός, πρέπει ο ένας τουλάχιστον τελεστέος να είναι δεκαδικός:

p = a / 3.0;

Άσκηση αυτοαξιολόγησης - S3_LA7LO56

Τι θα εμφανίσει το παρακάτω πρόγραμμα:

```
/* οι διαιρέσεις στη C */
#include < stdio.h>
main()
{
    printf("ακέραια διαίρεση: 5/3 είναι %d \n", 5/3);
    printf("ακέραια διαίρεση: 8/4 είναι %d \n", 8/4);
    printf("ακέραια διαίρεση: 7/5 είναι %f\n", 7/5);
    printf("διαίρεση κινητής υποδιαστολής: 7./4. είναι %1.2f \n", 7./4.);
    printf("μικτή διαίρεση: 7./4 είναι %1.2f \n", 7./4);
}
```

Άσκηση αυτοαξιολόγησης - S3_LA7LO57

Το παρακάτω πρόγραμμα μετατρέπει τα δευτερόλεπτα σε λεπτά και σε δευτερόλεπτα. Αν ο αριθμός sec που θα δώσει ο χρήστης είναι το 152, για να υπολογίσουμε πόσα λεπτά υπάρχουν σ' έναν αριθμό δευτερολέπτων sec, παίρνουμε το πηλίκο της ακέραιας διαίρεσης του sec με το SEC_PER_MIN, που είναι ουσιαστικά το 60, και για να βρούμε πόσα δευτερόλεπτα περίσσευσαν που δεν χώρεσαν σ' ένα λεπτό, παίρνουμε το υπόλοιπο της ακέραιας διαίρεσης του sec με το SEC_PER_MIN. Με βάση όσα αναφέρθηκαν τι θα εμφανίσει το συγκεκριμένο πρόγραμμα;

```
#include <stdio.h>
#define SEC_PER_MIN 60 /* 60 δευτερόλεπτα σ' ένα λεπτό */
main()
{
    int sec, min, left;
    printf("Μετατροπή δευτερολέπτων σε λεπτά και δευτερόλεπτα \n");
```

```
    printf("Δώστε τον αριθμό των δευτερολέπτων: \n");
    scanf("%d", &sec); /* διάβασμα του αριθμού των δευτερολέπτων */
    min = sec / SEC_PER_MIN; /* αριθμός λεπτών */
    left = sec % SEC_PER_MIN; /*αριθμός δευτερολέπτων που έμειναν*/
    printf("%d δευτερόλεπτα είναι %d λεπτά και %d δευτερόλεπτα.\n", sec, min, left);
}
```

Άσκηση αυτοαξιολόγησης - S3_LA7LO58

Συμπληρώστε τον πίνακα που ακολουθεί. Στην συνέχεια γράψτε ένα πρόγραμμα για να επιβεβαιώσετε τις τιμές που έχετε υπολογίσει.

int x = 5, y = -6, z = 8, w = -7, u = 11;

α/α	**Έκφραση**	**Τιμή**
1	x/y/z	
2	9 + z / - -w * u	
3	4 * x % - y + z - 2	
4	23 / - - - u - +12 % z	

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA7LO59**
Τίτλος	**Λογικοί τελεστές**
Τίτλος δραστηριότητας	*Κατηγορίες τελεστών* ανάλογα με τη διεργασία που εκτελούν
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι λογικοί τελεστές που χρησιμοποιούνται στη γλώσσα προγραμματισμού C, ποιοι είναι, ποιο το αποτέλεσμα μιας λογικής έκφρασης και εφαρμογές με παραδείγματα. Η C δεν έχει λογικούς τύπους δεδομένων, αλλά αν το αποτέλεσμα μιας σύγκρισης είναι αληθές ή ψευδές, τότε επιστρέφει αποτέλεσμα 1 ή 0, αντίστοιχα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	τελεστής !, τελεστές && και ||
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη χρήση των λογικών τελεστών • να δώσουν τουλάχιστον τρία παραδείγματα απλών και σύνθετων λογικών εκφράσεων

Για τη σύνδεση λογικών εκφράσεων η C παρέχει τους λογικούς τελεστές ! (NOT), && (AND) και || (OR).

Οι τελεστές && και || δρουν μεταξύ δύο λογικών ποσοτήτων ή εκφράσεων και σχηματίζουν μια νέα λογική ποσότητα ενώ ο τελεστής ! δρα σε μία, τη λογική έκφραση που τον ακολουθεί, και της αλλάζει την τιμή:

Το ! (4 > 3) είναι false (δηλαδή 0).

Το ! (4 < 3) είναι true (δηλαδή 1).

Η λογική έκφραση που σχηματίζεται συνδέοντας δύο εκφράσεις με τον τελεστή && έχει τιμή 1 (true) μόνο αν και οι δύο ποσότητες είναι αληθείς. Σε άλλη περίπτωση είναι 0 (false).

Το (4 > 3) && (3.0 > 2.0) είναι true

Το (4 < 3) && (3.0 > 2.0) είναι false

Ο τελεστής || μεταξύ δύο λογικών εκφράσεων σχηματίζει μια νέα ποσότητα με τιμή true αν έστω και μία από τις δύο ποσότητες είναι true, αλλιώς είναι false:

Το (4 > 3) || (3.0 < 2.0) είναι true

Το (4 < 3) || (3.0 < 2.0) είναι false

Η C δεν έχει λογικούς τύπους δεδομένων. Αν το αποτέλεσμα μιας σύγκρισης είναι αληθές, τότε επιστρέφει αποτέλεσμα 1, ενώ αν είναι ψευδές επιστρέφει αποτέλεσμα 0.

```
a = 5 > 3;          /* το a γίνεται ίσο με 1 */
b = 4 > 10;         /* το b γίνεται ίσο με 0 */
```

Άσκηση αυτοαξιολόγησης - S3_LA7LO60

Να αντιστοιχίσετε το αποτέλεσμα κάθε μιας εντολής της στήλης 1 με το αποτέλεσμα αυτής στην στήλη 2 του παρακάτω πίνακα:

Πρόταση	**/* Τυπώνει ….. */**
printf("%d\n", 1 + 1 == 2);	/* Τυπώνει 1 */
printf("%d\n", 1 > 2);	
printf("%d\n", 5 != 5);	
printf("%d\n", 1 <= 5);	/* Τυπώνει 0 */
printf("%d\n", 1 <= 1);	
printf("%d\n", 1 <= 0);	

Άσκηση αυτοαξιολόγησης - S3_LA7LO61

Συμπληρώστε τον πίνακα που ακολουθεί:

int x = 3, w = 6, u = 4, v = 2;

float y = 2.0;

α/α	**Έκφραση**	**Τιμή**
1	x > w && u > v	
2	x <!w*!x	
3	x + w <!u + v	
4	x - y * w/u && w * x	

Άσκηση αυτοαξιολόγησης - S3_LA7LO62

Δημιουργήστε ένα πρόγραμμα που θα ζητά από τον χρήστη δύο ακέραιους αριθμούς και θα εμφανίζει τη λογική τους άρνηση.

Άσκηση αυτοαξιολόγησης - S3_LA7LO63

Τι εμφανίζει το παρακάτω πρόγραμμα αν ο χρήστης δώσει τις τιμές 12 και 10;

```
#include <stdio.h>
int main(void)
{
    int k, m;
    printf("Dwse ton prwto akeraio:\n");
    scanf ("%d",&k);
    printf("Dwse ton deytero akeraio:\n");
    scanf ("%d",&m);
    printf("!%d = %d\n",k,!k);
    printf("!%d = %d\n",m,!m);
    printf("Kapoios arithmos einai arnhtikos -> %d\n",((k<0)||(m<0)));
    printf("Kai oi dyo arithmoi einai arnhtikoi -> %d\n",((k<0)&&(m<0)));
    getchar();
    getchar();
    return 0;
}
```

Παρατήρηση - ανατροφοδότηση για την άσκηση

Το παραπάνω πρόγραμμα έχει ως αποτέλεσμα να:

- εμφανίζει την λογική άρνηση 2 ακεραίων
- ελέγχει αν κάποιος από αυτούς είναι αρνητικός
- ελέγχει αν και οι δύο είναι αρνητικοί

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA7LO64**
Τίτλος	Συγκριτικοί τελεστές
Τίτλος δραστηριότητας	*Κατηγορίες τελεστών* ανάλογα με τη διεργασία που εκτελούν
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι σχεσιακοί ή συγκριτικοί τελεστές της γλώσσας με τη βοήθεια των οποίων σχηματίζονται λογικές εκφράσεις με αποτέλεσμα αληθές ή ψευδές. Επίσης περιγράφεται η προτεραιότητα των τελεστών αυτών σε σχέση με τους λογικούς και τους αριθμητικούς τελεστές και δίνονται παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Τελεστής ==, τελεστής !=, μεγαλύτερο, μικρότερο, μεγαλύτερο ή ίσο, μικρότερο ή ίσο
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να σχηματίζουν τουλάχιστον 3 παραδείγματα με τελεστές σύγκρισης • να εξηγούν το αποτέλεσμα λογικών εκφράσεων • να εξηγούν την προτεραιότητα των τελεστών σύγκρισης από αυτή των λογικών τελεστών • να εξηγούν την προτεραιότητα των τελεστών σύγκρισης από αυτή των αριθμητικών τελεστών

Η C υποστηρίζει τη σύγκριση ποσοτήτων με τη βοήθεια των *σχεσιακών* ή συγκριτικών τελεστών

==	ίσο	!=	άνισο
>	μεγαλύτερο	<	μικρότερο
>=	μεγαλύτερο ή ίσο	<=	μικρότερο ή ίσο

Το αποτέλεσμα της σύγκρισης είναι λογική ποσότητα και επομένως έχει τιμή **true** ή **false**. Π.χ. το 3.0>2.0 είναι **true** ενώ το 2!=1+1 είναι **false**. Οι αριθμητικοί τελεστές έχουν μεγαλύτερη προτεραιότητα από τους σχεσιακούς.

Οι τελεστές σύγκρισης συμβολίζονται όπως και στις άλλες γλώσσες, με τη διαφορά ότι για τη σύγκριση της ισότητας έχουμε το == και για την ανισότητα το !=.

(a == 1) /* Έλεγχος ισότητας */

(a = 1) /* Προσοχή! Καταχωρεί το 1 στο a και το αποτέλεσμα είναι αληθές (1) */

(a != 1) /* έλεγχος ανισότητας */

Η προτεραιότητα των τελεστών σύγκρισης είναι μεγαλύτερη από αυτή των λογικών τελεστών ενώ η προτεραιότητα και των δύο (σύγκρισης και λογικών) είναι μικρότερη από αυτή των αριθμητικών τελεστών!

Άσκηση αυτοαξιολόγησης - S3_LA7LO65

Ποια θα είναι η τιμή της μεταβλητής i μετά την εκτέλεση του παρακάτω κώδικα;

j = k =13;

i = j = = k;

Άσκηση αυτοαξιολόγησης - S3_LA7LO66

Συμπληρώστε τον πίνακα που ακολουθεί:

int x = 3, w = 6, u = 4, v = 2;

float y = 2.0;

Έκφραση	**Τιμή**
x > w	
x <>w*x	
x + w <=u + v	
x - y +w/u >0	

Άσκηση αυτοαξιολόγησης - S3_LA7LO67

Η λογική έκφραση που ακολουθεί υπολογίζει αν ένα έτος είναι δίσεκτο ή όχι.

(etos % 4 == 0 && etos % 100 != 0 || etos % 400 == 0)

Θεωρήστε ότι δίσεκτα έτη θεωρούνται όσα διαιρούνται ακριβώς με το 4 αλλά όχι με το 100. Εξαίρεση αποτελούν τα έτη που διαιρούνται ακριβώς με το 400, που θεωρούνται επίσης δίσεκτα.

Παρατηρήστε αν η έκφραση που ακολουθεί, κάνει τήν ίδια δουλειά:

(!(etos % 4) && (etos % 100) || !(etos % 400))

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA7LO68**
Τίτλος	**Τελεστές χειρισμού bits**
Τίτλος δραστηριότητας	*Κατηγορίες τελεστών* ανάλογα με τη διεργασία που εκτελούν
Περιγραφή	Στο συγκεκριμένο ΜΑ παρουσιάζεται ο χειρισμός των bits που σημαίνει τη δυνατότητα επέμβασης στα bits ενός byte ή μιας λέξης που αντιστοιχούν στους τύπους char και int. Δίνεται πίνακας με τους βασικότερους τελεστές αυτής της κατηγορίας και μερικά παραδείγματα εφαρμογής τους.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Τελεστής διαχείρισης των bits, parity bit, shift right, shift left, συμπλήρωμα ως προς 1
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν τη χρήση των τελεστών διαχείρισης bits • να δώσουν τουλάχιστον τρία παραδείγματα εφαρμογής τους

Υπάρχουν τελεστές που αντιμετωπίζουν τα ορίσματά τους ως σύνολο bits σε σειρά, δηλαδή ως ακολουθίες από 0 ή 1. Η δράση τους ελέγχει ή θέτει την τιμή του κάθε bit χωριστά. Τα ορίσματά τους είναι μεταβλητές με τύπο ακεραίου (**short int**, **int**, **long int**, **bool**, **char**) και **enum**, **signed** ή **unsigned.**

&	Σύζευξη επιπέδου bit (bitwise AND)
|	Διάζευξη επιπέδου bit (bitwise OR)
~	Συμπλήρωμα ως προς 1
>> n	Ολίσθηση κατά *n* bits δεξιά
<< n	Ολίσθηση κατά *n* bits αριστερά
^	Αποκλειστική διάζευξη επιπέδου bit (bitwise XOR)

Οι τελεστές << και >> μετατοπίζουν τα bits του αριστερού τους ορίσματος κατά τόσες θέσεις όσες ορίζει το δεξί τους όρισμα. Τα επιπλέον bits χάνονται. Ο τελεστής << γεμίζει τις κενές θέσεις με 0 ενώ ο >> κάνει το ίδιο αν το αριστερό όρισμα είναι unsigned.

Οι τελεστές &, ^ και | επιστρέφουν ακέραιο με bit pattern που προκύπτει αν εκτελεστεί το AND, XOR, OR αντίστοιχα στα ζεύγη bit των ορισμάτων τους.

Για παράδειγμα, η παράσταση (ch & 127) όπου ch είναι τύπου char εκχωρεί την τιμή 0 στο πιο σημαντικό bit (Most Significant Bit, MSB).

```
  ch: 1 0 0 1 1 1 0 1
 127: 0 1 1 1 1 1 1 1
 ____________________
   & 0 0 0 1 1 1 0 1
```

Εφαρμογή **Bitwise Τελεστής AND**

Δυαδικό "και" σε επίπεδο bit (σύζευξη / AND). Αυτός ο δυαδικός τελεστής παράγει μια νέα τιμή κάνοντας μια σύγκριση δυαδικού ψηφίου ανά δυαδικό ψηφίο μεταξύ των δύο τελεστέων. Για κάθε θέση το αποτέλεσμα είναι **1 μόνο αν και τα δύο αντίστοιχα δυαδικά ψηφία των τελεστέων είναι 1**, δηλ. αληθή.

ΠΡΟΣΟΧΗ: Υπάρχει και ο μοναδιαίος τελεστής της διεύθυνσης (&) με τον ίδιο συμβολισμό που όταν μπαίνει μπροστά σε μία μεταβλητή αναφέρεται στη διεύθυνσή της.

int a, b, c ;	00000000 00001000 01100000 00010100
a = 548884;	&
b = 1595404;	00000000 00011000 01011000 00001100
c = a & b ;	=
// **c now i s 540676**	00000000 00001000 01000000 00000100

Εφαρμογή **Bitwise Τελεστής OR**

Δυαδικό διαζευκτικό "ή" σε επίπεδο bit (διάζευξη / OR). Η εντολή var = var | ON τοποθετεί 1 σε εκείνες τις θέσεις των bits της var για τις οποίες **υπάρχει 1** στις αντίστοιχες θέσεις των bits της ON.

int a, b, c ;	00000000 00001000 01100000 00010100
a = 548884;	|
b = 1595404;	00000000 00011000 01011000 00001100
c = a | b ;	=
// **c now i s 1603596**	00000000 00011000 01111000 00001100

Εφαρμογή **Bitwise Τελεστής Αποκλειστικού OR (XOR)**

Αποκλειστική διάζευξη "ή" (exclusive OR) σε επίπεδο bit. Δηλαδή, η εντολή var1 ^ var2 στις θέσεις που τα bits και των δύο μεταβλητών έχουν την ίδια τιμή μπαίνει 0, ενώ στις θέσεις που **τα bits έχουν διαφορετική τιμή μπαίνει 1**.

int a, b, c ;	00000000 00001000 01100000 00010100
a = 548884;	^

b = 1595404; c = a ^ b ; // **c now i s 1062936**	00000000 00011000 01011000 00001100 = 00000000 00010000 00111000 00011000

Εφαρμογή **Αριστερή Μετατόπιση (<<)**

Το αποτέλεσμα είναι το ίδιο με πολλαπλασιασμό επί δύο.

Ολίσθηση (shift) προς αριστερά. Για παράδειγμα, η εντολή i << j το i μετακινείται αριστερά j bits. Οι θέσεις των bits που αδειάζουν "γεμίζουν" με 0.

ΠΡΟΣΟΧΗ: Προσέξτε διότι δεν λαμβάνεται υπόψη το bit του προσήμου!

int a, c ;
a = 548884;
c = a << 10;
// **c is 562057216**
same as a * 2^10
	00000000 00001000 01100000 00010100
	<<
	10
	=
	00100001 10000000 01010000 00000000

Εφαρμογή **Δεξιά Μετατόπιση (>>)**

Ολίσθηση (shift) προς δεξιά. Αν η μεταβλητή είναι unsigned τότε τα κενά bits "γεμίζουν" με 0. Αν η μεταβλητή έχει πρόσημο τότε τα bits που αδειάζουν, είτε παίρνουν το bit του προσήμου (0 ή 1 - θετική ή αρνητική μεταβλητή), είτε "γεμίζουν" με 0. Εξαρτάται από τον compiler.

int a, c ;
a = 548884;
c = a >> 10;
// **c is 536**
	00000000 00001000 01100000 00010100
	>>
	10
	=
	00000000 00000000 00000010 00011000

Εφαρμογή **Συμπλήρωμα**

Δυαδική άρνηση (όπου υπάρχει 1 γίνεται 0 και **όπου υπάρχει 0 γίνεται 1** - one's complement). Εφαρμόζεται σε έναν μόνο τελεστέο.

int a, c ; a = 2032; c = ~a ; // **c is -2033**	00000000 00000000 00000111 11110000 ~ 11111111 11111111 11111000 00001111

Άσκηση αυτοαξιολόγησης - S3_LA7LO69

Γράψτε εντολές που αλλάζουν το πέμπτο bit από τα δεξιά της μεταβλητής ch (από 0 το κάνουν 1 και αντιστρόφως).

Άσκηση αυτοαξιολόγησης - S3_LA7LO70

unsigned char x = 178;

x = x & 074;

x = x | 0xD6;

x = x ^ 104;

x = x >> 3;

x = x << 2;

x = ~x;

Ποια θα είναι η τιμή της μεταβλητής x μετά από κάθε εντολή, λαμβάνοντας κάθε φορά υπ' όψιν το αποτέλεσμα της προηγούμενης εντολής;

Άσκηση αυτοαξιολόγησης - S3_LA7LO71

Ο τελεστής << έχει τον τύπο: variable << number και μετακινεί όλα τα bits της μεταβλητής προς τ' αριστερά number θέσεις. Οι κενές θέσεις που δημιουργούνται από τα δεξιά αντικαθίστανται με 0. Ποια τιμή θα έχει στο τέλος η μεταβλητή x;

x=5;

x=x << 2;

Παρατήρηση: Θα πρέπει να σκεφτείτε ποια είναι η αντίστοιχη δυαδική τιμή για τον αριθμό 5 και ποια μετατροπή θα γίνει με τον παραπάνω τελεστή.

Άσκηση αυτοαξιολόγησης - S3_LA7LO72

Συμπληρώστε τη στήλη 3 με το δυαδικό αριθμό που θα δώσουν οι αντίστοιχες εκφράσεις:

(10010011) & (00111101)	==	
(10010011) | (00111101)		
(10010011) ^ (00111101)		
(10001010) << 2		
(10001010) >> 2		
~(10011010)		

Άσκηση αυτοαξιολόγησης - S3_LA7LO73

Αν υποθέσουμε ότι η μεταβλητή val είναι του τύπου unsigned int με τιμή 2, τότε στο δυαδικό σύστημα αυτός ο αριθμός είναι ο 00000010. Να υπολογίσετε και να εκτυπώσετε το ~val σε πρόγραμμα.

3.3.2 Δραστηριότητα 2 - Μοναδιαίοι, δυαδικοί και τριαδικοί τελεστές

Τίτλος Δραστηριότητας	**S3_LA8** - Μοναδιαίοι, Δυαδικοί και Τριαδικοί τελεστές
Τίτλος Ενότητας	Τελεστές - Εκφράσεις - προτάσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει τους τελεστές της γλώσσας C ανάλογα με τον αριθμό των τελεστέων που δρουν και αναφέρονται με παραδείγματα ανά κατηγορία. Έτσι, περιγράφονται οι μοναδιαίοι, δυαδικοί και τριαδικοί τελεστές της γλώσσας C. Επειδή κάποιοι από αυτούς είναι γνωστοί από την προηγούμενη δραστηριότητα, δίνεται βάρος στους μοναδιαίους τελεστές αύξησης και μείωσης, στον τελεστή sizeof και στους τριαδικούς τελεστές.
Γλώσσα	Ελληνικά
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τουλάχιστον 3 παραδείγματα μοναδιαίων τελεστών της γλώσσας προγραμματισμού C • να αναφέρουν τουλάχιστον 3 παραδείγματα δυαδικών τελεστών της γλώσσας προγραμματισμού C • να αναφέρουν 1 παράδειγμα τριαδικών τελεστών της γλώσσας προγραμματισμού C • να αναφέρουν παραδείγματα μοναδιαίων τελεστών αύξησης και μείωσης
Μαθησιακά Αντικείμενα	Μοναδιαίοι, δυαδικοί τελεστές Μοναδιαίος τελεστής sizeof Μοναδιαίοι τελεστές αύξησης και μείωσης Τριαδικοί τελεστές
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 10 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S3_LA8LO74, S3_LA8LO77, S3_LA8LO79 και S3_LA8LO85.
Λέξεις Κλειδιά	τελεστές αύξησης και μείωσης, δυαδικοί τελεστές, τριαδικοί τελεστές

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA8LO74**
Τίτλος	**Μοναδιαίοι, δυαδικοί τελεστές**
Τίτλος δραστηριότητας	Μοναδιαίοι, Δυαδικοί και Τριαδικοί τελεστές
Περιγραφή	Στο συγκεκριμένο ΜΑ, οι τελεστές ταξινομούνται, ανάλογα με τον αριθμό των τελεστέων στους οποίους δρουν, σε μοναδιαίους (unary), δυαδικούς (binary) και τριαδικούς (ternary). Κυρίως αναφέρονται οι δυαδικοί τελεστές, τόσο οι αριθμητικοί όσο και οι συγκριτικοί, με παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Τελεστές, μοναδιαίοι (unary) τελεστές, δυαδικοί (binary) τελεστές
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τουλάχιστον 3 παραδείγματα μοναδιαίων τελεστών της γλώσσας προγραμματισμού C, • να αναφέρουν τουλάχιστον 3 παραδείγματα δυαδικών τελεστών της γλώσσας προγραμματισμού C, • να αναφέρουν τον τύπο του πηλίκου μιας διαίρεσης, ανάλογα με τον τύπο των τελεστέων.

Στη C υπάρχουν διάφοροι τελεστές που εκτελούν συγκεκριμένες αριθμητικές πράξεις:

Μοναδιαίοι (unary) +,-: Δρώντας σε ένα αριθμό a μας δίνουν τον ίδιο αριθμό ή τον αντίθετό του, αντίστοιχα.

Δυαδικοί (binary) +,-,*: Δρώντας μεταξύ δύο αριθμών a, b μας δίνουν το άθροισμα, τη διαφορά και το γινόμενό τους, αντίστοιχα.

Δυαδικός τελεστής / μεταξύ πραγματικών a, b: δίνει το λόγο a/b.

Δυαδικοί τελεστές / και % μεταξύ ακεραίων δίνουν το πηλίκο και το υπόλοιπο της ακέραιας διαίρεσης, αντίστοιχα.

Τριαδικός τελεστής (ternary)

Προσέξτε ότι δεν υπάρχει τελεστής για ύψωση σε δύναμη. Αντ' αυτού χρησιμοποιείται η συνάρτηση pow() που περιλαμβάνεται στον header <math.h>.

Οι δυαδικοί τελεστές συνοψίζονται στον παρακάτω πίνακα της εικόνας 6:

Δυαδικός τελεστής	Σύμβολο	Δυαδικός τελεστής	Σύμβολο
Μικρότερο	<	Πρόσθεση	+
Μικρότερο ή ίσο	<=	Αφαίρεση	-
Ίσο	==	Πολλαπλασιασμός	*
Διάφορο	!=	Διαίρεση πραγματικών	/
Μεγαλύτερο	>	Πηλίκο διαίρεσης ακεραίων	/
Μεγαλύτερο ή ίσο	>=	Υπόλοιπο διαίρεσης ακεραίων	%

Εικόνα 6. Δυαδικοί τελεστές

Άσκηση αυτοαξιολόγησης - S3_LA8LO75

Τι θα εκτυπώσει το παρακάτω πρόγραμμα; Γιατί δεν συμβαίνει το ίδιο στην γραμμή 10 και στην γραμμή 11 του προγράμματος; Δικαιολογήστε την άποψή σας.

```
1   #include <stdio.h>
2
3   main()
4   {
5       int a, b;
6       double x, y;
7
8       a = 2;
9       b = 3;
10      x = a/b;
11      y = 2.0/3;
12      printf("a=%d, b=%d, x=%g, y=%g\n", a, b, x, y);
13  }
```

Παρατήρηση - ανατροφοδότηση άσκησης

Το πρόβλημα εδώ είναι στη γραμμή 10, όπου το x παίρνει την τιμή 0 αντί για 0.66... Ο C compiler όταν βλέπει την έκφραση a/b, την κατανοεί ως το πηλίκο της διαίρεσης του ακεραίου a δια του ακεραίου b, άρα το πηλίκο της διαίρεσης 2/3 είναι 0, άρα ορθώς το x

παίρνει τελικά την τιμή 0. Γιατί δεν συμβαίνει το ίδιο και στη γραμμή 11, όπου βλέπουμε ότι το y παίρνει τη σωστή του τιμή τελικά; Ο λόγος είναι ότι η προς υπολογισμό έκφραση 2.0/3 έχει τον ένα από τους δύο συμμετέχοντες αριθμούς να μην είναι ακέραιος. Αυτό ακριβώς κάνει ο C compiler και συγκεκριμένα μετατρέπει και τον άλλο operand (το 3) στον αντίστοιχο δεκαδικό, και κάνει διαίρεση ανάμεσα σε δυο δεκαδικούς αριθμούς. Το αποτέλεσμα είναι 0.666 που τελικά εκχωρείται στο y.

Άσκηση αυτοαξιολόγησης - S3_LA8LO76

Έστω a=1001101, b=1110101, c=0101101

Να συμπληρωθούν οι παρακάτω εκφράσεις με δυαδικούς τελεστές διαχείρισης bits:

~ not ~a=

& and a & b=

| or a | b=

^ xor a ^ b=

<< shift left a<<2=

>> shift right a>>1=

c>>1=

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA8LO77**
Τίτλος	**Τελεστής sizeof**
Τίτλος δραστηριότητας	Μοναδιαίοι, Δυαδικοί και Τριαδικοί τελεστές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται ένας ειδικός τελεστής, ο οποίος χρησιμοποιείται για να υπολογίσει το μέγεθος κάποιας μεταβλητής ή σταθεράς.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, κείμενο
Λέξεις Κλειδιά	Τελεστής sizeof
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • Να αναφέρουν τουλάχιστον 3 παραδείγματα για τον τελεστή sizeof της γλώσσας προγραμματισμού C

Επειδή το εύρος των διαφόρων τύπων δεδομένων είναι δυνατόν να είναι διαφορετικό από compiler σε compiler, με τη χρήση του τελεστή sizeof (που μοιάζει με συνάρτηση, αλλά δεν είναι) μπορούμε να δούμε το εύρος κάθε τύπου δεδομένων στον compiler που χρησιμοποιείται. Η sizeof() χρησιμοποιείται για να υπολογίσει το μέγεθος κάποιας μεταβλητής ή σταθεράς, αλλά μπορεί να πάρει ως παραμέτρους και τύπους δεδομένων:

```
#include <stdio.h>
main()
{
    printf("\nSize of type char: %d bytes", sizeof(char));
    printf("\nSize of type int: %d bytes", sizeof(int));
    printf("\nSize of type short: %d bytes", sizeof(short));
    printf("\nSize of type long: %d bytes", sizeof(long));
    printf("\nSize of type unsigned char: %d bytes", sizeof(unsigned char));
    printf("\nSize of type unsigned int: %d bytes", sizeof(unsigned int));
    printf("\nSize of type float: %d bytes", sizeof(float));
    printf("\nSize of type double: %d bytes", sizeof(double));
}
```

Όπως βλέπουμε, με την εκτέλεση του παραπάνω προγράμματος μπορούμε να μάθουμε το μέγεθος κάθε τύπου δεδομένων στον compiler που χρησιμοποιούμε.

Το αποτέλεσμα του παραπάνω προγράμματος για ένα συγκεκριμένο υπολογιστή είναι:

Size of type char: 1 bytes

Size of type int: 4 bytes

Size of type short: 2 bytes

Size of type long: 4 bytes

Size of type unsigned char: 1 bytes

Size of type unsigned int: 4 bytes

Size of type float: 4 bytes

Size of type double: 8 bytes

Παράδειγμα εκμάθησης της συνάρτησης sizeof

```
#include
main() {
    char c;
    short s;
    int i;
    double d;
    int a[10];
    printf("%d\n", sizeof(c));
    printf("%d\n", sizeof(s));
    printf("%d\n", sizeof(i));
    printf("%d\n", sizeof(d));
    printf("%d\n", sizeof(a));
    printf("%d\n", sizeof(char));
} /* end of main */
```

(τυπώνει 1, 2, 4, 4, 40, 1 σε υπολογιστές Pentium)

Παρατήρηση: η πρόταση int a[10]; δηλώνει ένα μονοδιάστατο πίνακα 10 θέσεων για ακεραίους.

Άσκηση αυτοαξιολόγησης - S3_LA8LO78

Να εκτελέσετε ένα από τα παραπάνω προγράμματα και να δείτε το εύρος κάθε τύπου δεδομένων στον compiler που χρησιμοποιείτε.

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA8LO79**
Τίτλος	**Μοναδιαίοι τελεστές αύξησης και μείωσης**
Τίτλος δραστηριότητας	Μοναδιαίοι, Δυαδικοί και Τριαδικοί τελεστές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι μοναδιαίοι τελεστές αύξησης και μείωσης στη γλώσσα C. Οι τελεστές αυτοί έχουν υψηλή προτεραιότητα και χρησιμοποιούνται κυρίως γιατί εκτελούνται πιο γρήγορα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Τελεστής αύξησης ++, τελεστής μείωσης --
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τουλάχιστον 3 παραδείγματα τελεστών αύξησης της γλώσσας προγραμματισμού C • να αναφέρουν τουλάχιστον 3 παραδείγματα τελεστών μείωσης της γλώσσας προγραμματισμού C • να αναφέρουν τη διαφορά της εντολής ++i με την i++, όπου i μια μεταβλητή τύπου ακεραίου

Οι αριθμητικοί τελεστές της C συμβολίζονται όπως και στις άλλες γλώσσες προγραμματισμού, αλλά η C έχει επιπλέον και τους εξής:

```
i++     /* αυξάνει το i κατά ένα μετά τη χρήση του i */
++i     /* αυξάνει το i κατά ένα πριν τη χρήση του i */
i--     /* μειώνει το i κατά ένα μετά τη χρήση του i */
--i     /* μειώνει το i κατά ένα πριν τη χρήση του i */
```

Δηλαδή, το i++ **σημαίνει:**

"κάνε χρήση του i **και μετά αύξησέ το κατά 1"**

και το ++i **σημαίνει:**

"αύξησε το i **κατά 1 και μετά χρησιμοποίησέ το"**

Οι τελεστές αύξησης και μείωσης έχουν υψηλή προτεραιότητα και μόνο οι παρενθέσεις έχουν υψηλότερη. Πρέπει πάντως να αποκτήσει κανείς αρκετή εμπειρία με τους τελεστές αύξησης και μείωσης, να μάθει τις ιδιαιτερότητες της έκδοσης της C με την οποία δουλεύει και μετά να τους χρησιμοποιεί.

Πρέπει να έχουμε υπόψη μας τα εξής:

Δεν πρέπει να χρησιμοποιούμε τελεστές αύξησης ή μείωσης σε μεταβλητή που αποτελεί μέρος περισσότερων του ενός ορισμάτων ή συναρτήσεων.

Δεν πρέπει να χρησιμοποιούμε τελεστές αύξησης ή μείωσης σε μεταβλητή που εμφανίζεται περισσότερες από μία φορές σε μια έκφραση.

Δηλαδή, οι παρακάτω εκφράσεις μπορεί να οδηγήσουν σε απρόβλεπτα αποτελέσματα:

```
printf("%10d %10d\n", num, num*num++);
a = num/2 + 5*(1 + num++);
b = n++ + n++;
```

Τέλος, οι συνδυασμένοι τελεστές, όπως οι παραπάνω, χρησιμοποιούνται και ως απλές πράξεις, ενώ, στο πρότυπο της ANSI C μας γλυτώνουν 1 επεξεργαστικό κύκλο (Hertz), το οποίο σημαίνει πως εκτελούνται πιο γρήγορα.

Άσκηση αυτοαξιολόγησης - S3_LA8LO80

Να φτιάξετε πρόγραμμα που θα αρχικοποιεί τις μεταβλητές a και b με την τιμή 1 και θα εμφανίζει το αποτέλεσμα:

```
a       aplus     b      plusb
2         1       2        2
```

Όπου aplus η τιμή της a πριν την αύξησή της και plusb η τιμή της b μετά την αύξησή της.

Άσκηση αυτοαξιολόγησης - S3_LA8LO81

Να συμπληρώσετε την τρίτη και την τέταρτη στήλη

Υποθέστε: int c = 3, d = 5, e = 4, f = 6, g = 12;

Τελεστής εκχώρησης	**Δείγμα Παράστασης**	**Εξήγηση**	**Εκχωρεί το**
+=	c += 7		
-=	d -= 4		
*=	e *= 5		
/=	f /= 3		
%=	g %= 9		

Άσκηση αυτοαξιολόγησης - S3_LA8LO82

Τι θα τυπωθεί στην οθόνη όταν εκτελεστεί ο παρακάτω κώδικας; Στη συνέχεια γράψτε ένα πρόγραμμα για να ελέγξετε τις απαντήσεις σας. Να δώσετε κατά σειρά τους τρεις ακεραίους που δίνουν οι 4 εντολές printf.

```
int x, y = 6, z = 8;
x = y++ + ++z;
printf (“%d %d %d/n’’, x, y, z);
x = ++y + z++;
printf (“%d %d %d/n’’, x, y, z);
x = ++y + ++z;
printf (“%d %d %d/n’’, x, y, z);
x = --y + z--;
printf (“%d %d %d/n’’, x, y, z);
```

Άσκηση αυτοαξιολόγησης - S3_LA8LO83

Έχει σημασία αν οι τελεστές αύξησης/μείωσης τεθούν πριν (++i) ή μετά (i--) τη μεταβλητή. Στο πρόγραμμα που ακολουθεί ποιο αποτέλεσμα θα πάρουμε από την εντολή printf;

```
#include <stdio.h>
main()
{
    int a=3, b=3;
    int syna, synb;
    syna=a++;
    synb=++b;
    printf ("a= %d, syna= %d, b= %d, synb=%d\n", a, syna, b, synb);
}
```

Άσκηση αυτοαξιολόγησης - S3_LA8LO84

Να συμπληρώσετε τις τιμές των x και y:

πρόγραμμα	x	y
int x=10;		
int y=20;		
++x;		
y=--x;		
y+=x--;		
y-=x++;		

πρόγραμμα	x	y
int x, y;	x	y
int x=1;		
y=x++;		
y=++x;		
y=x--;		
y=--x;		
y=y + x--;		
y=y+ ++x;		

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA8LO85**
Τίτλος	**Τριαδικός τελεστής**
Τίτλος δραστηριότητας	Μοναδιαίοι, Δυαδικοί και Τριαδικοί τελεστές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται ο υποθετικός τελεστής της C που ανήκει στην κατηγορία των τριαδικών τελεστών. Ο υποθετικός τελεστής επιλέγει μία από δύο δυνατές τιμές ανάλογα με την τιμή μιας συνθήκης. Π.χ. συνθήκη ? τιμή1 : τιμή2 Εάν η συνθήκη είναι αληθής τότε το αποτέλεσμα είναι η τιμή1, διαφορετικά το αποτέλεσμα είναι η τιμή2.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	τριαδικός τελεστής, υποθετικός τελεστής
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τουλάχιστον 3 παραδείγματα για τον υποθετικό τελεστή της γλώσσας προγραμματισμού C

Ο υποθετικός τελεστής αποτελείται από δύο σύμβολα (? και :), ανήκει στην κατηγορία των τελεστών που αποτελούνται από συνδυασμό συμβόλων και δεν ακολουθούν καμία από τις postfix, prefix ή infix σημειολογίες. Όταν τα σύμβολα ή οι λέξεις του τελεστή είναι διάσπαρτα στους τελεστέους στους οποίους εφαρμόζεται ο τελεστής, λέμε ότι ο τελεστής είναι σε μικτή σημειολογία (mixfix notation). Για παράδειγμα, στην έκφραση x > z ? x : z τα σύμβολα ? και : του υποθετικού τελεστή παρεμβάλλονται μεταξύ των x > z, x και z. Η έκφραση που σχηματίζει ο υποθετικός τελεστής έχει τη μορφή:

εκφρ1 ? εκφρ2 : εκφρ3

Η τιμή της συνολικής έκφρασης είναι η τιμή εκφρ2, εάν η εκφρ1 είναι αληθής, αλλιώς είναι η τιμή εκφρ3. Η εκφρ1 που αποτελεί τη συνθήκη ελέγχου πρέπει να είναι βαθμωτού τύπου. Μόνο μία από τις δύο εκφράσεις υπολογίζεται ανάλογα με την τιμή της εκφρ1.

Έτσι, η έκφραση x>z ? x : z έχει τιμή x εάν το x > z είναι αληθές, διαφορετικά έχει τιμή z.

Παράδειγμα:

```
int a,b;
scanf("%d",&a);
scanf("%d",&b);
printf("Ο μεγαλύτερος αριθμός είναι: %d\n",a>b?a:b);
```

Άσκηση αυτοαξιολόγησης - S3_LA8LO86

Να γράψετε πρόγραμμα που θα εκχωρεί σε μια μεταβλητή max τον μεγαλύτερο από δύο ακεραίους χρησιμοποιώντας τον τριαδικό τελεστή που αναφέραμε παραπάνω.

Άσκηση αυτοαξιολόγησης - S3_LA8LO87

Τροποποιήστε το παραπάνω πρόγραμμα ώστε να τυπώνει τον μικρότερο. Τέλος, δώστε μια έκδοση για τρεις αριθμούς.

3.3.3 Δραστηριότητα 3 - Εκφράσεις και Προτάσεις

Τίτλος Δραστηριότητας	**S3_LA9** - Εκφράσεις και Προτάσεις
Τίτλος Ενότητας	Τελεστές - Εκφράσεις - Προτάσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει τις εκφράσεις στη γλώσσα C, δηλαδή συνδυασμούς τελεστών με μεταβλητές ή σταθερές, καθώς και τους κανόνες που ισχύουν για τον υπολογισμό τους. Επίσης αναφέρονται τα είδη των προτάσεων με χαρακτηριστικά παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν παραδείγματα απλών και σύνθετων εκφράσεων της γλώσσας προγραμματισμού C • να αναφέρουν παραδείγματα εκφράσεων της γλώσσας προγραμματισμού C, ο υπολογισμός των οποίων επηρεάζεται από την προτεραιότητα των τελεστών • να αναφέρουν παραδείγματα αριστερής και δεξιάς προσεταιριστικότητας • να εξηγούν τη σημασία των επιπέδων προτεραιότητας των τελεστών • να εξηγούν με παραδείγματα τη σημασία της προσεταιριστικότητας των τελεστών
Μαθησιακά Αντικείμενα	Εκφράσεις Προτεραιότητα και προσεταιριστικότητα τελεστών Προτάσεις Μετατροπές τύπων
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 8 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S3_LA9LO88, S3_LA9LO90, S3_LA9LO93 και S3_LA9LO95.
Λέξεις Κλειδιά	Απλές σύνθετες εκφράσεις, προτάσεις

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA9LO88**
Τίτλος	**Εκφράσεις**
Τίτλος δραστηριότητας	Εκφράσεις και Προτάσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται σύνολα τελεστέων και τελεστών, κανόνες υπολογισμού διαφορετικών εκφράσεων με παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο
Λέξεις Κλειδιά	Απλές και σύνθετες εκφράσεις, ένθετος τελεστής, προθεματικός, επιθεματικός τελεστής
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν παραδείγματα απλών και σύνθετων εκφράσεων της γλώσσας προγραμματισμού C • να αναφέρουν παραδείγματα ένθετων, προθεματικών και επιθεματικών τελεστών

Μία έκφραση, στη γενική της περίπτωση, αποτελείται από έναν ή περισσότερους τελεστέους και, προαιρετικά, από ένα ή περισσότερους τελεστές. Οι τελεστέοι, μεταβλητές, σταθερές και κλήσεις συναρτήσεων αποτελούν από μόνοι τους εκφράσεις, αλλά μπορούν να συνδυαστούν με τους τελεστές για να σχηματίσουν σύνθετες εκφράσεις. Ο υπολογισμός της τιμής κάθε έκφρασης γίνεται σύμφωνα με κανόνες, ορισμένοι από τους οποίους είναι απλοί και ευνόητοι, όπως οι κανόνες που εφαρμόζονται για τον υπολογισμό των παρακάτω απλών και σύνθετων εκφράσεων, και άλλοι είναι περισσότερο σύνθετοι και θέλουν ιδιαίτερη αναφορά.

Απλές εκφράσεις	Σύνθετες εκφράσεις
8	12 * 20
count	count + 1
func()	num / (count + 1)
MAX_VALUE	func() / 4

Οι γλώσσες προγραμματισμού χρησιμοποιούν διάφορους συμβολισμούς για το σχηματισμό εκφράσεων, με πιο συνηθισμένο το συμβολισμό που τοποθετεί τον τελεστή μεταξύ των τελεστέων. Ειδικότερα, ένας δυαδικός τελεστής μπορεί να τοποθετηθεί:

μεταξύ των δεδομένων στα οποία ενεργεί, όπως στην έκφραση x + y, οπότε έχουμε τη σημειολογία ένθεσης ή **ένθετου τελεστή** (infix notation),

πριν από τους τελεστέους όπως στην έκφραση +x y, οπότε έχουμε τη σημειολογία πρόθεσης ή **προπορευόμενου τελεστή** (prefix notation) και

μετά από τους τελεστέους όπως στην έκφραση x y +, οπότε έχουμε τη σημειολογία επίθεσης ή **παρελκόμενου τελεστή** (postfix notation).

Σε κάθε περίπτωση, όμως, η τιμή που προκύπτει από την εφαρμογή της διεργασίας που αναπαριστά ο τελεστής είναι η ίδια. Η τιμή αυτή δεν αλλάζει και στην περίπτωση που η έκφραση κλειστεί σε παρενθέσεις. Είναι προφανές ότι η έκφραση x + y έχει την ίδια τιμή με την έκφραση (x + y).

Η σημειολογία του ένθετου τελεστή παράγει κατανοητές και εύκολες στην ανάγνωση εκφράσεις, ενώ αυτή του παρελκόμενου τελεστή έχει το πλεονέκτημα του ευκολότερου μηχανικού υπολογισμού με τη χρήση της έννοιας της στοίβας,

Οι εκφράσεις χωρίζονται σε:

σταθερές εκφράσεις (μόνο σταθερές τιμές),

ακέραιου τύπου ή τύπου κινητής υποδιαστολής ή μικτές,

εκφράσεις δείκτη (τιμές διεύθυνσης) δηλ. με δείκτες, τελεστή διεύθυνσης &, αλφαριθμητικά και πίνακες.

Άσκηση αυτοαξιολόγησης- S3_LA9LO89

Να συμπληρώσετε τη στήλη 2, λαμβάνοντας σε κάθε έκφραση υπ' όψη τις τιμές που έχουν δοθεί σε προηγούμενες εκφράσεις.

α/α	Έκφραση	Τιμή
1	4	
2	4 + 21	
3	q = 5*2	
4	q > 3	
5	x = ++q % 3	
6	c = 3 + 8	
7	6 + (c = 3 + 8)	
8	q * (x + c/11)/20	
9	5>3	

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA9LO90**
Τίτλος	**Προτεραιότητα και προσεταιριστικότητα τελεστών**
Τίτλος δραστηριότητας	**Εκφράσεις και Προτάσεις**
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται η έννοια της προτεραιότητας των τελεστών, δηλαδή ποια πράξη εκτελείται πρώτη, καθώς και η έννοια της προσεταιριστικότητας, δηλαδή ποιος είναι ο προσδιορισμός της κατεύθυνσης εφαρμογής ενός τελεστή όταν έχουμε περισσότερους ίδιους τελεστές στο ίδιο επίπεδο.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Προτεραιότητα τελεστών
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν παραδείγματα εκφράσεων της γλώσσας προγραμματισμού C, ο υπολογισμός των οποίων επηρεάζεται από την προτεραιότητα των τελεστών • να αναφέρουν παραδείγματα αριστερής και δεξιάς προσεταιριστικότητας • να εξηγούν τη σημασία των επιπέδων προτεραιότητας των τελεστών • να εξηγούν με παραδείγματα τη σημασία της προσεταιριστικότητας των τελεστών

Για τον υπολογισμό της τιμής μιας έκφρασης, ουσιαστικής σημασίας είναι οι έννοιες της προτεραιότητας (precedence) και της προσεταιριστικότητας (associativity), οι οποίες επηρεάζουν τον τρόπο με τον οποίο οι τελεστέοι προσδένονται στους τελεστές.

Η ύπαρξη περισσότερων τελεστών στο ίδιο επίπεδο επιβάλει τον προσδιορισμό της κατεύθυνσης εφαρμογής, με την από αριστερά προς τα δεξιά κατεύθυνση να είναι ευρύτερα χρησιμοποιούμενη. Ένας τελεστής, λέμε πως είναι αριστερής προσεταιριστικότητας (left associative), όταν σε εκφράσεις που περιέχουν πολλά στιγμιότυπα του τελεστή, η εφαρμογή γίνεται από αριστερά προς τα δεξιά. Έτσι, η έκφραση 10 - 8 -2 υπολογίζεται σαν (10 - 8) - 2. Οι τελεστές +, -, * και / είναι όλοι αριστερής προσεταιριστικότητας.

Παράδειγμα τελεστή δεξιάς προσεταιριστικότητας αποτελεί, για την C, η ύψωση σε δύναμη καθώς και ο τελεστής ανάθεσης (=). Έτσι, στην έκφραση num1 = num2 = 10

εφαρμόζεται πρώτα ο δεξιός τελεστής ανάθεσης, με αποτέλεσμα πρώτα να ανατεθεί η τιμή 10 στη μεταβλητή num2 και μετά η ίδια τιμή στη μεταβλητή num1.

Προτεραιότητα	**Τελεστές**
Υψηλότερη	( ) [ ] ->
	! ~ ++ -- -(type) * & sizeof
	* / %
	- +
	<< >>
	< <= > >=
	== !=
	&
	^
	|
	&&
	||
	?
Χαμηλότερη	= += -= *= /=

Ο παρακάτω πίνακας της εικόνας 7 παρουσιάζει την προτεραιότητα αλλά και την προσεταιριστικότητα των τελεστών:

Α/Α	**Όνομα**	**Τελεστής**	**Σειρά υπολογισμού**
1	Επιλογή μέλους, δείκτης, κλήση συναρτήσεων, αύξηση και μείωση κατάληξης	. -> () ++ --	Αριστερά στα Δεξιά
2	Αύξηση και μείωση προθέματος, συμπληρωματικό, not, μοναδιαίο μείον και συν, and, διεύθυνση και εμφάνιση διεύθυνσης, μετατροπή, μέγεθος μνήμης	++ -- ^ ! - + & * () sizeof()	Δεξιά στα Αριστερά
3	Επιλογή μέλους για δείκτη	. * -> *	Αριστερά στα Δεξιά
4	Πολλαπλασιασμός, διαίρεση, υπόλοιπο	* / %	Αριστερά στα Δεξιά
5	Πρόσθεση, αφαίρεση	+ -	Αριστερά στα Δεξιά
6	Μετατόπιση	<< >>	Αριστερά στα Δεξιά
7	Ισότητα, ανισότητα	== !=	Αριστερά στα Δεξιά
8	AND σε επίπεδο bit	&	Αριστερά στα Δεξιά
9	Αποκλειστικό OR σε επίπεδο bit	^	Αριστερά στα Δεξιά
10	OR σε επίπεδο bit	|	Αριστερά στα Δεξιά

Α/Α	Όνομα	Τελεστής	Σειρά υπολογισμού
11	Λογικό AND	&&	Αριστερά στα Δεξιά
12	Λογικό OR	||	Αριστερά στα Δεξιά
13	Υποθετικός τελεστής	? :	Δεξιά στα Αριστερά
14	Τελεστές δήλωσης	= += -= *= /= %= <<= >>= &= |= ^=	Δεξιά στα Αριστερά

Εικόνα 7. Προτεραιότητα και προσεταιριστικότητα τελεστών

Άσκηση αυτοαξιολόγησης - S3_LA9LO91

Εφαρμόστε τους κανόνες προτεραιότητας και προσεταιριστικότητας των τελεστών για να υπολογίσετε την τιμή της έκφρασης 2 + ((4 + 2)/(7 - 5) - 6).

Άσκηση αυτοαξιολόγησης - S3_LA9LO92

Να γίνει η αντιστοίχιση της δεξιάς στήλης με τις τιμές της αριστερής στήλης.

Έκφραση	Τιμή
-4 + 6	true (1)
c=3+8	17
χ	2
sqrt(16.0)	η τιμή της x
1821	false
5>3	4.0
8<1	11
6 + (c=3+8)	false (0)
(5> 1) && (6>7)	1821

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA9LO93**
Τίτλος	**Προτάσεις**
Τίτλος δραστηριότητας	Εκφράσεις και Προτάσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι προτάσεις στη γλώσσα C, δηλαδή πλήρεις εντολές (commands) προς τον υπολογιστή που προσδιορίζουν την εκτέλεση συγκεκριμένων λειτουργιών. Αναφέρονται οι κατηγορίες προτάσεων με χαρακτηριστικά παραδείγματα.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Πρόταση δήλωσης, ανάθεσης, ελέγχου ροής, κλήσης συνάρτησης
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν τη διαφορά ανάμεσα σε μια έκφραση και μια πρόταση στη γλώσσα C • να αναφέρουν τουλάχιστον τρία παραδείγματα προτάσεων ανά κατηγορία • να εξηγούν τη σύνθετη πρόταση μέσα σε αγκύλες δίνοντας παραδείγματα • να αναφέρουν παραδείγματα προτάσεων του προεπεξεργαστή

Οι προτάσεις είναι τα πρωταρχικά δομικά στοιχεία ενός προγράμματος, δηλ. το πρόγραμμα είναι μια σειρά προτάσεων ή αλλιώς εντολών. Στη C οι προτάσεις κλείνουν με το σύμβολο ';' Έτσι, η a = 4 είναι μια έκφραση, αλλά η a= 4; είναι μια πρόταση. Στο παράδειγμα: x = 6 + (y=5); η υποέκφραση y=5 είναι μια ολοκληρωμένη εντολή, αλλά απλά είναι μέρος μιας πρότασης. Επειδή συνεπώς μια ολοκληρωμένη εντολή δεν είναι και απαραίτητα μια πρόταση, το σύμβολο (;) χρειάζεται για να χαρακτηρίζει τις εντολές που είναι αληθινές προτάσεις.

Στο επόμενο παράδειγμα χρησιμοποιούμε τέσσερα είδη προτάσεων:

```
#include < stdio.h>
main() /* βρίσκει το άθροισμα των 20 πρώτων ακεραίων */
{
    int count, sum;                    /* πρόταση δήλωσης */
    count = 0;                         /* πρόταση καταχώρησης */
    sum = 0;                           /* πρόταση καταχώρησης */
    while (count++ < 20)               /* πρόταση ελέγχου while */
        sum = sum + count;             /* πρόταση */
    printf("άθροισμα = %d\n", sum);    /* πρόταση - συνάρτηση */
}
```

Μια σύνθετη πρόταση αποτελείται από δύο ή περισσότερες προτάσεις που περικλείονται από αγκύλες. Λέγεται επίσης και μπλοκ (block).

Ακολουθούν παραδείγματα:

```
/* τμήμα προγράμματος 1 */
index = 0;
while (index++ < 10)
    sum = 10*index + 2;
printf("sum = %d\n", sum);

/* τμήμα προγράμματος 2 */
index = 0;
while (index++ < 10)
{
    sum = 10*index + 2;
    printf("sum = %d\n", sum);
}
```

Στο τμήμα προγράμματος 1, ο βρόχος while περιλαμβάνει μόνο μια πρόταση αντικατάστασης, δηλ. όταν λείπουν τα { και }, μια πρόταση while τρέχει από το while μέχρι το επόμενο σύμβολο ;.

Στο τμήμα προγράμματος 2, τα σύμβολα { και } δηλώνουν ότι και οι δύο προτάσεις αποτελούν μέρος του βρόχου while. Ολόκληρη η σύνθετη πρόταση θεωρείται σαν μια απλή πρόταση με την έννοια της δομής της πρότασης while.

Μια ειδική κατηγορία προτάσεων που θα συναντήσουμε σε C προγράμματα, είναι οι προτάσεις του **προεπεξεργαστή**. Ο προεπεξεργαστής της C είναι ένα πρόγραμμα που επεξεργάζεται τον πηγαίο κώδικα πριν από τον μεταγλωττιστή. Αναγνωρίζει ορισμένες εντολές και προκαλεί ένα σύνολο από τροποποιήσεις στον πηγαίο κώδικα πριν αρχίσει το

έργο της μεταγλώττισης. Στην κατηγορία αυτή ανήκει η πρόταση συμπερίληψης που ήδη συναντήσαμε και έχει την μορφή:

#include <προσδιορισμός αρχείου>

Η πρόταση δίνει εντολή στον προεπεξεργαστή να εισάγει στη θέση της, το αρχείο το οποίο προσδιορίζει η πρόταση.

Οι προτάσεις ανάλογα με το έργο που επιτελούν διακρίνονται σε κατηγορίες, οι σημαντικότερες των οποίων δίνονται στον πίνακα με αντίστοιχα παραδείγματα.

Κατηγορία πρότασης	**Παράδειγμα**
δήλωσης	int num;
κλήσης συνάρτησης	printf(“Hello World”);
ελέγχου ροής	if (a>b) then a; else b;
Ανάθεσης - καταχώρησης	num = 21;
μηδενική	;

Άσκηση αυτοαξιολόγησης - S3_LA9LO94

Μετατρέψτε τις παρακάτω μαθηματικές εκφράσεις στη γλώσσα C.

Μαθηματική έκφραση	**Έκφραση στη γλώσσα C**
m2-n2	
ax2+ bx+ c	
-b + 4ac	
2ab/(c+d)	
((-a)b)+((-c)d)	

Μαθησιακό Αντικείμενο	
Όνομα	**S3_LA9LO95**
Τίτλος	**Οι Μετατροπές Τύπων δεδομένων**
Τίτλος δραστηριότητας	Εκφράσεις και Προτάσεις
Περιγραφή	Το συγκεκριμένο ΜΑ αναφέρεται στο πολύ σημαντικό θέμα της μετατροπής τύπων και τον τρόπο που ο μηχανισμός μετατροπής τύπων διευκολύνει τη δημιουργία εκφράσεων από διαφορετικού τύπου τελεστέους. Η C χρησιμοποιεί ορισμένους κανόνες για να μετατρέψει αυτόματα διαφορετικούς τύπους δεδομένων στην ίδια έκφραση, αλλά μπορεί επίσης να δεχθεί μετατροπές που θα κάνει ο προγραμματιστής σε κάποια έκφραση.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Κείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	long double, double, float, unsigned long, long, unsigned int, int
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρουν την ταξινόμηση των τύπων της γλώσσας ανάλογα με το μέγεθος της μνήμης που απαιτούν για αποθήκευση • να αναφέρουν παραδείγματα υπονοούμενης μετατροπής τύπων • να αναφέρουν παραδείγματα ρητής μετατροπής τύπων • να εξηγούν τους κανόνες με τους οποίους η C μετατρέπει αυτόματα τους τύπους δεδομένων

Η C είναι, σε σχέση με τις άλλες γλώσσες προγραμματισμού, προετοιμασμένη, όταν ανακατεύουμε μεταβλητές και σταθερές διαφορετικών τύπων δεδομένων.

Οι μετατροπές τύπων μπορεί να είναι είτε **υπονοούμενες** (implicit conversions), οπότε εκτελούνται αυτόματα από το σύστημα, είτε να προσδιορίζονται **ρητά** από τον προγραμματιστή (explicit conversions).

Υπονοούμενη μετατροπή

Έστω ότι έχουμε την έκφραση: **sum=3.0+1/2;**

Παρατηρούμε ότι στην ίδια έκφραση έχουμε δυο τύπους δεδομένων δηλαδή int και float. Τι τιμή τελικά θα έχει η μεταβλητή sum; Θα γίνει αυτόματη μετατροπή από int σε float. Η μετατροπή αυτή ονομάζεται **υπονοούμενη μετατροπή.**

Η C βασίζει τη διαδικασία της αυτόματης μετατροπής στον κανόνα ο οποίος ορίζει ότι **ο στενότερος τύπος μετατρέπεται στον ευρύτερο** χωρίς να υπάρχει απώλεια πληροφορίας.

Ο κανόνας αυτός βασίζεται στο γεγονός ότι όλοι οι τύποι της γλώσσας ταξινομούνται ανάλογα με το μέγεθος της μνήμης που απαιτούν για αποθήκευση (char < int < long < float < double).

Παράδειγμα: Αν int i; και float f; η έκφραση f = i = 3.3; θα αποδώσει τιμή 3 στην μεταβλητή i και στη συνέχεια 3.0 στην μεταβλητή f.

Παράδειγμα: 3.0 + 1.0/2 = 3.5

Ρητές μετατροπές (casting)

Εκτός των παραπάνω υπονοούμενων μετατροπών, η C μας επιτρέπει να κάνουμε ρητές μετατροπές τιμών ενός τύπου σε ένα διαφορετικό τύπο. Η ρητή αυτή μετατροπή είναι γνωστή σαν **προσαρμογή** ή **casting** και εκτελείται με μια κατασκευή γνωστή με το όνομα cast. Για να μετατρέψετε τον τύπο μίας έκφρασης, τοποθετήστε τον τύπο προορισμού, περικλείοντάς τον μέσα σε παρενθέσεις, μπροστά από την έκφραση. Για παράδειγμα, στην έκφραση j =(float)2, μετατρέπεται ο ακέραιος 2 σε float πριν εκχωρηθεί στη μεταβλητή j.

ΠΡΟΣΟΧΗ: Αν η j είναι ακέραιος, ο μεταγλωττιστής αυτόματα θα μετατρέψει τον float σε int και μετά θα τον εκχωρήσει στην j.

Παράδειγμα 1

Ποιο το αποτέλεσμα των παρακάτω εκφράσεων, όταν η μεταβλητή res1 έχει δηλωθεί ως float:

Α. res1 = (float)(2.56 + 3/2);

Β. res1 = 2.56 + (float)(3/2);

Γ. res1 = 2.56 + (float)3/2;

Τότε το αποτέλεσμα θα είναι:

Α. res1 = 3.56;

Β. res1 = 3.56;

Γ. res1 = 4.06;

Παράδειγμα 2

```
a = (int) 3.2 + (int) 4.8        /* το αποτέλεσμα είναι 3 + 4 = 7 */
b = (float) 3 / 1                /* το αποτέλεσμα είναι 3.0 */
```

Συνοψίζοντας, η C χρησιμοποιεί ορισμένους κανόνες για να μετατρέψει αυτόματα τους τύπους:

1. Όταν εμφανίζονται σε εκφράσεις, τόσο ο τύπος char όσο και ο τύπος short, με πρόσημο ή χωρίς, αυτόματα μετατρέπονται σε τύπο int. Επειδή αυτές είναι μετατροπές προς κάποιο μεγαλύτερο τύπο, λέγονται **προαγωγές**.

2. Σε κάθε πράξη όπου εμπλέκονται δύο τύποι, οι δύο τιμές μετατρέπονται στον τύπο αυτής με τον "υψηλότερο" βαθμό.

3. Η ιεραρχία των τύπων από τους υψηλότερους προς τους χαμηλότερους είναι η εξής: long double, double, float, unsigned long, long, unsigned int και int.

4. Σε μια πρόταση καταχώρησης, το τελικό αποτέλεσμα των υπολογισμών μετατρέπεται στον τύπο της μεταβλητής στην οποία καταχωρήθηκε η τιμή. Έτσι, όμως, μπορεί μια τιμή να μετατραπεί σε τύπο χαμηλότερου βαθμού, όταν π.χ. καταχωρούμε τύπο float σε τύπο int και γίνεται, όπως είδαμε, στρογγυλοποίηση του αριθμού.

Παράδειγμα 1

Κατά την εκχώρηση η τιμή της παράστασης μετατρέπεται στον τύπο της μεταβλητής στο αριστερό μέρος της παράστασης. Για παράδειγμα:

```
int i;

char ch;

float f;


void function(void)
{
    ch=i; /* ch= ο χαρακτήρας που αντιστοιχεί στο δεύτερο byte του i */
    i=f; /* i= το ακέραιο μέρος του f */
    f=ch; /* f= ο αριθμός που αντιστοιχεί στο ένα byte του ch */
    f=i; /* f= ο αριθμός που αντιστοιχεί στα δύο bytes του int */
}
```

Παράδειγμα 2

Μετασχηματισμός του τύπου μιας παράστασης μπορεί να γίνει προσδιορίζοντας τον νέο τύπο μέσα σε παρένθεση πριν από την παράσταση. π.χ.

(τύπος)<παράσταση>

Για παράδειγμα

```
int i;

(float) i/2;
```

Άσκηση αυτοαξιολόγησης - S3_LA9LO96

Υπολογίστε τις τιμές των εκφράσεων 2.56 + 3/2 και 2.56 + 3.0/2. Περιγράψτε τις μετατροπές που λαμβάνουν χώρα για τον υπολογισμό της κάθε τιμής. Στη συνέχεια, αναπτύξτε ένα πρόγραμμα που θα τις υπολογίζει και θα τις τυπώνει στην οθόνη.

Άσκηση αυτοαξιολόγησης - S3_LA9LO97

Θεωρήστε την πρόταση res1 = 2.56 + 3/2; Ποια από τις παρακάτω μετατροπές δίνει το ίδιο αποτέλεσμα με αυτό της res2 = 2.56 + 3.0 / 2;

A. res1 = (float)(2.56 + 3/2);

B. res1 = 2.56 + (float)(3/2);

Γ. res1 = 2.56 + (float)3/2;

Άσκηση αυτοαξιολόγησης - S3_LA9LO98

Να γράψετε τα αποτελέσματα του παρακάτω προγράμματος:

```
/* prog20.c - αυτόματη μετατροπή τύπων */
#include <stdio.h>
main()
{
    char ch;
    int i;
    float fl;
    fl = i = ch = ‘A’;
    printf("ch = %c, i = %d, fl = %2.2f \n", ch, i, fl);
    ch = ch + 1;
    i = fl + 2 * ch;
    fl = 2.0 * ch + i;
    printf("ch = %c, i = %d, fl = %2.2f \n", ch, i, fl);
}
```

Άσκηση αυτοαξιολόγησης - S3_LA9LO99

Έστω το παρακάτω κομμάτι κώδικα. Ποια θα είναι η τιμή της μεταβλητής f1, f2 και f3:

```
int i, j;
float f1,f2,f3;
i=5; j=2;
f1= i/j + 0.5; /* Αποτέλεσμα: …*/
f2 = (float) i / (float) j + 0.5; /* Αποτέλεσμα: … */
f3 = i/j + 0.5; /* Αποτέλεσμα: …. */
```

3.4 Ενότητα 4 - Πίνακες Δείκτες

Τίτλος Ενότητας	**S4 -** Πίνακες Δείκτες
Τίτλος Μαθήματος	Βασικές αρχές προγραμματισμού με τη γλώσσα προγραμματισμού ANSI-C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστούν οι βασικές έννοιες των πινάκων και των δεικτών στη C, καθώς και ο τρόπος με τον οποίο οι δύο αυτοί τύποι χρησιμοποιούνται για τη δήλωση μεταβλητών. Ιδιαίτερη έμφαση δίνεται στο χειρισμό των αλφαριθμητικών, ως πινάκων χαρακτήρων.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να επιλέγουν το είδος του πίνακα • να ορίζουν τους πίνακες σε ένα πρόγραμμα • να εισάγουν, να επεξεργάζονται και να τυπώνουν τα στοιχεία ενός πίνακα • να αποφασίζουν αν είναι απαραίτητη η χρήση πίνακα • να αναφέρουν τις βασικές επεξεργασίες σε ένα πίνακα
Μαθησιακά Αποτελέσματα **(ΜΑπ)**	Όταν θα έχουν μελετήσει οι εκπαιδευόμενοι την ενότητα αυτή θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση του τύπου του πίνακα, • να δίνουν παραδείγματα δήλωσης πινάκων και απόδοσης αρχικών τιμών, • να αναφέρονται σε συγκεκριμένο στοιχείο πίνακα, • να δηλώνουν και να δίνουν αρχική τιμή σε αλφαριθμητικά, • να δίνουν τον ορισμό του τύπου του δείκτη, • να δηλώνουν μεταβλητές δείκτη και να αποδίδουν σε αυτές αρχική τιμή
Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: • Στην πρώτη, παρουσιάζουμε την έννοια του πίνακα σαν μια δομή δεδομένων της γλώσσας C. • Στη δεύτερη, παρουσιάζουμε τη χρήση δεικτών για τους πίνακες της γλώσσας C
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας και των εκπαιδευόμενων.

Αξιολόγηση	Η ενότητα του μαθήματος αυτού χρησιμοποιεί σε κάθε δραστηριότητα και σε κάθε μαθησιακό αντικείμενο ασκήσεις αυτοαξιολόγησης. Τέτοιες ασκήσεις μπορεί να είναι ερωτήσεις επίλυσης προβλημάτων ή ερωτήσεις αντικειμενικού τύπου. Για τη διεκπεραίωση αυτών των ασκήσεων μπορεί να χρησιμοποιηθούν διάφορα εργαλεία λογισμικού όπως προγράμματα σχεδιασμού ερωτήσεων συμπλήρωσης κενού, σταυρόλεξα κλπ. ή λογισμικό σχεδιασμού ανάπτυξης προγραμμάτων στη γλώσσα C.
Συνολικός χρόνος Ενότητας	1 εβδομάδα

3.4.1 Δραστηριότητα 1 - Εισαγωγή στους Πίνακες

Τίτλος Δραστηριότητας	***S4_LA7*** - Εισαγωγή στους Πίνακες
Τίτλος Ενότητας	Πίνακες, Δείκτες
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Η χρήση μεταβλητών με δείκτες στην άλγεβρα είναι ένας ιδιαίτερα δυναμικός τρόπος για τη διαχείριση μεγάλου αριθμού δεδομένων ιδίου τύπου. Οι γλώσσες προγραμματισμού δανείζονται την έννοια των μεταβλητών με δείκτες και χρησιμοποιούν τους πίνακες για τον ίδιο λόγο. Ένας πίνακας είναι ένα σύνολο δεδομένων του ίδιου τύπου, στα οποία αναφερόμαστε με ένα όνομα μεταβλητής και έναν ακέραιο (δείκτη) με αρχή το μηδέν. Ένας δείκτης είναι μια μεταβλητή που περιέχει μια διεύθυνση μνήμης. Για να βρούμε τη διεύθυνση μνήμης μιας μεταβλητής χρησιμοποιούμε τον τελεστή &. Ένας δείκτης μπορεί να δείχνει στο πρώτο στοιχείο ενός πίνακα και κάνοντας πράξεις πάνω στους δείκτες μπορούμε να προσπελάσουμε όλα τα στοιχεία του πίνακα.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση του τύπου του πίνακα, • να δίνουν παραδείγματα δήλωσης πινάκων και απόδοσης αρχικών τιμών, • να αναφέρονται σε συγκεκριμένο στοιχείο πίνακα
Μαθησιακά Αντικείμενα	Εισαγωγή στους πίνακες Απόδοση Τιμών σε Πίνακες
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 7 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S4_LA10LO100 και S4_LA10LO105.
Λέξεις Κλειδιά	Μονοδιάστατος πίνακας, διδιάστατος πίνακας, πολυδιάστατος πίνακας

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA10LO100**
Τίτλος	**Εισαγωγή στους πίνακες**
Τίτλος δραστηριότητας	Εισαγωγή στους Πίνακες
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι πίνακες. Ένας πίνακας (array) στη C είναι ένας αριθμός από ομοειδή δεδομένα, στο καθένα από τα οποία μπορούμε να αναφερθούμε με το όνομα του πίνακα και τον αύξοντα αριθμό του μέσα στον πίνακα (array). Δίνονται παραδείγματα πινάκων είτε μονοδιάστατων είτε διδιάστατων και περιγράφονται τα χαρακτηριστικά τους.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Πίνακας τύπου double, int ή *float,* δείκτης πίνακα, μέγεθος πίνακα, διδιάστατος πίνακας
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρονται σε συγκεκριμένο στοιχείο πίνακα • να γνωρίζουν την αναγκαιότητα της χρήσης των πινάκων • να δώσουν τουλάχιστον τρία παραδείγματα δήλωσης πινάκων διαφορετικού τύπου και μεγέθους

Ένας πίνακας είναι ένα σύνολο δεδομένων (του ίδιου τύπου) στα οποία αναφερόμαστε με ένα σύμβολο μιας μεταβλητής και με έναν ακέραιο (δείκτη) με αρχή το μηδέν.

Ο πίνακας (ή διάταξη) έχει επομένως τα εξής δύο χαρακτηριστικά:

- Είναι διατεταγμένος. Τα επιμέρους στοιχεία ενός πίνακα είναι δυνατόν να απαριθμηθούν: πρώτο, δεύτερο, κ.ο.κ.
- Είναι ομοιογενής. Κάθε τιμή που αποθηκεύεται σε έναν πίνακα πρέπει να είναι του ιδίου τύπου. Είναι δυνατόν να οριστεί ένας πίνακας ακεραίων, ή ένας πίνακας αριθμών κινητής υποδιαστολής, αλλά όχι ένας μικτός πίνακας που να περιλαμβάνει ακεραίους και αριθμούς κινητής υποδιαστολής.

Ένας πίνακας στη γλώσσα C προσδιορίζεται:

- από τον τύπο δεδομένων των στοιχείων του

- από το μέγεθος του πίνακα, που είναι ο αριθμός των στοιχείων του πίνακα.

Η δήλωση ενός πίνακα πρέπει να πραγματοποιείται (όπως και στις μεταβλητές) πριν από τη χρήση του.

Ο τρόπος σύνταξης για τη δήλωση ενός πίνακα είναι:

τύπος_δεδομένων όνομα_πίνακα[μέγεθος];

Παραδείγματα

double X[Nmax]: πίνακας με Nmax δεδομένα τύπου double (Nmax: σταθερά)

int a[10]: ένας πίνακας με 10 δεδομένα τύπου int που αντιστοιχούν στις μεταβλητές a[0], a[1], a[2], …., a[9].

Ακολουθούν μερικές δηλώσεις πινάκων:

```
main()
{
    float times[365];       /* πίνακας με 365 στοιχεία τύπου float */
    char cities[12];        /* πίνακας με 12 στοιχεία τύπου char */
    int numbers[50];        /* πίνακας με 50 στοιχεία τύπου int */
}
```

Οι αγκύλες [και] μάς λένε ότι πρόκειται για πίνακα και ο αριθμός που είναι μέσα στις αγκύλες μας δείχνει τον αριθμό των στοιχείων του πίνακα. Για να αναφερθούμε σ' ένα συγκεκριμένο στοιχείο του πίνακα, χρησιμοποιούμε έναν αριθμό που ονομάζεται ***δείκτης πίνακα*** και που η αρίθμησή του αρχίζει από το 0.

Ο πίνακας, δηλαδή, είναι μια ***μεταβλητή με δείκτη*** και για να αναφερθούμε πλήρως σ' ένα στοιχείο του, χρειαζόμαστε το όνομα του πίνακα και την τιμή του δείκτη του πίνακα. Έτσι, π.χ. το times[0] είναι το πρώτο στοιχείο του πίνακα times και το times[364] είναι το 365^{ο} και τελευταίο στοιχείο του.

Για το μέγεθος ενός πίνακα πρέπει να γνωρίζουμε τα εξής:

- Για τους πίνακες που έχουμε δει μέχρι τώρα, πρέπει το μέγεθός τους να είναι γνωστό κατά τη διάρκεια της μεταγλώττισης
- Μπορεί να είναι μια σταθερά

```
int X[100];
```

- Μπορεί να έχει δηλωθεί ως σταθερά του προ-επεξεργαστή με το #define

```
#define SIZE 100
int X[SIZE];
```

- Μπορεί να βρεθεί από τη λίστα αρχικοποίησης στην αρχή του προγράμματος. Οι πίνακες αυτοί ονομάζονται **στατικοί**. Υπάρχουν και **δυναμικοί πίνακες** των οποίων το μέγεθος προσδιορίζεται κατά τη διάρκεια εκτέλεσης του προγράμματος.

Πολυδιάστατοι Πίνακες

- Στη C μπορούμε να ορίσουμε και πολυδιάστατους πίνακες. Για παράδειγμα, ένας διδιάστατος πίνακας μπορεί να αποτελείται από ν γραμμές και κ στήλες, όπου ν, κ ακέραιοι.

Παράδειγμα: Διδιάστατος Πίνακας

Βαθμοί 5 φοιτητών σε 3 μαθήματα

int bathmoi[5][3];

Αρχικοποίηση:

bathmoi[5][3]=

{{7,8,6},

{3,7,4},

{3,5,5},

{7,9,8},

{5,8,3}}

Η γλώσσα C αποθηκεύει τους διδιάστατους πίνακες σε μια μήτρα με γραμμές και στήλες, όπου ο πρώτος δείκτης δηλώνει τη γραμμή και ο δεύτερος δείκτης δηλώνει τη στήλη. Αυτή η δομή σημαίνει ότι ο δεξιότερος δείκτης αλλάζει ταχύτερα από τον αριστερότερο όταν προσπελαύνουμε τα στοιχεία του πίνακα με τη σειρά που η C τα αποθηκεύει στη μνήμη.

- Ένας πίνακας μπορεί να έχει στοιχεία που είναι πίνακες

int array[4][12]; /* 4 γραμμές, 12 στήλες */

array[0][1] /* το στοιχείο της πρώτης γραμμής και δεύτερης στήλης */

array[0,1] /* ΛΑΘΟΣ */

Ερωτήσεις αντικειμενικού τύπου - S4_LA10LO101

Να απαντήσετε ποιες από τις παρακάτω προτάσεις είναι Σωστές ή λανθασμένες;

1. Οι δομές δεδομένων διακρίνονται σε στατιστικές και δυναμικές
2. Δυναμικές είναι οι δομές που αποθηκεύονται σε συνεχόμενες θέσεις μνήμης
3. Ένας πίνακας έχει σταθερό μέγεθος αλλά μεταβαλλόμενο περιεχόμενο
4. Ένας πίνακας μπορεί να αποθηκεύσει ακεραίους αριθμούς και ονόματα
5. Τα στοιχεία ενός πίνακα είναι απαραίτητο να είναι όλα του ίδιου τύπου
6. Η θέση ενός στοιχείου σ' έναν διδιάστατο πίνακα καθορίζεται από δύο αριθμούς
7. Οι διαστάσεις ενός πίνακα μπορούν να μεταβληθούν κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου
8. Στο pinakas[a][b] το a αντιστοιχεί στη γραμμή του πίνακα και το b στη στήλη
9. Οι διδιάστατοι πίνακες μπορούν να θεωρηθούν ως μονοδιάστατοι πίνακες, όπου σε κάθε θέση τους βρίσκεται ένας άλλος μονοδιάστατος πίνακας.
10. Η δήλωση ενός πίνακα πρέπει να πραγματοποιείται (όπως και στις μεταβλητές) πριν από τη χρήση του.

Άσκηση αυτοαξιολόγησης - S4_LA10LO102

Ένας πίνακας που χρησιμοποιεί δύο δείκτες για τον πλήρη προσδιορισμό της θέσης του κάθε στοιχείου του είναι πάντα

α) γραμμικός β) διδιάστατος γ) μονοδιάστατος δ) πολυδιάστατος

Άσκηση αυτοαξιολόγησης - S4_LA10LO103

Να γραφεί πρόγραμμα σε C, όπου θα δηλώνεται μια μεταβλητή x τύπου int, ένας πίνακας y, 34 ακεραίων στοιχείων, μια μεταβλητή z, τύπου char και ένας πίνακας w τύπου char που θα περιλαμβάνει τις τιμές 'a', 'b', 'c'.

Άσκηση αυτοαξιολόγησης - S4_LA10LO104

Ο τελεστής sizeof() επιστρέφει το μέγεθος (σε bytes) μίας μεταβλητής, ενός array, ή όπως θα δούμε αργότερα μίας δομής που εμείς έχουμε κατασκευάσει. Για τις προκαθορισμένες δομές (int, short κλπ.) ΔΕΝ υπάρχουν standards, αλλά το μέγεθός τους μπορεί να εξαρτηθεί από το εκάστοτε λειτουργικό σύστημα. Τι θα εμφανίσει το παρακάτω πρόγραμμα για το δικό σας λειτουργικό σύστημα;

```
#include <stdio.h>

int main()
{
    short x[100];
    float a[100];
    double d[100];
    printf("Bytes allocated for short array of 100 pos: %d.\n", sizeof(x));
    printf("Bytes allocated for float array of 100 pos: %d.\n", sizeof(a));
    printf("Bytes allocated for double array of 100 pos: %d.\n", sizeof(d));
    return 0;
}
```

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA10LO105**
Τίτλος	**Απόδοση Τιμών σε Πίνακες**
Τίτλος δραστηριότητας	Εισαγωγή στους Πίνακες
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι τρόποι απόδοσης τιμών σε πίνακες είτε με την αρχικοποίηση του πίνακα ταυτόχρονα με τη δήλωση του ονόματος και του μεγέθους του, είτε με την ανάθεση τιμών με τρεις τρόπους δηλαδή εισαγωγή τιμών από μονάδα εισόδου, εισαγωγή τιμών σε κάθε στοιχείο ξεχωριστά και εισαγωγή τιμών μέσω ενός βρόχου επανάληψης.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Αρχικοποίηση πινάκων, ανάθεση τιμών σε πίνακες
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να εξηγήσουν με παραδείγματα την απόδοση αρχικών τιμών σε πίνακα • να αναθέσουν τιμές σε τρεις τουλάχιστον πίνακες δίνοντας τιμές από το πληκτρολόγιο • να αναθέσουν τιμές σε τρεις τουλάχιστον πίνακες δίνοντας τιμές σε κάθε στοιχείο ξεχωριστά

Η απόδοση των τιμών σε έναν πίνακα μπορεί να γίνει είτε με την αρχικοποίηση του πίνακα, δηλαδή ταυτόχρονα με τη δήλωση του ονόματός του δίνεται και το μέγεθός του και οι τιμές που θα περιέχει, είτε μετά την δήλωση του ονόματος και του μεγέθους του γίνεται ανάθεση τιμών με τους παρακάτω τρόπους:

- Εισαγωγή τιμών από μονάδα εισόδου
- Τιμές σε κάθε στοιχείο ξεχωριστά
- Τιμές μέσω ενός βρόχου επανάληψης

Αρχικοποίηση

Ο τελεστής ανάθεσης μπαίνει μετά τη διάσταση του πίνακα και ακολουθεί μέσα σε άγκιστρα η λίστα με τις τιμές που θα αποδοθούν στα στοιχεία του πίνακα. Η πρόταση

float num[5] = {1, 2, 3.5, 4, 5};

αρχικοποιεί όλα τα στοιχεία του πίνακα num με τις τιμές της λίστας, ενώ η πρόταση

float num[5] = {1, 2, 3.5};

αρχικοποιεί τα 3 πρώτα στοιχεία δηλαδή τα num[0], num[1] και num[2] με τις τιμές της λίστας και τα υπόλοιπα με μηδέν.

float num[5] ={[0]=1, [2]=3.5};

αρχικοποιεί το 1ο και 3ο στοιχείο με τις τιμές 1 και 3.5 αφήνοντας τα υπόλοιπα 0.

float num[2][2]={ 1, 2, 3.5, 4};

αρχικοποιεί έναν πίνακα δυο διαστάσεων γραμμή προς γραμμή.

float num[2][2]={{1, 2},{3.5, 4}};

αρχικοποιεί έναν πίνακα δυο διαστάσεων γραμμή προς γραμμή διαχωρίζοντας την κάθε γραμμή.

- **Ανάθεση τιμών** μπορεί να γίνει με την εισαγωγή δεδομένων από το πληκτρολόγιο

Παράδειγμα

int num[5];

scanf(”%d %d %d %d %d”, &num[0], &num[1], &num[2], &num[3], &num[4],);

- **Ανάθεση τιμών** σε κάθε στοιχείο ξεχωριστά εφόσον έχει δηλωθεί το όνομα και το μέγεθος του πίνακα

Παράδειγμα

int num[5];

num[0]=4; num[1]=4; num[2]=4; num[3]=4; num[4]=4;

- **Ανάθεση τιμών** με τη χρήση επανάληψης όπως θα δούμε στην επόμενη ενότητα.

Ακολουθεί ένα πρόγραμμα που καταχωρεί τον αριθμό των ημερών ανά μήνα σ’ έναν πίνακα με απόδοση αρχικών τιμών και μετά τους εκτυπώνει.

```
/* prog41.c - απόδοση αρχικών τιμών στις ημέρες κάθε μήνα */
#include <stdio.h>
#define MONTHS 12
int days[MONTHS] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
main()
{
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 1, days[0]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 2, days[1]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 3, days[2]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 4, days[3]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 5, days[4]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 6, days[5]);
```

```
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 7, days[6]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 8, days[7]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 9, days[8]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 10, days[9]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 11, days[10]);
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 12, days[11]);
    getchar();
    return 0;
}
```

Το αποτέλεσμα θα είναι ως εξής:

Ο μήνας 1 έχει 31 ημέρες.
Ο μήνας 2 έχει 28 ημέρες.
Ο μήνας 3 έχει 31 ημέρες.
Ο μήνας 4 έχει 30 ημέρες.
Ο μήνας 5 έχει 31 ημέρες.
Ο μήνας 6 έχει 30 ημέρες.
Ο μήνας 7 έχει 31 ημέρες.
Ο μήνας 8 έχει 31 ημέρες.
Ο μήνας 9 έχει 30 ημέρες.
Ο μήνας 10 έχει 31 ημέρες.
Ο μήνας 11 έχει 30 ημέρες.
Ο μήνας 12 έχει 31 ημέρες.

Βλέπουμε στο παραπάνω παράδειγμα ότι ο αριθμός των στοιχείων της λίστας πρέπει να είναι ίδιος με το μέγεθος του πίνακα. Αν δώσουμε περισσότερα στοιχεία σ' έναν πίνακα απ' ότι είναι το δηλωμένο μέγεθός του, τότε αυτό θεωρείται λάθος.

Η C, όμως, δεν επιτρέπει την καταχώρηση τιμών από έναν πίνακα σ' έναν άλλον, ούτε μπορούμε να χρησιμοποιήσουμε τη μορφή της λίστας στοιχείων μέσα σε άγκιστρα, εκτός κι αν πρόκειται για απόδοση αρχικών τιμών.

Ακολουθεί ένα πρόγραμμα που μας λέει τι δεν μπορούμε να κάνουμε με τους πίνακες στη C:

```
#include <stdio.h>

#define SIZE 5

main()

{
    int pina[SIZE] = {5, 3, 2, 8};   /* όλα εντάξει */
    int pinb[SIZE];
    pinb = pina;                     /* δεν επιτρέπεται */
    pinb[SIZE] = pina[SIZE];         /* ανεπίτρεπτο */
    pinb[SIZE] = {5, 3, 2, 8};       /* δεν δουλεύει εδώ */
}
```

Ερωτήσεις αντικειμενικού τύπου - S4_LA10LO106

Να απαντήσετε ποιες από τις παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες;

1. Η γλώσσα C επιτρέπει την καταχώρηση τιμών από έναν πίνακα σ' έναν άλλον
2. Ο αριθμός των στοιχείων της λίστας με την οποία αρχικοποιούμε έναν πίνακα πρέπει να είναι ίδιος με το μέγεθος του πίνακα
3. Η απόδοση των τιμών σε ένα πίνακα μπορεί να γίνει μόνο με την αρχικοποίηση του πίνακα
4. Η εντολή float num[6] ={[1]=1, [4]=3.5}; αρχικοποιεί το 2ο και 5ο στοιχείο με τις τιμές 1 και 3.5 αφήνοντας τα υπόλοιπα 0
5. Η εντολή float num[2][2]={ 1, 2, 3.5, 4, 5, 1}; αρχικοποιεί έναν πίνακα δύο διαστάσεων γραμμή προς γραμμή
6. Η εντολή float num[5] = {1, 2, 3.5}; αρχικοποιεί τα 3 πρώτα στοιχεία με τις τιμές της λίστας και τα υπόλοιπα με μηδέν
7. Ο τελεστής ανάθεσης μπαίνει μετά τη διάσταση του πίνακα και ακολουθεί μέσα σε άγκιστρα η λίστα με τις τιμές που θα αποδοθούν στα στοιχεία του πίνακα
8. Ανάθεση τιμών ενός πίνακα δεν μπορεί να γίνει με την εισαγωγή δεδομένων από το πληκτρολόγιο
9. Η εντολή int num[5] ={[0]=4, [2]=3}; αρχικοποιεί το 1ο και 2ο στοιχείο με τις τιμές 4 και 3 αφήνοντας τα υπόλοιπα 0
10. Ανάθεση τιμών ενός πίνακα μπορεί να γίνει σε κάθε στοιχείο ξεχωριστά εφόσον έχει δηλωθεί το όνομα και το μέγεθος του πίνακα

Άσκηση αυτοαξιολόγησης - S4_LA10LO107

Να καταγράψετε την εντολή που αρχικοποιεί έναν πίνακα ακεραίων με όνομα NUMBERS και περιεχόμενο τους ζυγούς αριθμούς από το 10 έως το 20.

Επίσης να εκτυπώσετε το πρώτο στοιχείο του πίνακα με την παρακάτω μορφή:

το πρώτο στοιχείο του πίνακα είναι το 10

Άσκηση αυτοαξιολόγησης - S4_LA10LO108

Ποια από τις παρακάτω εντολές αυξάνει το πρώτο στοιχείο του πίνακα Α κατά 1 μονάδα.

Α. A[1]=A[0]+1

Β. A[0]=A[0]+1

Γ. A[0]=A[1]+1

Δ. A[1]=A[1]++

3.4.2 Δραστηριότητα 2 - Δείκτες σε Πίνακες

Τίτλος Δραστηριότητας	**S4_LA11 -** Δείκτες σε Πίνακες
Τίτλος Ενότητας	Πίνακες, Δείκτες
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Στη δραστηριότητα αυτή περιγράφονται οι δείκτες, μία άλλη περίπτωση μεταβλητών, οι οποίες δεν περιέχουν μία τιμή αλλά μία διεύθυνση μνήμης. Στη διεύθυνση αυτήν βρίσκεται τελικά η τιμή στην οποία αναφέρονται. Όταν δηλώνουμε έναν δείκτη ξεχωρίζει από τις απλές μεταβλητές με έναν αστερίσκο ο οποίος προηγείται του ονόματος. Αρχικοποιούμε έναν δείκτη, εξαναγκάζοντάς τον να δείχνει μία υπάρχουσα μεταβλητή με τη χρήση του συμβόλου & (με το οποίο ανατίθεται στο δείκτη η διεύθυνση της μεταβλητής) ή δείχνοντας το πρώτο στοιχείο ενός πίνακα.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τρία τουλάχιστον παραδείγματα δήλωσης δεικτών στη γλώσσα C • να κατανοήσουν τη διαφορά όταν αναφέρονται στη διεύθυνση μιας μεταβλητής και όταν αναφέρονται στο περιεχόμενό της • να χρησιμοποιούν τους δείκτες για να αλλάζουν το περιεχόμενο μεταβλητών • να δώσουν τρία τουλάχιστον παραδείγματα δεικτών που δείχνουν σε πίνακες • να αναφέρονται στο επόμενο ή στο προηγούμενο στοιχείο πινάκων με χρήση δεικτών • να δώσουν παραδείγματα που κάνουν πράξεις με τις τιμές ενός δείκτη
Μαθησιακά Αντικείμενα	Δείκτες Σχέση δεικτών και πινάκων
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 8 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S4_LA11LO109 και S4_LA11LO114.
Λέξεις Κλειδιά	Δείκτης, τελεστής έμμεσης αναφοράς, διεύθυνση μνήμης μεταβλητής, τύπος δείκτη, αριθμητική δεικτών, πρόσθεση δείκτη, αφαίρεση δείκτη

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA11LO109**
Τίτλος	Δείκτες
Τίτλος δραστηριότητας	Δείκτες σε Πίνακες
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται η έννοια του δείκτη και πώς αυτός δείχνει στη διεύθυνση μιας μεταβλητής. Δίνονται πολλά παραδείγματα δήλωσης δεικτών με συγκεκριμένο τύπο ανάλογα με τη μεταβλητή στην οποία δείχνουν. Αναφέρονται οι δύο τελεστές που σχετίζονται με τις τιμές των δεικτών είτε για να συνδέσουν το δείκτη με τη διεύθυνση μιας μεταβλητής, είτε για να αναφερθούν στο περιεχόμενό της.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Δείκτης, τελεστής έμμεσης αναφοράς, διεύθυνση μνήμης μεταβλητής, τύπος δείκτη
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τρία τουλάχιστον παραδείγματα δήλωσης δεικτών στη γλώσσα C • να κατανοήσουν την διαφορά όταν αναφέρονται στη διεύθυνση μιας μεταβλητής και όταν αναφέρονται στο περιεχόμενό της • να χρησιμοποιούν τους δείκτες για να αλλάζουν το περιεχόμενο μεταβλητών

Ένας **δείκτης (pointer)** είναι μία μεταβλητή που μπορεί να λάβει ως τιμή τη διεύθυνση μιας μεταβλητής. Μια κανονική μεταβλητή αποθηκεύει συγκεκριμένες τιμές (άμεση αναφορά - direct reference). Ένας δείκτης αποθηκεύει τη διεύθυνση μιας μεταβλητής που περιέχει μια συγκεκριμένη τιμή (έμμεση αναφορά - indirect reference). Οι δείκτες είναι ένα ισχυρό χαρακτηριστικό για τη γλώσσα C αν και είναι δύσκολοι στη διαχείρισή τους. Σχετίζονται άμεσα με τους πίνακες και τα αλφαριθμητικά, καθώς το όνομα ενός πίνακα (άρα και ενός αλφαριθμητικού) ταυτίζεται με τη διεύθυνση του πρώτου στοιχείου του. Χρειαζόμαστε κάποιο τρόπο να ξεχωρίσουμε τις δηλώσεις των «απλών» μεταβλητών από τις δηλώσεις των «ειδικών» μεταβλητών που ονομάζονται δείκτες. Όπως και οποιαδήποτε άλλη μεταβλητή,

ένας δείκτης πρέπει να δηλωθεί πριν από τη χρήση του. Η δήλωση ενός δείκτη πραγματοποιείται ως εξής:

τύπος_βάσης *μεταβλητή_δείκτης;

Για παράδειγμα, η δήλωση **int *p;** δηλώνει ότι η μεταβλητή p είναι δείκτης προς ακέραιο. Η γλώσσα προγραμματισμού C διαθέτει δύο βασικούς τελεστές που σχετίζονται με τιμές δεικτών:

Τελεστής έμμεσης αναφοράς &: εφαρμόζεται μπροστά από μεταβλητές και δείχνει τη διεύθυνσή τους. Για παράδειγμα:

p=&v Ο δείκτης p δείχνει την μεταβλητή v

Τελεστής *: εφαρμόζεται μπροστά από έναν δείκτη και δείχνει την τιμή που είναι αποθηκευμένη στη διεύθυνση που δείχνει ο δείκτης.

v=*p Επιστρέφει την τιμή της μεταβλητής προς την οποία ‘’δείχνει’’ ο δείκτης

Κάθε δείκτης δηλώνεται με έναν τύπο (char, int, float κλπ.) που οφείλει να εκφράζει τον τύπο της μεταβλητής στην οποία δείχνει.

Άλλο παράδειγμα, στο ακόλουθο τμήμα προγράμματος:

```
int x = 3;

int *p;

p = &x; /*αναθέτει στο δείκτη p τη διεύθυνση μνήμης της μεταβλητής x. */

printf("The value of x is: ", *p); /* εκτυπώνει την τιμή της μεταβλητής x, δηλαδή 3 */
```

Άλλο Παράδειγμα

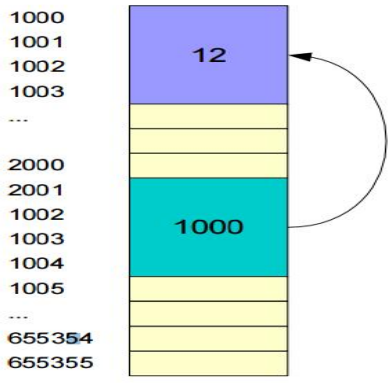
```
int i;

int *p;

p = &i;
```

Η p έχει την τιμή 1000

Η *p έχει την τιμή 12



Εικόνα 8. Αναπαράσταση δεικτών που αναφέρονται στο παράδειγμα

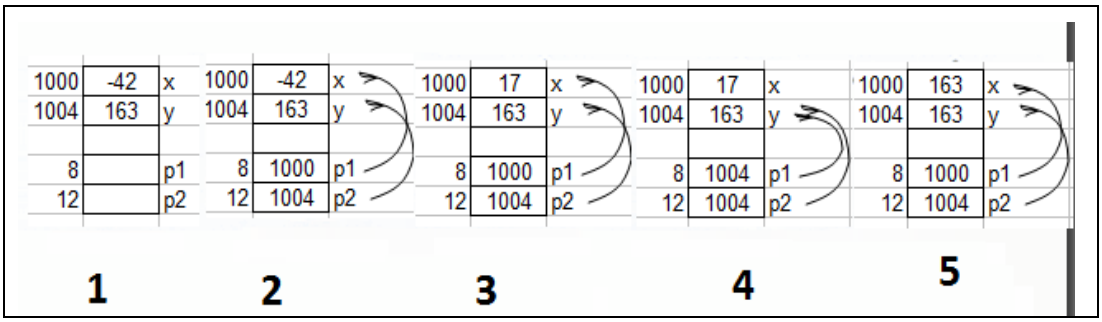
Είδαμε στα προηγούμενα ότι οι δείκτες (pointers) αποτελούν έναν συμβολικό τρόπο για να χρησιμοποιούμε διευθύνσεις μεταβλητών. Στην πραγματικότητα, ο συμβολισμός των πινάκων είναι μια μεταμφιεσμένη χρήση των δεικτών.

Το όνομα ενός πίνακα είναι και ένας δείκτης που δείχνει στο πρώτο στοιχείο του. Δηλαδή, αν pina είναι ένας πίνακας, τότε το pina ισοδυναμεί με το &pina[0] δηλαδή η διεύθυνση μνήμης του πρώτου στοιχείου του πίνακα.

Άσκηση αυτοαξιολόγησης - S4_LA11LO110

Να αντιστοιχίσετε τα νούμερα της εικόνας με τους κώδικες που βλέπετε στον πίνακα:

1	2	3	4	5
p1=&x ; P2=&y ;	int x, y; int *p1, *p2; x=-42 ; y=163 ;	*p1=17 ;	*p1=*p2 ;	p1=p2 ;



Άσκηση αυτοαξιολόγησης - S4_LA11LO111

Να αντιστοιχίσετε τις προτάσεις της πρώτης στήλης του παρακάτω πίνακα με τα σχόλια προγράμματος της δεύτερης στήλης του πίνακα που εξηγούν τη λειτουργία τους.

Πρόταση	**Σχόλιο προγράμματος**
int x=1;	/* Αρχικοποίηση y */
int y=2;	/* y = x άρα y=1 */
int *p, *q;	/* Δήλωση p, q */
p = &x;	/* Αρχικοποίηση x */
y = *p;	/* x=x+1 άρα x=2 */
*p = 0;	/* Τα περιεχόμενα του p αντιγράφονται στον q */
*p+=1;	/* Το p θα δείχνει στο επόμενο στοιχείο */
++*p;	/* x=x+1 άρα x=3 */
(*p)++;	/* x=x+1 άρα x=1 */
*p++;	/* x=0 */
q = p;	/* Ο p δείχνει στην x */

Άσκηση αυτοαξιολόγησης - S4_LA11LO112

Έστω ότι ισχύουν οι παρακάτω προτάσεις:

```
int x=1, y=2, z=[10];
int *ip, *iq;
ip=&x;
*ip=*ip+1;
y=*ip+3;
*ip=0;
ip=&z[6];
iq=ip;
```

Να βάλετε τα παρακάτω σχόλια προγράμματος στη σωστή σειρά με βάση τις παραπάνω εντολές:

```
/*η x παίρνει την τιμή 5*/
/*ο ip δείχνει τώρα την x*/
/*η x παίρνει την τιμή 0*/
/*ο iq δείχνει τώρα το z[6] δηλαδή εκεί που δείχνει και ο ip */
/*ο ip δείχνει τώρα το z[6] */
/*η μεταβλητή που δείχνει ο ip (η x) αυξάνεται κατά 1*/
```

Άσκηση αυτοαξιολόγησης - S4_LA11LO113

Υπολογίστε τους τύπους των εκφράσεων δεξιά και αριστερά από κάθε ανάθεση και γράψτε ποιες από τις παρακάτω γραμμές είναι σωστές και ποιες όχι.

```
int a, b, c, *p1, *p2, *p3;
double da, db, dc, *pd1, *pd2, *pd3;
    p1 = &a;
    *p1 = 4;
    p2 = &p1;
    *p2 = *c;
    p3 = a;
    p2 = p3;
    pd1 = p2;
    pd2 = &da;
    dc = *pd3;
    db = *p2;
```

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA11LO114**
Τίτλος	Σχέση μεταξύ δεικτών και πινάκων
Τίτλος δραστηριότητας	Δείκτες σε Πίνακες
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται η **αριθμητική των δεικτών.** Ένας δείκτης μπορεί να δείχνει στο πρώτο στοιχείο ενός πίνακα και κάνοντας πράξεις πάνω στους δείκτες μπορούμε να προσπελάσουμε όλα τα στοιχεία του πίνακα. Επιτρέπονται μόνο 2 αριθμητικές πράξεις, πρόσθεση και αφαίρεση. Γενικά κάθε φορά που ένας δείκτης αυξάνεται κατά 1 δείχνει στην θέση μνήμης του επόμενου στοιχείου του βασικού του τύπου, ενώ αν μειώνεται κατά 1 δείχνει στη θέση μνήμης του προηγούμενου στοιχείου του βασικού του τύπου.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Βιντεοδιάλεξη, παρουσίαση
Λέξεις Κλειδιά	Αριθμητική δεικτών, πρόσθεση δείκτη, αφαίρεση δείκτη
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τρία τουλάχιστον παραδείγματα δεικτών που δείχνουν σε πίνακες • να αναφέρονται στο επόμενο ή στο προηγούμενο στοιχείο πινάκων με χρήση δεικτών • να δώσουν παραδείγματα που κάνουν πράξεις με τις τιμές ενός δείκτη

Η C είναι από τις ελάχιστες γλώσσες που υποστηρίζουν αριθμητική δεικτών (pointer arithmetic), και σε αυτό οφείλει μεγάλο μέρος της ταχύτητας και της ευελιξίας της.

Αν ένας δείκτης *p* "δείχνει" στο πρώτο στοιχείο ενός πίνακα A και *k* είναι οποιοσδήποτε ακέραιος, η παράσταση *p+k* είναι ισοδύναμη με την παράσταση &A[k]. Η ιδιότητα αυτή της C ονομάζεται ***αριθμητική δεικτών***.

- Στην πραγματικότητα, για έναν πίνακα A η έκφραση A[k] είναι συντομογραφία της έκφρασης *(A + k). Όταν προσθέτουμε ή αφαιρούμε από έναν δείκτη μια σταθερή τιμή, κάνουμε το δείκτη να δείχνει στο προηγούμενο ή στο επόμενο στοιχείο

Έστω ότι ορίζεται ένας πίνακας A, πέντε στοιχείων ως εξής:

int A[5] = {2,3,5,7,9};

Αν ο *p* είναι δείκτης προς ακέραιο που "δείχνει" στο πρώτο στοιχείο του πίνακα Α σύμφωνα με την εντολή:

int *p = &A[0];

τότε ο δείκτης p μπορεί να χρησιμοποιηθεί για την προσπέλαση οποιουδήποτε στοιχείου του πίνακα Α. Αυξάνοντας για παράδειγμα την τιμή του δείκτη κατά 3, ο δείκτης "δείχνει" στο 4^{ο} στοιχείο του πίνακα.

Η εντολή για παράδειγμα:

printf(" %d ", *(p+3));

εκτυπώνει το περιεχόμενο του 4^{ου} στοιχείου του πίνακα.

Η αντιστοιχία μεταξύ δεικτοδότησης πινάκων και αριθμητικής δεικτών γίνεται ακόμη πιο εμφανής από το γεγονός ότι το **όνομα ενός πίνακα είναι συνώνυμο με τη διεύθυνση του πρώτου στοιχείου του** καθώς και από το ότι αν ο p είναι δείκτης σε πίνακα, οι εκφράσεις *(*p*+*k*) και *p*[*k*] είναι ισοδύναμες.

Στο ανωτέρω παράδειγμα, η εντολή:

p = A;

έχει ως αποτέλεσμα την ανάθεση της διεύθυνσης του πρώτου στοιχείου του πίνακα Α, στη μεταβλητή δείκτη p. Επίσης, η προηγούμενη εντολή εκτύπωσης, μπορεί να γραφεί και ως:

printf(" %d ", p[3]);

Μιας και τα arrays καταλαμβάνουν συνεχόμενη μνήμη, μπορούμε με απλές αριθμητικές πράξεις να πηγαίνουμε τον pointer p σε όποιο στοιχείο του Α θέλουμε, και το *p να είναι ο ακέραιος που περιέχει αυτό το στοιχείο του Α. Στην πραγματικότητα, μια μεταβλητή τύπου *int* (ακέραιος αριθμός) δεν δεσμεύει μονάχα ένα "κουτάκι" μνήμης, αλλά πολλά συνεχόμενα (συνήθως 4, αλλά εξαρτάται από τον κάθε υπολογιστή). Το κάθε "κουτάκι" αντιστοιχεί σε 1 byte κι αν θέλετε να δείτε πόσα bytes δεσμεύονται για καθέναν από τους τέσσερις βασικούς τύπους μεταβλητών στη C στο δικό σας υπολογιστή, μπορείτε να χρησιμοποιήσετε τον τελεστή: ***sizeof***, που έχουμε εξηγήσει σε προηγούμενη ενότητα.

Οπότε και ο πίνακας Α που έχουμε δηλώσει να έχει θέσεις π.χ. για 5 ακεραίους καταλαμβάνει 20 bytes μνήμης (4 για καθένα από τα 5 στοιχεία του).

Τα παραπάνω εξηγούν γιατί όταν δηλώνουμε έναν pointer πρέπει να δηλώσουμε και σε τι τύπο μεταβλητών θέλουμε να δείχνει. Για να "ξέρει" πόσα bytes καταλαμβάνει στη μνήμη η μεταβλητή στην οποία θα τον βάλουμε να δείχνει.

Από τη στιγμή που ξέρει λοιπόν, όταν γράφουμε…

p += 4; /* ισοδυναμεί με p = p + 4; */

μετακινείται προς τα δεξιά όχι 4 "κουτάκια" μνήμης, αλλά 16 (επειδή του έχουμε πει να δείχνει σε μεταβλητές τύπου *int* και ο τύπος *int* καταλαμβάνει 4 "κουτάκια", άρα 4*4=16). Μας γλιτώνει λοιπόν από τον πονοκέφαλο του χειροκίνητου υπολογισμού για κάθε τύπο.

Προσοχή: ο δείκτης είναι μεταβλητή και έτσι p = A και p++ είναι έγκυρες εκφράσεις. Το όνομα ενός πίνακα όμως δεν είναι μεταβλητή, επομένως κατασκευές όπως A = p και A++ δεν είναι έγκυρες.

Ακολουθεί ένα παράδειγμα που κάνει πράξεις με τις τιμές ενός δείκτη:

```
/* prog42.c - πρόσθεση δείκτη */
#include <stdio.h>
#define SIZE 4
main()
{
    short int pina[SIZE], *pti, index;
    float pinb[SIZE], *ptf;
    pti = pina;        /* καταχωρεί τη διεύθυνση του πίνακα σε δείκτη */
    ptf = pinb;
    for (index=0; index<SIZE; index++)
        printf("δείκτες + %d: %10u %10u\n", index, pti + index, ptf + index);
}
```

Το αποτέλεσμα θα είναι ως εξής:

δείκτες + 0: 56014 56026

δείκτες + 1: 56016 56030

δείκτες + 2: 56018 56034

δείκτες + 3: 56020 56038

Στην πρώτη γραμμή τυπώνονται οι πρώτες διευθύνσεις των δύο πινάκων, όποιες είναι αυτές. Στην επόμενη γραμμή βλέπουμε το αποτέλεσμα της πρόσθεσης του 1 στη διεύθυνση.

Για να δούμε τι γίνεται από κοντά:

56014 + 1 = 56016 ;

56026 + 1 = 56030 ;

Ξέρουμε ότι ο τύπος short int χρησιμοποιεί 2 bytes για αποθήκευση και ο τύπος float 4 bytes. Όταν λοιπόν προσθέτουμε 1 στον αντίστοιχο δείκτη, τότε η C προσθέτει μια μονάδα αποθήκευσης. Για τους πίνακες, όμως, αυτό σημαίνει ότι η διεύθυνση αυξάνεται στη διεύθυνση του επόμενου στοιχείου και όχι στο επόμενο byte.

Προσθέτοντας έναν ακέραιο σ' έναν δείκτη ή αυξάνοντας έναν δείκτη, αλλάζει η τιμή του δείκτη κατά τον αριθμό των bytes του στοιχείου που δείχνει ο δείκτης.

Ακολουθούν μερικές ενδιαφέρουσες εκφράσεις:

Η διεύθυνση *(dates + 2)* είναι ίδια με τη διεύθυνση *&dates[2]*

Το στοιχείο ***(dates + 2)** γράφεται και ως *dates[2]*

Αυτές οι αντιστοιχίες δείχνουν τη σχέση μεταξύ πινάκων και δεικτών. Αυτό σημαίνει ότι μπορούμε να χρησιμοποιήσουμε ένα δείκτη για να προσδιορίσουμε ένα συγκεκριμένο στοιχείο ενός πίνακα και να πάρουμε την τιμή του.

Δεν πρέπει, όμως, να συγχέουμε την τιμή της *(dates+2) με την τιμή *dates+2, γιατί:

**(dates + 2)* είναι η τιμή του 3ου στοιχείου του πίνακα dates

**dates + 2* είναι η τιμή του 1ου στοιχείου του πίνακα dates αυξημένο κατά 2

Μπορούμε συνεπώς να χρησιμοποιούμε και δείκτες για να δουλέψουμε με τους πίνακες.

Ακολουθεί ένα πρόγραμμα που κάνει την ίδια δουλειά με το πρόγραμμα που είδαμε προηγουμένως, μόνο που τώρα χρησιμοποιούμε δείκτες αντί για πίνακες:

```
/* prog43.c - χρησιμοποιεί δείκτες αντί για πίνακες */
#include <stdio.h>
#define MONTHS 12
int days[MONTHS] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
main()
{
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 1, *(days));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 2, *(days+1));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 3, *(days+2));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 4, *(days+3));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 5, *(days+4));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 6, *(days+5));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 7, *(days+6));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 8, *(days+7));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 9, *(days+8));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 10, *(days+9));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 11, *(days+10));
    printf ("Ο μήνας %d έχει %d ημέρες.\n", 12, *(days+11));
    getchar();
    return 0;
}
```

Οι Λειτουργίες των Δεικτών

Συνοπτικά θα λέγαμε ότι η C έχει πέντε βασικές λειτουργίες δεικτών. Για να δούμε από κοντά τα αποτελέσματα της κάθε λειτουργίας, τυπώνουμε την τιμή του δείκτη, δηλαδή τη διεύθυνση που δείχνει, την τιμή που είναι αποθηκευμένη στη διεύθυνση που δείχνει ο δείκτης και ακόμη και τη διεύθυνση του ίδιου του δείκτη.

Ακολουθεί ένα παράδειγμα:

```
/* prog45.c - λειτουργίες δεικτών */
#include <stdio.h>
main()
{
   short int pina[3] = {100, 200, 300};
```

```
    short int *ptr1, *ptr2;
    ptr1 = pina; /* καταχώρηση της διεύθυνσης του πίνακα σ' έναν δείκτη */
    ptr2 = &pina[2]; /* καταχώρηση της διεύθυνσης του 3ου στοιχείου */

    printf("ptr1 = %u, *ptr1 = %d, &ptr1 = %u\n", ptr1, *ptr1, &ptr1);
    ptr1++; /* αυξάνουμε τον δείκτη */
    printf("ptr1 = %u, *ptr1 = %d, &ptr1 = %u\n", ptr1, *ptr1, &ptr1);

    printf("ptr2 = %u, *ptr2 = %d, &ptr2 = %u\n", ptr2, *ptr2, &ptr2);
    ++ptr2; /* περνάμε τα όρια του πίνακα */
    printf("ptr2 = %u, *ptr2 = %d, &ptr2 = %u\n", ptr2, *ptr2, &ptr2);

    printf("ptr2 - ptr1 = %u\n", ptr2 - ptr1); /* τυπώνουμε τη διαφορά των δεικτών */
}
```

Το αποτέλεσμα θα είναι ως εξής:

```
ptr1 = 234, *ptr1 = 100, &ptr1 = 3606
ptr1 = 236, *ptr1 = 200, &ptr1 = 3606
ptr2 = 238, *ptr2 = 300, &ptr2 = 3604
ptr2 = 240, *ptr2 = 1910, &ptr2 = 3604
ptr2 - ptr1 = 2
```

Ας δούμε από κοντά τις πέντε βασικές λειτουργίες που μπορούμε να κάνουμε με τις μεταβλητές δείκτη:

1. Καταχώρηση. Μπορούμε να καταχωρήσουμε μια διεύθυνση σ' έναν δείκτη, είτε χρησιμοποιώντας ένα όνομα πίνακα ή χρησιμοποιώντας τον τελεστή διεύθυνσης & και την τιμή μιας μεταβλητής.

2. Εύρεση τιμής. Ο τελεστής * μάς δίνει την τιμή που βρίσκεται αποθηκευμένη στη θέση που δείχνεται από τον δείκτη.

3. Απόκτηση της διεύθυνσης ενός δείκτη. Και οι μεταβλητές δείκτη έχουν μια διεύθυνση και μια τιμή.

4. Αύξηση-μείωση ενός δείκτη. Με την αύξησή του ένας δείκτης μετακινείται στο επόμενο στοιχείο του πίνακα. Φυσικά, με ανάλογο τρόπο μπορούμε και να μειώσουμε έναν δείκτη. Η λειτουργία, όμως, ++ptr2 στο προηγούμενο παράδειγμα είχε σαν αποτέλεσμα ο ptr2 να μετακινηθεί κατά δύο bytes και να δείχνει οτιδήποτε μπορεί να αποθηκευθεί μετά τον πίνακα. Επίσης, πρέπει να θυμόμαστε ότι μπορούμε να χρησιμοποιούμε τους τελεστές αύξησης ή μείωσης με τις μεταβλητές δείκτη, αλλά όχι με σταθερές δείκτη. Δηλαδή, το ptr2 = urn++; δεν επιτρέπεται, όπως δεν επιτρέπεται και το 3++.

5. Διαφορά. Μπορούμε να βρούμε τη διαφορά μεταξύ δύο δεικτών.

Άσκηση αυτοαξιολόγησης - S4_LA11LO115

Ποιες από τις πιο κάτω εντολές είναι έγκυρες;

```
int t[10], i = 5, *p;
p = t;
p[2] = 3;
++p;
*p = 14;
scanf("%d", &p[i+4]);
*(t+i) = 33;
++t;
```

Άσκηση αυτοαξιολόγησης - S4_LA11LO116

Συμπληρώστε τον πίνακα με τις τιμές των μεταβλητών μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

```
int a, b, c;
int *ip1, *ip2, *ip3;
    a = 3;
    b = 4;
    ip1 = &a;
    *ip1 = 8;
    ip3 = &c;
    ip2 = ip3;
    b = *ip1;
    *ip2 = *ip1;
```

Άσκηση αυτοαξιολόγησης - S4_LA11LO117

Τι πιστεύετε ότι θα εμφανίσουν οι τελευταίες εντολές εκτύπωσης όταν εκτελεστεί το παρακάτω πρόγραμμα; Μεταξύ των ορισμάτων εκτύπωσης αφήστε 8 κενά.

```
#include <stdio.h>
#define MAX 10
int i_array[MAX] = { 0,1,2,3,4,5,6,7,8,9 };
int *i_ptr, count;
float f_array[MAX] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
0.7, 0.8, 0.9 };
float *f_ptr;
```

```
main()
{
    i_ptr = i_array;
    f_ptr = f_array;
    printf("%d\t%f\n", *i_ptr+3, *f_ptr+2);
    printf("%d\t%f\n", *i_ptr, *f_ptr);
    printf("%d\t%f\n", *i_ptr+10, *f_ptr+2);
    return 0;
}
```

Άσκηση αυτοαξιολόγησης - S4_LA11LO118

Αν υποθέσουμε ότι η διεύθυνση της μεταβλητής x είναι η 1245064 τι θα εκτυπώσει το παρακάτω πρόγραμμα:

```
#include <stdio.h>
void main()
{
    int x, *p1;
    x=5;
    p1=&x;
    printf("p1 is %d \n", p1);
    p1++;
    printf("(after ++) p1 is %d \n", p1);
    p1--;
    printf("(after --) p1 is %d \n", p1);
    getchar();
}
```

Συμβουλές!

- Μην προσπαθήσετε να εκτελέσετε μαθηματικές πράξεις, όπως διαίρεση, πολλαπλασιασμός, κλπ. με δείκτες.
- Με την αφαίρεση ή την πρόσθεση ένας δείκτης μεταβάλλεται με βάση το μέγεθος του τύπου δεδομένων που δείχνει.
- Αρχικοποιείτε τους δείκτες. Πρέπει να το κάνετε οι ίδιοι. Ο compiler δεν θα το κάνει αυτό για εσάς!
- Αν δεν θέλουμε να αρχικοποιήσουμε σε συγκεκριμένη διεύθυνση αρχικοποιούμε σε NULL:

```
int *ptr=NULL;
```

3.4.3 Δραστηριότητα 3 - Συμβολοσειρές

Τίτλος Δραστηριότητας	**S4_LA12 -** Συμβολοσειρές
Τίτλος Ενότητας	Πίνακες Δείκτες
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Η δραστηριότητα αυτή παρουσιάζει την έννοια της συμβολοσειράς στη C η οποία ορίζεται σαν μια σειρά δεδομένων του τύπου char που τερματίζονται με ένα μηδενικό χαρακτήρα (NULL). Η C χειρίζεται τη συμβολοσειρά με διάφορες συναρτήσεις που καλούνται και είναι σχεδιασμένες να εκτελούν τη λειτουργία τους μέχρι να συναντήσουν τον μηδενικό χαρακτήρα. Στη δραστηριότητα αυτή θα δοθούν μόνο οι συναρτήσεις που χρησιμοποιούνται για την απόδοση περιεχομένου σε συμβολοσειρές.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τρία τουλάχιστον παραδείγματα συμβολοσειρών στη γλώσσα C • να χρησιμοποιούν τις συναρτήσεις για την είσοδο περιεχομένου σε μια συμβολοσειρά • να δώσουν τρία τουλάχιστον παραδείγματα εισαγωγής μιας συμβολοσειράς με την συνάρτηση scanf()
Μαθησιακά Αντικείμενα	Αλφαριθμητικά Δείκτες και Αλφαριθμητικά Συμβολοσειρές
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 8 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S4_LA11LO109 και S4_LA11LO114.
Λέξεις Κλειδιά	Συνάρτηση gets(), συνάρτηση scanf(), τύπος string, χαρακτήρας NULL, δείκτης αλφαριθμητικού**,** πίνακας χαρακτήρων, πίνακας δεικτών χαρακτήρα

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA12LO119**
Τίτλος	**Αλφαριθμητικά**
Τίτλος δραστηριότητας	Συμβολοσειρές
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι συμβολοσειρές και πώς αυτές χρησιμοποιούνται στη γλώσσα C. Στη C δεν υπάρχει ο τύπος string, αλλά αυτός υλοποιείται ως πίνακας χαρακτήρων. Υπάρχουν διάφοροι τρόποι να ορίσουμε ένα string στη C, ανάλογα με το αν γνωρίζουμε το string πριν τη μεταγλώττιση ή όχι. Επίσης περιγράφονται οι συναρτήσεις που χρησιμοποιούνται για την απόδοση περιεχομένου σε συμβολοσειρές.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Συνάρτηση gets(), συνάρτηση scanf(), τύπος string, χαρακτήρας NULL
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τρία τουλάχιστον παραδείγματα συμβολοσειρών στη γλώσσα C • να χρησιμοποιούν τις συναρτήσεις για την είσοδο περιεχομένου σε μια συμβολοσειρά • να δώσουν τρία τουλάχιστον παραδείγματα εισαγωγής μιας συμβολοσειράς με τη συνάρτηση scanf()

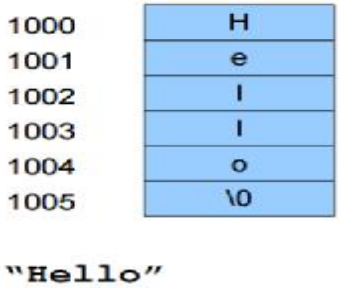
Όταν ο μεταγλωττιστής της C συναντά κάτι ανάμεσα σε διπλά εισαγωγικά " ", το θεωρεί σαν σταθερά συμβολοσειράς ή αλλιώς αλφαριθμητικό και οι χαρακτήρες μέσα στα εισαγωγικά αυτά μαζί με τον χαρακτήρα '\0' αποθηκεύονται σε γειτονικές θέσεις της μνήμης.

printf("Τρέξε, Μαρία! είπε η Βάσω.\n");

Το αποτέλεσμα θα είναι το εξής: Τρέξε, Μαρία! είπε η Βάσω.

Όλη η φράση μέσα στα εισαγωγικά δρα σαν δείκτης, εκεί που αποθηκεύεται η συμβολοσειρά, κάτι δηλαδή ανάλογο με το όνομα ενός πίνακα, που δρα σαν ένας δείκτης στη θέση του πίνακα.

Ένα αλφαριθμητικό ή συμβολοσειρά (string) αναπαρίσταται εσωτερικά ως ένας πίνακας χαρακτήρων που τερματίζεται από τον ειδικό χαρακτήρα NULL (‘\0’). Η χρήση του μηδενικού χαρακτήρα '\0' εξηγεί ότι πρόκειται για συμβολοσειρά και όχι για πίνακα χαρακτήρων. Και στις δύο μορφές, ο μεταγλωττιστής μετράει τους χαρακτήρες και δίνει στον πίνακα το αντίστοιχο μέγεθος.



Εικόνα 9.Πίνακας χαρακτήρων

Μπορώ να αποθηκεύσω το παραπάνω αλφαριθμητικό με τις εξής εντολές:

```
char carray[6];
carray[0]=’H’;
carray[1]=’e’;
carray[2]=’l’;
carray[3]=’l’;
carray[4]=’o’;
carray[5]=’\0’;
```

Μπορούμε να δώσουμε αρχικές τιμές σε μια συμβολοσειρά χαρακτήρα και να έχουμε το ίδιο αποτέλεσμα με τις εντολές:

```
char carray[]=’Hello’;
char carray[6]=’Hello’;
char carray[]={’H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};
char carray[6]={’H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};
```

Για το διάβασμα μιας συμβολοσειράς σ' ένα πρόγραμμα, πρέπει να κάνουμε δύο πράγματα: να δεσμεύσουμε το χώρο αποθήκευσης γι αυτή τη συμβολοσειρά και να χρησιμοποιήσουμε μια συνάρτηση εισόδου για το διάβασμά της. Ο πιο απλός τρόπος για να δηλώσουμε με σαφήνεια το μέγεθος ενός πίνακα είναι η εξής δήλωση:

```
char name[81];
```

Αφού έχει δημιουργηθεί χώρος στη μνήμη για τη συμβολοσειρά, μπορούμε πλέον να τη διαβάσουμε. Θα δούμε τις συναρτήσεις διαβάσματος συμβολοσειρών gets() και scanf().

Η συνάρτηση gets()

Η συνάρτηση αυτή διαβάζει χαρακτήρες μέχρι να συναντήσει το χαρακτήρα enter '\n'. Η συνάρτηση αυτή δέχεται όλους τους χαρακτήρες πριν από το χαρακτήρα '\n', προσθέτει τον

μηδενικό χαρακτήρα '\0' και δίνει τη συμβολοσειρά στο καλούν πρόγραμμα. Ο χαρακτήρας νέας γραμμής διαβάζεται και αγνοείται.

Ακολουθεί ένα απλό παράδειγμα χρήσης της συνάρτησης gets():

```
/* prog46.c - διάβασμα ενός ονόματος */
#include <stdio.h>
#define MAX 81
main()
{
    char name[MAX];     /* δέσμευση χώρου */
    printf("Γεια σου, ποιο είναι το όνομά σου ;\n");
    gets(name);
    printf("Γεια σου, %s.\n", name);
}
```

Η συνάρτηση scanf()

Η βασική διαφορά των δύο συναρτήσεων είναι στον τρόπο που αποφασίζουν ότι έχουν φθάσει στο τέλος της συμβολοσειράς. Μπορούμε να πούμε ότι η scanf() είναι περισσότερο μια συνάρτηση "απόκτησης λέξης" παρά "απόκτησης συμβολοσειράς".

Η συνάρτηση scanf() έχει δύο επιλογές. Ξεκινά πάντα με τον πρώτο μη-κενό χαρακτήρα που συναντάει. Αν χρησιμοποιούμε τον **προσδιοριστή %s,** η συμβολοσειρά φθάνει μέχρι τον επόμενο κενό χαρακτήρα χωρίς να τον περιλαμβάνει. Αν καθορίσουμε ένα εύρος πεδίου, π.χ. %10s, τότε η scanf() λαμβάνει υπόψη της μέχρι 10 χαρακτήρες ή μέχρι τον πρώτο κενό χαρακτήρα.

Όπως είδαμε και στα προηγούμενα, η τιμής επιστροφής της scanf() είναι μια ακέραια τιμή, που είναι ίση με το αριθμό των στοιχείων που διαβάστηκαν με επιτυχία ή με τον χαρακτήρα EOF αν συναντήσει το τέλος αρχείου.

Ακολουθεί ένα παράδειγμα:

```
/* prog47. c - χρήση της scanf() */
#include <stdio.h>
main()
{
    static char name1[11], name2[11];
    int count;
    printf("Παρακαλώ δώστε 2 ονόματα. \n");
    count = scanf("%5s %10s", name1, name2);
    printf("Διάβασα τα %d ονόματα %s και %s. \n", count, name1, name2);
}
```

Το αποτέλεσμα θα είναι:

```
    Παρακαλώ δώστε 2 ονόματα.
```

Τάκης Αντώνης

Διάβασα τα 2 ονόματα Τάκης και Αντώνης.

ΠΡΟΣΟΧΗ:

- scanf (“%s”, name):

Δεν χρησιμοποιείτε το &, γιατί το name είναι πίνακας.

Θυμηθείτε ότι σε άλλους τύπους δεδομένων χρησιμοποιείται π.χ.

scanf(“%d”, &a);

- Αν έχουμε να δώσουμε μόνο κείμενο από το πληκτρολόγιο, τότε είναι καλύτερα να χρησιμοποιήσουμε τη συνάρτηση gets(), ενώ η scanf() συνιστάται για την είσοδο μικτών τύπων σε μια τυποποιημένη μορφή, όπως π.χ. αν κάθε γραμμή εισόδου περιέχει το όνομα ενός εργαλείου, τον κωδικό του αριθμού και την τιμή του.
- Η διαφορά char και string

char c = ‘H’; Ο χαρακτήρας H βρίσκεται κάπου στην μνήμη

?	?	?	H	?	?	?

char c[ ]=“H”; Το string H βρίσκεται κάπου στην μνήμη.

H	\0			
1	2	3	4	5

char c="H"; ΛΑΘΟΣ στην μεταγλώττιση

Άσκηση αυτοαξιολόγησης - S4_LA12LO120

Να χαρακτηρίσετε τις παρακάτω δηλώσεις της συμβολοσειράς “Hello” σαν λάθος/σωστό:

1. char msg[ ]={‘H’,’e’,’l’,’l’,’o’,’\0’};
2. char msg[40]=“Hello”
3. char msg[6]={‘H’,’e’,’l’,’l’,’o’,’\0’};
4. char msg[5]={'H','e','l','l','o','\0'}; Δεν δεσμεύουμε αρκετό χώρο
5. char msg[ ]={‘H’,’e’,’l’,’l’,’o’}; Πίνακας χαρακτήρων που δεν τελειώνει σε '\0'. Επομένως δεν είναι string
6. char msg[6];

   msg[0] = 'H';

   msg[1] = 'e';

   msg[2] = 'l';

   msg[3] = 'l';

   msg[4] = 'o';

   msg[5] = '\0';

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA12LO121**
Τίτλος	**Δείκτες και Αλφαριθμητικά**
Τίτλος δραστηριότητας	Συμβολοσειρές
Περιγραφή	Στο συγκεκριμένο ΜΑ, παρουσιάζονται οι δείκτες αλφαριθμητικών. Αρχικά δηλώνουμε το δείκτη (char * pmsg;) και μετά αναθέτουμε τιμή σε αυτόν με χρήση διπλών εισαγωγικών (pmsg = “Today is Thursday”;).
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Δείκτης αλφαριθμητικού**,** πίνακας χαρακτήρων, πίνακας δεικτών χαρακτήρα
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δίνουν τουλάχιστον τρία παραδείγματα δεικτών σε αλφαριθμητικά • να κατανοούν τη διαφορά στον τρόπο δήλωσης αλφαριθμητικών με δείκτες και χωρίς δείκτες

Δήλωση δείκτη αλφαριθμητικού

Αρχικά δηλώνουμε το δείκτη που αντιστοιχεί σε αλφαριθμητικό

char *<όνομα δείκτη>;

Παράδειγμα: char *str_ptr;

Ανάθεση τιμής σε δείκτη αλφαριθμητικού

Κατόπιν αναθέτουμε τιμή στο δείκτη

<όνομα δείκτη> = <αλφαριθμητικό>;

Παράδειγμα: str_ptr = “Hello”;

Μην ξεχνάτε ότι το αλφαριθμητικό είναι ένας πίνακας χαρακτήρων που τερματίζει με το χαρακτήρα '\0'.

Τα αλφαριθμητικά στη γλώσσα χρησιμοποιούνται ως αφηρημένος τύπος δεδομένων.

Η δήλωση char *s; είναι ισοδύναμη με τη δήλωση ενός string με όνομα s.

Δείκτες και μεταβλητές πίνακα για αλφαριθμητικά

Το παρακάτω τμήμα κώδικα είναι έγκυρο

char *carray;

carray=”A fool on the hill”;

το παρακάτω τμήμα κώδικα δεν θα μεταγλωττιστεί

char carray[32];

carray=”Another Green World”;

Θα προκύψει ένα μήνυμα της μορφής “Lvalue required….”

Άσκηση αυτοαξιολόγησης - S4_LA12LO122

1	Διεύθυνση του πίνακα χαρακτήρων year	Διεύθυνση της μεταβλητής year_prt	1821	1

Με δεδομένες τις δηλώσεις char *year_prt; year_ptr="1821"; και char year[]="1821";

συμπληρώστε τις παραπάνω αντιστοιχήσεις βάζοντας τα περιεχόμενα στη σωστή τους θέση.

Μαθησιακό Αντικείμενο	
Όνομα	**S4_LA12LO123**
Τίτλος	**Πίνακες δεικτών**
Τίτλος δραστηριότητας	Συμβολοσειρές
Περιγραφή	Η συμβολοσειρά είναι ένας ειδικός τύπος πίνακα με δεδομένα του τύπου char και με τερματισμό μέσω του μηδενικού χαρακτήρα. Εφόσον έχουμε δείκτες που δείχνουν κάθε συμβολοσειρά όπως ακριβώς και στους πίνακες και αφού οι δείκτες είναι μεταβλητές, μπορούν και αυτοί να αποθηκεύονται σε πίνακες. Πολλές φορές χρησιμοποιούμε τους πίνακες δεικτών σε αλφαριθμητικά για να κάνουμε ταξινόμηση κατά αλφαβητική σειρά σε ολόκληρες σειρές με πιο εύκολο τρόπο.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Δείκτες, πίνακες δεικτών, δείκτες σε αλφαριθμητικά, πίνακας αλφαριθμητικών
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να κατανοήσουν την έννοια των δεικτών που δείχνουν σε συμβολοσειρές • να κατανοήσουν τη διαφορά ανάμεσα σε ένα πίνακα δεικτών και σε ένα πίνακα αλφαριθμητικών

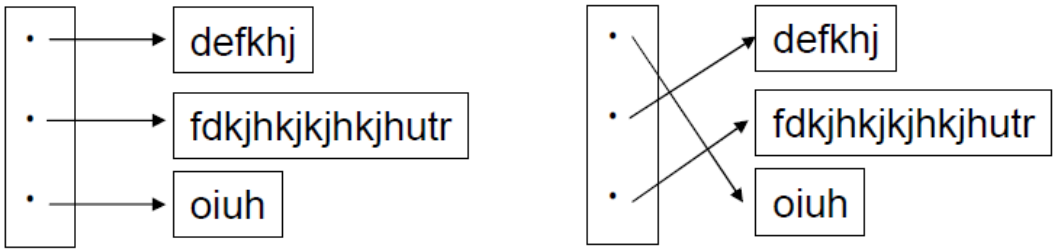
Αφού οι δείκτες είναι μεταβλητές, μπορούν και αυτοί να αποθηκεύονται σε πίνακες

char str[5]; (Αλφαριθμητικό μήκους 5 χαρακτήρων)

char *ptr[5]; (Πίνακας 5 δεικτών σε αλφαριθμητικά)

Έτσι υπάρχει ευκολία στον χειρισμό αλφαριθμητικών διαφορετικού μεγέθους.

Παράδειγμα



- Τα αλφαριθμητικά στα οποία δείχνουν οι δείκτες μπορεί να είναι διαφορετικού μήκους
- Αν θέλουμε να αλλάξουμε τη θέση των αλφαριθμητικών απλά αλλάζουμε τη θέση των δεικτών

Προσοχή στις διαφορές

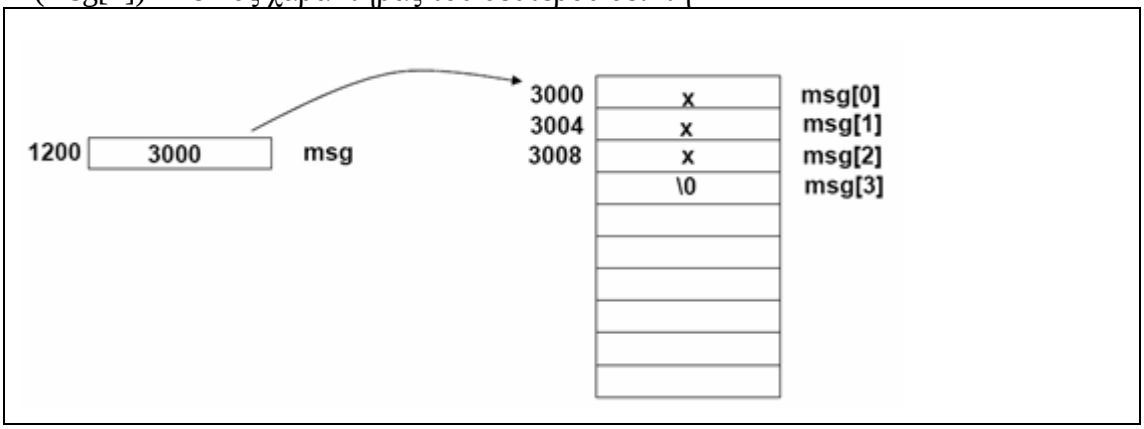
char msg[ ] = “Today is Thursday”; (πίνακας χαρακτήρων)

char *pmsg = “Today is Thursday”; (δείκτης σε πίνακα, μπορεί να πάρει άλλη τιμή)

char *msg[18]; (πίνακας δεικτών χαρακτήρα)

msg[1] -> ο 2ος δείκτης του παραπάνω πίνακα

*(msg[1]) -> ο 1ος χαρακτήρας του δεύτερου δείκτη



Εικόνα 10. Δείκτες χαρακτήρων

Παραδείγματα

```
char *amsg = "xxx";
char msg[] = "xxx";
msg = "yyyy"; /* δεν επιτρέπεται */
amsg = "yyyy"; /* bigger, ok */
```

Αρχικοποίηση πίνακα δεικτών

Πίνακας δεικτών:

char *name[] = {“Λάθος μήνας”, “Ιαν”, “Φεβ”, “Μάρ” };

Δέσμευση όσων θέσεων μνήμης χρειάζονται ανάλογα με τον τον αριθμό των δεικτών.

Πίνακας αλφαριθμητικών:

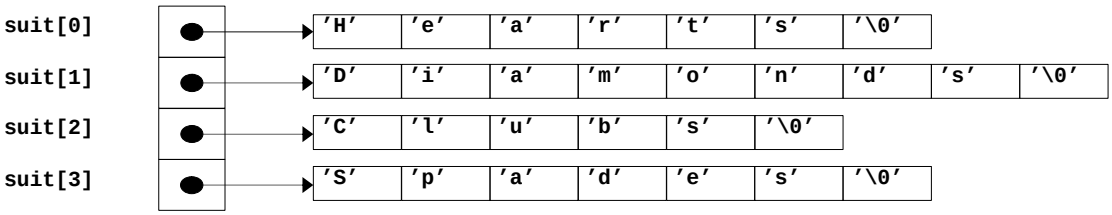
char aname[][15] = {"Λάθος μήνας", "Ιαν", "Φεβ", "Μάρ"};

Δέσμευση 15*4 θέσεων μνήμης για char

Άσκηση αυτοαξιολόγησης - S4_LA12LO124

Όπως αναφέραμε, οι πίνακες μπορεί να περιέχουν δείκτες. Με βάση την παρακάτω εικόνα να δηλώσετε και να αρχικοποιήσετε ταυτόχρονα τον πίνακα **suit. Ισχύει ότι:**

- Κάθε στοιχείο του **suit** δείχνει σε **char *** (a string)
- Ο πίνακας δεν αποθηκεύει strings, μόνο δείκτες σε strings
- Ο πίνακας **suit** έχει σταθερό μέγεθος, αλλά τα strings μπορεί να είναι οποιουδήποτε μεγέθους



3.5 Ενότητα 5 - Προτάσεις ελέγχου ροής

Τίτλος Ενότητας	***S5*** **-** Προτάσεις ελέγχου ροής
Τίτλος Μαθήματος	Διαδικασιακός προγραμματισμός - Γλώσσα προγραμματισμού C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστεί ένα από τα σημαντικά έργα του προγραμματιστή στην προστακτική μορφή προγραμματισμού που είναι ο καθορισμός της ροής ελέγχου του προγράμματος, δηλαδή ο καθορισμός της σειράς εκτέλεσης των εντολών ή των προτάσεων από τις οποίες απαρτίζεται το πρόγραμμα. Σκοπός της ενότητας είναι να εισάγει την έννοια της ροής ελέγχου σε ένα πρόγραμμα, δίνοντας ταυτόχρονα τις βασικές κατηγορίες προτάσεων που επιτρέπουν στον προγραμματιστή να διαμορφώσει τη ροή εκτέλεσης ανάλογα με τις απαιτήσεις της κάθε εφαρμογής.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να γνωρίσουν τις βασικές δομές του δομημένου προγραμματισμού • να κατανοούν την χρήση των εντολών ροής ελέγχου καθώς και τις διαφορές τους • να φτιάχνουν σύντομα, κατανοητά και τελικά αποδοτικότερα προγράμματα
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να περιγράψουν τις βασικές κατηγορίες προτάσεων διαμόρφωσης της ροής ελέγχου σε ένα πρόγραμμα, • να αναγνωρίσουν ποια από τις κατηγορίες προτάσεων ελέγχου ροής θα πρέπει να χρησιμοποιήσουν σε δεδομένη περίπτωση, • να δώσουν τρεις τουλάχιστον προτάσεις για επανάληψη, • να δώσουν δύο τουλάχιστον προτάσεις για υπό συνθήκη διακλάδωση, • να γράψουν τη σύνταξη των προτάσεων ελέγχου ροής της C, • να εξηγήσουν τη διαφορά μεταξύ βρόχου επανάληψης συνθήκης εισόδου και αντίστοιχου συνθήκης εξόδου, • να διαμορφώσουν τη ροή ελέγχου του προγράμματός τους ώστε να ανταποκρίνεται στις απαιτήσεις που έχουν από αυτό.

Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: • Δομημένος προγραμματισμός • Προτάσεις ελέγχου ροής - διακλάδωση • Προτάσεις ελέγχου ροής - επανάληψη
Εμπλεκόμενοι Ρόλοι	Κατά τη διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας και των εκπαιδευομένων.
Αξιολόγηση	Η ενότητα του μαθήματος αυτού χρησιμοποιεί σε κάθε δραστηριότητα και σε κάθε μαθησιακό αντικείμενο ασκήσεις αυτοαξιολόγησης. Τέτοιες ασκήσεις μπορεί να είναι ερωτήσεις επίλυσης προβλημάτων ή ερωτήσεις αντικειμενικού τύπου. Για τη διεκπεραίωση αυτών των ασκήσεων μπορεί να χρησιμοποιηθούν διάφορα εργαλεία λογισμικού, όπως προγράμματα σχεδιασμού ερωτήσεων συμπλήρωσης κενού, σταυρόλεξα ή λογισμικό σχεδιασμού ανάπτυξης προγραμμάτων στη γλώσσα C.
Συνολικός χρόνος Ενότητας	1 εβδομάδα

3.5.1 Δραστηριότητα 1 - Δομημένος προγραμματισμός

Τίτλος Δραστηριότητας	***S5_LA13*** **-** Δομημένος προγραμματισμός
Τίτλος Ενότητας	Προτάσεις ελέγχου ροής
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν επαναληπτική θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Στη δραστηριότητα αυτή περιγράφεται ο δομημένος προγραμματισμός, δηλαδή μια τεχνική σχεδίασης προγράμματος που χρησιμοποιεί τις αρχές του ιεραρχικού και του τμηματικού προγραμματισμού. Ορίζονται οι αρχές του ιεραρχικού και του τμηματικού προγραμματισμού και εξετάζεται γενικά και ανεξαρτήτως γλώσσας προγραμματισμού το θέμα της διαμόρφωσης της ροής ελέγχου του προγράμματος με τις βασικές συνιστώσες του δομημένου προγραμματισμού.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν τα είδη προγραμματισμού και να περιγράφουν τα βασικά χαρακτηριστικά των τεχνικών που χρησιμοποιούνται, • να διατυπώνουν τα πλεονεκτήματα του δομημένου προγραμματισμού, • να δίνουν παραδείγματα δομημένου προγραμματισμού.
Μαθησιακά Αντικείμενα	Δομημένος προγραμματισμός
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 1 μαθησιακό αντικείμενο που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακό αντικείμενο S5_LA13LO125**.**
Λέξεις Κλειδιά	Δομημένος προγραμματισμός, ιεραρχικός προγραμματισμός, τμηματικός προγραμματισμός, δομή επιλογής, δομή επανάληψης, ακολουθιακή δομή

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA13LO125**
Τίτλος	**Δομημένος προγραμματισμός**
Τίτλος δραστηριότητας	Δομημένος προγραμματισμός
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι έννοιες δομημένος, ιεραρχικός και τμηματικός προγραμματισμός. Στο δομημένο προγραμματισμό, το πρόγραμμα αποτελείται από ανεξάρτητα τμήματα με βάση ένα προκαθορισμένο σχέδιο. Ο δομημένος προγραμματισμός χρησιμοποιεί τις βασικές αλγοριθμικές δομές της ακολουθίας, της επιλογής και της επανάληψης. Τέλος περιγράφονται τα πλεονεκτήματα της συγκεκριμένης τεχνικής ανάπτυξης προγραμμάτων.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Δομημένος προγραμματισμός, ιεραρχικός προγραμματισμός, τμηματικός προγραμματισμός, δομή επιλογής, δομή επανάληψης, ακολουθιακή δομή
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν τα είδη προγραμματισμού όπως ο τμηματικός, ο ιεραρχικός και ο δομημένος προγραμματισμός, • να διατυπώνουν τα πλεονεκτήματα του δομημένου προγραμματισμού, • να δίνουν παραδείγματα δομημένου προγραμματισμού, • να περιγράφουν τα βασικά χαρακτηριστικά των τεχνικών που χρησιμοποιούνται στο δομημένο προγραμματισμό.

Ο πλέον συνηθισμένος τρόπος εκτέλεσης των εντολών που συναντάμε σε κάθε προστακτική γλώσσα είναι ο ακολουθιακός τρόπος. Δύο ή περισσότερες εντολές, οι οποίες γράφονται η μία μετά την άλλη, εκτελούνται διαδοχικά. Για να επιτευχθεί οποιαδήποτε διαφοροποίηση από την ακολουθιακή εκτέλεση χρησιμοποιούνται ειδικές τεχνικές. Οι τεχνικές που χρησιμοποιούνται για να επιτευχθεί η επιθυμητή ροή ελέγχου ενός προγράμματος

διασφαλίζουν ταυτόχρονα τη δόμηση του προγράμματος έτσι ώστε: «Η δομή του πηγαίου κώδικα να μας βοηθά να κατανοήσουμε τι κάνει το πρόγραμμα». Μια γλώσσα που έχει τεχνικές ελέγχου που υποστηρίζουν αυτή την αρχή λέγεται γλώσσα δομημένου προγραμματισμού, ο δε προγραμματισμός με τέτοιες κατασκευές λέγεται δομημένος προγραμματισμός (structured programming). Ο αναγνώστης δομημένου πηγαίου κώδικα μπορεί εύκολα να κατανοήσει τι συμβαίνει όταν το πρόγραμμα εκτελείται.

Δομημένος προγραμματισμός, λοιπόν, καλείται η τεχνική σχεδίασης αλγορίθμου και του αντίστοιχου προγράμματος, η οποία χρησιμοποιεί τις αρχές του ιεραρχικού και του τμηματικού προγραμματισμού. Άρα θα πρέπει να ορίσουμε και τις έννοιες του τμηματικού και ιεραρχικού προγραμματισμού.

Τμηματικός προγραμματισμός ονομάζεται ο προγραμματισμός που ακολουθεί την ιεραρχική σχεδίαση και είναι η υλοποίηση του προγράμματος μέσα από ανεξάρτητες λογικές ενότητες - τμήματα. Τα μεμονωμένα τμήματα πρέπει να είναι όσο το δυνατόν μικρότερα ώστε να είναι εύκολο να διορθωθούν.

Ιεραρχικός προγραμματισμός καλείται η τεχνική σχεδίασης προγραμμάτων κατά την οποία το πρόβλημα διασπάται σε μια σειρά από απλούστερα προβλήματα, τα οποία, όταν επιλυθούν, οδηγούν στην επίλυση του αρχικού προβλήματος. Η σχεδίαση του προγράμματος πρέπει να προχωρά από πάνω προς τα κάτω. Στα ανώτερα επίπεδα ξεκινάμε από τα γενικά και προχωρούμε σε κατώτερα επίπεδα προσθέτοντας λεπτομέρειες. Η ανάλυση σε κατώτερα επίπεδα σταματά όταν η επεξεργασία περιέχει τόσες λεπτομέρειες ώστε να μπορεί να κωδικοποιηθεί σε μια γλώσσα προγραμματισμού.

Στο δομημένο προγραμματισμό, ο αλγόριθμος αποτελείται από ανεξάρτητα τμήματα με βάση ένα προκαθορισμένο σχέδιο. Έτσι, με τον ίδιο τρόπο θα αναπτυχθεί αμέσως μετά και η κωδικοποίηση του αλγορίθμου σε γλώσσα προγραμματισμού δημιουργώντας τελικά το πρόγραμμα. Ο δομημένος προγραμματισμός χρησιμοποιεί μόνο τις βασικές αλγοριθμικές δομές της ακολουθίας, της επιλογής και της επανάληψης, ενώ αποφεύγει τη χρήση της εντολής GO TO. Ένα δομημένο πρόγραμμα διαβάζεται και συντηρείται εύκολα.

Τα κύρια πλεονεκτήματα του δομημένου προγραμματισμού είναι:

- Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα.
- Άμεση μεταφορά των αλγορίθμων σε προγράμματα.
- Ευκολία και ταχύτητα στην κωδικοποίηση.
- Καλύτερη ποιότητα προγραμμάτων.
- Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
- Ευκολία στις διορθώσεις και τη συντήρηση.
- Τεκμηρίωση που περιέχεται σχεδόν εξ ολοκλήρου στο ίδιο το πρόγραμμα.
- Δημιουργία απλούστερων προγραμμάτων.

Ο δομημένος προγραμματισμός χρησιμοποιεί, όπως αναφέραμε, τις βασικές αλγοριθμικές δομές της επιλογής ή της υπό συνθήκη διακλάδωσης (conditional branching) και της επανάληψης (looping), έτσι ώστε να γίνεται διαμόρφωση της ροής ελέγχου στο πρόγραμμα. Σύμφωνα με τη δομή επανάληψης, μία εντολή ή ένα σύνολο εντολών εκτελείται επανειλημμένα, ενώ σύμφωνα με τη δομή της υπό συνθήκη διακλάδωσης, διαφορετικές εντολές ή ομάδες εντολών εκτελούνται ανάλογα με την τιμή μιας έκφρασης.

Η επίδραση της δόμησης στην αποδοτικότητα (efficiency) του προγράμματος είναι μηδαμινή. Προσεκτικά σχεδιασμένος δομημένος κώδικας είναι εξίσου αποδοτικός όσο ο αντίστοιχος μη δομημένος. Αλλά, είναι ευκολότερο να βελτιώσεις την αποδοτικότητα ενός δομημένου πηγαίου κώδικα από ότι ενός μη δομημένου.

Ερωτήσεις αντικειμενικού τύπου - S5_LA13LO126

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες Λανθασμένες;

1. Ιεραρχικός προγραμματισμός καλείται η τεχνική σχεδίασης προγραμμάτων κατά την οποία το πρόβλημα διασπάται σε μια σειρά από απλούστερα προβλήματα, τα οποία, όταν επιλυθούν, οδηγούν στην επίλυση του αρχικού προβλήματος.
2. Σύμφωνα με την κατασκευή επανάληψης, διαφορετικές προτάσεις ή ομάδες προτάσεων εκτελούνται ανάλογα με την τιμή μιας έκφρασης.
3. Ο αναγνώστης δομημένου πηγαίου κώδικα δεν μπορεί εύκολα να κατανοήσει τι συμβαίνει όταν το πρόγραμμα εκτελείται.
4. Ένα δομημένο πρόγραμμα διαβάζεται και συντηρείται εύκολα.
5. Δομημένος προγραμματισμός καλείται η τεχνική σχεδίασης προγράμματος, η οποία χρησιμοποιεί τις αρχές του λογικού και του προστακτικού προγραμματισμού.
6. Ο πλέον συνηθισμένος τρόπος εκτέλεσης των εντολών που συναντάμε σε κάθε προστακτική γλώσσα είναι ο ακολουθιακός τρόπος.
7. Ο δομημένος προγραμματισμός καταλήγει στη χρήση της εντολής GO TO.
8. Αποτέλεσμα του δομημένου προγραμματισμού είναι η δημιουργία απλούστερων προγραμμάτων.

Άσκηση αυτοαξιολόγησης - S5_LA13LO127

Να συμπληρώσετε τα κενά με τις λέξεις που λείπουν:

α. Ο δομημένος προγραμματισμός προϋποθέτει την …… του προγράμματος.

β. Ο δομημένος προγραμματισμός αποτελείται από ……. τμήματα με βάση ένα προκαθορισμένο σχέδιο.

γ. Ένα δομημένο πρόγραμμα είναι πολύ ……. ……. στην κατανόηση από προγράμματα που είναι γραμμένα με διαφορετικό τρόπο.

δ. Ένα δομημένο πρόγραμμα διαβάζεται εύκολα και, επειδή αποτελείται από σαφώς καθορισμένα και ……. τμήματα, έχει εύκολη και τη …… του.

ε. Οι βασικές αλγοριθμικές δομές δεν είναι αρκετές για την κωδικοποίηση ενός …… προγράμματος, αλλά πρέπει να αναπτυχθούν χρησιμοποιώντας συγκεκριμένες ……..

3.5.2 Δραστηριότητα 2 - Προτάσεις ελέγχου ροής - διακλάδωση

Τίτλος Δραστηριότητας	**S5_LA14 -** Προτάσεις ελέγχου ροής-διακλάδωση
Τίτλος Ενότητας	Προτάσεις ελέγχου ροής
Εκπαιδευτική Στρατηγική	*Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.*
Περιγραφή	Στη δραστηριότητα αυτή περιγράφουμε τις βασικές κατηγορίες προτάσεων που θα επιτρέψουν τη διαμόρφωση, ανάλογα με την εφαρμογή, της ροής ελέγχου του αντίστοιχου προγράμματος, διαφοροποιώντας την από την κλασσική ακολουθιακή εκτέλεση. Οι προτάσεις διακλάδωσης χωρίζονται στις προτάσεις διακλάδωσης υπό συνθήκη, με χαρακτηριστική την πρόταση if, και στις προτάσεις διακλάδωσης χωρίς συνθήκη με χαρακτηριστική την πρόταση switch.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει οι εκπαιδευόμενοι τη δραστηριότητα αυτή θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα χρήσης για κάθε δομή απλής επιλογής, σύνθετης επιλογής και πολλαπλής επιλογής, • να δώσουν παραδείγματα χρήσης εμφωλευμένης επιλογής, • να αναγνωρίζουν τους κανόνες που διέπουν τη λειτουργία της εντολής switch και να δίνουν παραδείγματα χρήσης της εντολής αυτής, • να ξεχωρίζουν πότε γίνεται χρήση της εντολής switch και πότε γίνεται χρήση πολλαπλής επιλογής if - else if.
Μαθησιακά Αντικείμενα	Προτάσεις διακλάδωσης υπό συνθήκη Προτάσεις διακλάδωσης χωρίς συνθήκη
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 2 μαθησιακά αντικείμενα - ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S5_LA14LO128 και S5_LA14LO131.
Λέξεις Κλειδιά	πρόταση if, πρόταση switch, δομή επιλογής, σύνθετη επιλογή, πολλαπλή επιλογή, απλή επιλογή

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA14LO128**
Τίτλος	Προτάσεις διακλάδωσης υπό συνθήκη
Τίτλος Δραστηριότητας	Προτάσεις ελέγχου ροής - διακλάδωση
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι προτάσεις διακλάδωσης υπό συνθήκη. Η εντολή if καλείται εντολή διακλάδωσης, γιατί δημιουργεί ένα σημείο διασταύρωσης, από το οποίο το πρόγραμμα έχει να διαλέξει μεταξύ δύο δυνατών κατευθύνσεων που μπορεί να ακολουθήσει. Η C δίνει τη δυνατότητα επιλογής μεταξύ δύο μπλοκ εντολών με τη χρήση της δομής if else. Αν η έκφραση είναι αληθής, τότε εκτελείται η εντολή που ακολουθεί την if, αλλιώς εκτελείται η εντολή που ακολουθεί την else. Επίσης, δίνει τη δυνατότητα επιλογής μεταξύ πολλαπλών μπλοκ εντολών με τη χρήση της δομής if - else if.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Δομή επιλογής, απλή επιλογή, σύνθετη επιλογή, πολλαπλή επιλογή, εμφωλευμένη επιλογή
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει οι εκπαιδευόμενοι την ενότητα αυτή θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα χρήσης της δομής απλής επιλογής, • να δώσουν τουλάχιστον δύο παραδείγματα χρήσης της δομής σύνθετης επιλογής, • να δώσουν τουλάχιστον δύο παραδείγματα χρήσης της δομής πολλαπλής επιλογής, • να δώσουν τουλάχιστον δύο παραδείγματα χρήσης της εμφωλευμένης επιλογής.

Μια πρόταση διακλάδωσης υπό συνθήκη, περιέχει έναν αριθμό υποπροτάσεων, από τις οποίες επιλέγεται και εκτελείται μόνο μία. Η πρόταση if, την οποία συναντάμε σε όλες τις προστακτικές γλώσσες είναι η πλέον γνωστή πρόταση αυτής της κατηγορίας και έχει τη μορφή:

if (<ΣΥΝΘΗΚΗ>)

Ενότητα -Α

else

Ενότητα-Β

Με βάση τον παραπάνω γενικό τύπο της σύνθετης επιλογής, η <ΣΥΝΘΗΚΗ> μπορεί να είναι λογική έκφραση, έκφραση σύγκρισης, αποτέλεσμα κάποιας πράξης, είτε ακόμα και κάποια μεταβλητή. Η έκφραση, που τις περισσότερες φορές είναι σύγκριση, υπολογίζεται και αν η τιμή της είναι αληθής (κάθε τιμή διάφορη του 0 για τη C είναι αληθής) εκτελείται η Ενότητα-Α, αλλιώς εκτελείται η Ενότητα-Β. Η Ενότητα-Α και η Ενότητα-Β μπορεί να περιλαμβάνουν μία εντολή ή πολλές εντολές (block) που περικλείονται σε άγκιστρα {…}. Οι εντολές (ή η εντολή) της Ενότητας-Α εκτελούνται αν η <ΣΥΝΘΗΚΗ> είναι αληθής (όχι 0), ενώ οι εντολές (ή η εντολή) της Ενότητας-Β εκτελείται αν η <ΣΥΝΘΗΚΗ> είναι ψευδής (δηλαδή 0). Το else είναι προαιρετικό και όταν υφίσταται αναφέρεται στο πλησιέστερο πριν από αυτό if που δεν έχει else. Κάθε ενότητα είναι δυνατόν να περικλείει και άλλες if - else ενότητες (blocks) εντολών.

Παράδειγμα σύνθετης επιλογής

Να γραφεί πρόγραμμα που να ζητά από το χρήστη ένα μη αρνητικό αριθμό και να τυπώνει την τετραγωνική του ρίζα.

Η περιγραφή της διεργασίας που πρέπει να ακολουθήσει ο υπολογιστής και ο αντίστοιχος κώδικας σε C έχουν την παρακάτω μορφή:

Περιγραφή διεργασίας	**Κώδικας σε C**

	#include <stdio.h>
	#include <math.h> /*βιβλιοθήκη συναρτήσεων*/
	main()
	{
	double num;
	printf(“ Δώσε ένα θετικό αριθμό:);
	scanf(“%lf” &num);
	if(num<0)
	printf("Λάθος είσοδος: Αρνητικός αριθμός \n");
	else
	printf("Η τετραγωνική ρίζα του %lf είναι %f\n", sqrt(num));
	exit(0);
	}

πάρε τον αριθμό εάν ο αριθμός είναι αρνητικός τύπωσε μήνυμα λάθους αλλιώς τύπωσε την τετραγωνική ρίζα του αριθμού τερμάτισε	

Μια απλοποιημένη παραλλαγή της εντολής if έχει τη μορφή:

if συνθήκη then εντολή;

Με βάση τον παραπάνω γενικό τύπο της απλής επιλογής, υπολογίζεται η λογική έκφραση και, αν η τιμή της είναι αληθής, εκτελείται η εντολή, αλλιώς ο έλεγχος μεταφέρεται στην εντολή που είναι μετά την if.

Παράδειγμα απλής επιλογής

```
if (a >= b) max = a;
```

Ευρεία χρήση στην πρόταση if βρίσκουν οι τελεστές σύγκρισης, δημιουργώντας εκφράσεις όπως οι παρακάτω:

Α) `if ((num > 10) && (num < 14))`

Β) `if ((ch == '\n') || (ch =='\t'))`

Γ) `if (!found)`

Δ) `if ((ch > 'a') && (ch < 'z'))`

Σε μια πιο σύνθετη μορφή της, η πρόταση if επιτρέπει επιλογή από μεγαλύτερο (συνήθως απεριόριστο) αριθμό προτάσεων, με την ένθεση διαδοχικών προτάσεων συνθήκης. Στην περίπτωση αυτή, το σχήμα που βελτιώνει την αναγνωσιμότητα του κώδικα έχει την παρακάτω μορφή:

if συνθήκη 1 then {εντολές 1}

else if συνθήκη 2 then {εντολές 2}

else if συνθήκη 3 then {εντολές 3}

:

else if συνθήκη n then {εντολές n}

else {εντολές k}

Οι λογικές εκφράσεις υπολογίζονται σειριακά και η πρώτη που θα δώσει αληθή τιμή οδηγεί στην εκτέλεση της αντίστοιχης πρότασης.

Αν, για παράδειγμα, η συνθήκη 1 και συνθήκη 2 είναι ψευδείς και η συνθήκη 3 δώσει αληθή τιμή, τότε εκτελούνται οι εντολές 3. Αν καμία από τις λογικές εκφράσεις δεν δώσει αληθή τιμή εκτελούνται οι εντολές k.

Παράδειγμα πολλαπλής επιλογής

```
int n;
n = GetInteger();
if (n > 0) { printf(“Positive”); }
else if (n == 0) { printf(“Zero”); }
else { printf(“Negative”); }
printf(“\n”);
```

Παράδειγμα εμφωλευμένης επιλογής

Τα παρακάτω δύο προγράμματα εκτυπώνουν τον μεγαλύτερο από τρείς αριθμούς με τη χρήση της εντολής if. Παρατηρήστε σε τι διαφέρουν τα προγράμματα Α και Β.

Πρόγραμμα α	Πρόγραμμα β

int a, b, c;
.....
if (a>b)
  if (a>c)
  printf ("max is a: %d", a);
  else
  printf ("max is c: %d", c);
else
  if (b>c)
  printf ("max is b: %d", b);
  else
  printf("max is c: %d", c);
	int a, b, c;
	.....
	if (a>b && a>c)
	  printf ("max is a: %d", a);
	if (b>a && b>c)
	  printf ("max is b: %d", b);
	if (c>a && c>b)
	  printf("max is c: %d", c);

Ερωτήσεις αντικειμενικού τύπου - S5_LA14LO129

Ποιες από τις παρακάτω προτάσεις είναι Σωστές και ποιες είναι Λανθασμένες;

1. Χρησιμοποιούμε την εντολή if όταν θέλουμε μια ομάδα εντολών να εκτελεστεί πολλές φορές.
2. Η εντολή if περιλαμβάνει τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής).
3. Στη δομή απλής επιλογής, η ομάδα εντολών εντός της δομής εκτελείται όταν η συνθήκη είναι αληθής.
4. Στην πολλαπλή επιλογή κάθε περίπτωση αντιστοιχεί σε διαφορετική τιμή της συνθήκης.
5. Στην εντολή if υπάρχει περίπτωση κάποιες εντολές να μην εκτελεστούν ποτέ.
6. Όταν πρέπει να εκτελεστούν κάποιες εντολές υπό κάποια συνθήκη χρησιμοποιείται η δομή ακολουθίας.
7. Ευρεία χρήση στην πρόταση if βρίσκουν οι τελεστές σύγκρισης δημιουργώντας λογικές εκφράσεις.
8. Δεν μπορούμε να χρησιμοποιήσουμε μια εντολή if μέσα σε μια άλλη εντολή if.
9. Όταν έχουμε παραπάνω από δύο συνθήκες που πρέπει να εξεταστούν για να εκτελεστεί μια ομάδα εντολών τότε έχουμε την απλοποιημένη παραλλαγή της εντολής if.
10. Η εντολή if αποτελεί μια πρόταση διακλάδωσης χωρίς συνθήκη.

Άσκηση αυτοαξιολόγησης - S5_LA14LO130

Στο παρακάτω πρόγραμμα να συμπληρώσετε τη συνθήκη και τις εντολές εκτύπωσης έτσι ώστε να τυπώνεται ο μεγαλύτερος από τους δύο αριθμούς που εισάγονται από το πληκτρολόγιο.

```
#include <stdio.h>
int main()
{
    int a, b;
    printf("dwse ta a kai to b\n");
    scanf("%d, %d", &a, &b);
    if ……..
        ……..
    ……
    return 0;
}
```

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA14LO131**
Τίτλος	Προτάσεις διακλάδωσης χωρίς συνθήκη
Τίτλος Δραστηριότητας	Προτάσεις ελέγχου ροής - διακλάδωση
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφεται η γενικευμένη μορφή της εντολής switch, σύμφωνα με την οποία ελέγχεται η τιμή μιας έκφρασης και από μία ομάδα εντολών εκτελείται αυτή της οποίας προηγείται μια ετικέτα που ταιριάζει με την τιμή της έκφρασης. Συνήθως, η τελευταία από την ομάδα εντολή έχει την ετικέτα default και είναι αυτή που εκτελείται στην περίπτωση που η τιμή της έκφρασης δεν ταιριάζει με καμία από τις υπάρχουσες ετικέτες.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Εντολή switch, εντολές break, exit ή return
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει οι εκπαιδευόμενοι την ενότητα αυτή θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα με χρήση της εντολής switch, • να αναγνωρίζουν τους κανόνες που διέπουν τη λειτουργία της εντολής switch, • να ξεχωρίζουν πότε γίνεται χρήση της εντολής switch και πότε γίνεται χρήση πολλαπλής επιλογής if - else if.

Μια πρόταση διακλάδωσης χωρίς συνθήκη περιέχει έναν αριθμό υποπροτάσεων, από τις οποίες επιλέγεται και εκτελείται μόνο μία. Η πρόταση switch επιτρέπει τον προσδιορισμό απεριόριστου αριθμού διαδρομών ανάλογα με την τιμή μιας έκφρασης.

Η γενικευμένη μορφή της εντολής είναι:

```
switch (έκφραση)
{
    case σταθερά 1:
        εντολή 1;
```

```
        break;
    case σταθερά 2:
        εντολή 2;
        break;
    default:
        εντολή N;
        break;
}
```

Υπολογίζεται η έκφραση και η τιμή της συγκρίνεται διαδοχικά με τις σταθερές (σταθερά 1, σταθερά 2, …). Ο έλεγχος μεταφέρεται μετά τη σταθερά με την οποία ισούται η τιμή της έκφρασης, δηλαδή σε μία εκ των εντολή 1, εντολή 2, … Αν δεν ισούται με καμία από τις σταθερές, ο έλεγχος μεταφέρεται στην εντολή που ακολουθεί την ετικέτα default, εάν βέβαια αυτή υπάρχει, αλλιώς στην εντολή μετά την ολοκλήρωση του σώματος της switch.

Ένα σημαντικό στοιχείο είναι ότι η ροή του προγράμματος συνεχίζει από την επιλεχθείσα case μέχρι το τέλος της εντολής switch ή μέχρι να συναντηθεί μια εντολή άμεσης μεταφοράς ελέγχου. Αυτό σημαίνει πως το σύστημα εκτελεί τις εντολές κάτω από την επιλεχθείσα case έως ότου συναντήσει μία από τις εντολές **break, goto ή return.** Η εντολή break, αν και πολλές φορές δίνεται στη γενική μορφή της switch, είναι προαιρετική και η εκτέλεσή της μεταφέρει τον έλεγχο έξω από την εντολή switch. Έτσι, αν για παράδειγμα λείπουν οι εντολές break από μια switch, η εκτέλεση των εντολών που ακολουθούν την επιλεγείσα case θα ακολουθηθεί από την εκτέλεση και των εντολών των επόμενων case ετικετών. Παρ' όλο που η break είναι προαιρετική, στην πράξη τη συναντάμε σχεδόν πάντα.

Η λειτουργία της switch διέπεται από ένα σύνολο κανόνων, οι σημαντικότεροι από τους οποίους είναι οι εξής:

- Κάθε case πρέπει να έχει μια int ή char σταθερά ή μια σταθερά έκφραση.
- Δύο case δεν μπορούν να έχουν εκφράσεις με την ίδια τιμή.
- Οι προτάσεις κάτω από την ετικέτα default εκτελούνται όταν δεν ικανοποιείται καμία από τις άλλες ετικέτες.
- case και default μπορούν να τοποθετηθούν με οποιαδήποτε σειρά. Αυτό σημαίνει πως η default δεν είναι απαραίτητα η τελευταία ετικέτα. Παρ' όλα αυτά, εκτελείται όταν δεν επιλεγεί καμία case προηγούμενη ή επόμενη.
- Η break μετά τη τελευταία ετικέτα αποτελεί καλή τακτική αν και δεν είναι απαραίτητη.

Παράδειγμα

```
#include <stdio.h>
main()
{
    int a;
    printf("Enter an integer: ");
    scanf("%d", &a);
```

```
    /* Check values of a */
    switch (a)
    {
        case 1:
            puts("You entered 1");
            break;
        case 2:
            puts("You entered 2");
            break;
        case 3:
            puts("You entered 3");
            break;
        case 4:
            puts("You entered 4");
            break;
        case 5:
            puts("You entered 5");
            break;
        case 6:
            puts("You entered 6");
            break;
        default:
            puts("You enter an integer less than 1 or more than 6");
    }
}
```

Ερωτήσεις αντικειμενικού τύπου - S5_LA14LO132

Να απαντήσετε ποιες από τις παρακάτω προτάσεις είναι Σωστές η Λανθασμένες.

1. Οι εντολές break είναι απαραίτητες σε όλες τις case ετικέτες.
2. Οι προτάσεις κάτω από την ετικέτα default εκτελούνται όταν καμία από τις case ετικέτες δεν ικανοποιείται.
3. Αν δεν ισούται η τιμή της έκφρασης με καμία από τις σταθερές, ο έλεγχος μεταφέρεται στην εντολή που ακολουθεί την ετικέτα default.
4. Κάθε case πρέπει να έχει μια λογική έκφραση.
5. Η εντολή switch είναι μια εντολή διακλάδωσης υπό συνθήκη.
6. Η ετικέτα default είναι απαραίτητα η τελευταία ετικέτα της εντολής switch.

7. Η πρόταση switch επιτρέπει τον προσδιορισμό απεριόριστου αριθμού διαδρομών ανάλογα με την τιμή μιας έκφρασης.
8. Στην εντολή switch η ετικέτα default εκτελείται πάντα.
9. Μια εντολή switch μπορεί να αντικατασταθεί από διαδοχικές απλές εντολές if.
10. Κάθε case πρέπει να έχει μια int ή char σταθερά ή μια σταθερά έκφραση.

Άσκηση αυτοαξιολόγησης - S5_LA14LO133

Έστω ένα πρόγραμμα σε γλώσσα C το οποίο διαβάζει τα αποτελέσματα των εξετάσεων αγγλικών ενός φοιτητή. Αν ο φοιτητής πήρε A το πρόγραμμα θα εμφανίζει το μήνυμα ARISTA, με B το μήνυμα KALA, με C το μήνυμα METRIA και με F το μήνυμα APETYXE. Να συμπληρώσετε τις εντολές που λείπουν.

```
#include <stdio.h>
main()
{
    char grade;
    printf("Dose Ba8mo Foititi, A, B, C, F: ");
    scanf("%c", &grade);
    …….
    {
        case 'A':
            printf("ARISTA ");
            break;
        ……
            printf("KALA ");
            break;
        case 'C':
            …….
            break;
        ……..
            printf("APETYXE ");
            break;
        default:
            printf("Edwses La8os Gramma ");
            break;
    }
    printf("\n");
    system("PAUSE");
}
```

3.5.3 Δραστηριότητα 3 - Προτάσεις ελέγχου ροής - επανάληψη

Τίτλος Δραστηριότητας	**S5_LA15 -** Προτάσεις ελέγχου ροής - επανάληψη
Τίτλος Ενότητας	Προτάσεις ελέγχου ροής
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης.
Περιγραφή	Στη δραστηριότητα αυτή, εξετάζουμε τη δομή επανάληψης. Επαναληπτικές δομές ονομάζονται μπλοκ κώδικα που εκτελούνται παραπάνω από μία φορές ανάλογα με τη συνθήκη που έχουμε ορίσει. Οι προτάσεις επανάληψης (iterative ή loop statements) διακρίνονται σε δύο κατηγορίες, ανάλογα με το αν γνωρίζουμε τον αριθμό των επαναλήψεων ή όχι. Στην πρώτη κατηγορία θα περιγράψουμε την εντολή for και στη δεύτερη κατηγορία τις εντολές while και do - while.
Γλώσσα	Ελληνικά
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει την δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα με χρήση των εντολών for, while και do while, • να αναπαραστήσουν τα παραπάνω παραδείγματα με διαγράμματα ροής, • να εφαρμόσουν την εντολή for για προσπέλαση ή εισαγωγή στοιχείων σε πίνακα.
Μαθησιακά Αντικείμενα	Πρόταση επανάληψης while Πρόταση επανάληψης do - while Πρόταση επανάληψης for
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 3 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S5_LA15LO134, S5_LA15LO138 και S5_LA15LO142.
Λέξεις Κλειδιά	Εντολή while, εντολή do - while, εντολή for, διάγραμμα ροής

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA15LO134**
Τίτλος	**Πρόταση επανάληψης while**
Τίτλος Δραστηριότητας	Προτάσεις ελέγχου ροής - επανάληψη
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται η εντολή while. Η while ανήκει στην κατηγορία των υπό συνθήκη προτάσεων επανάληψης και αποτελεί την πιο συχνά χρησιμοποιούμενη πρόταση αυτής της κατηγορίας. Η εκτέλεση του βρόχου εξαρτάται από την τιμή μιας έκφρασης που υπολογίζεται πριν από την εκτέλεση του βρόχου. Η εντολή αυτή χρησιμοποιείται όταν δεν γνωρίζουμε εξ αρχής τον αριθμό των επαναλήψεων.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Εντολή while, δομή επανάληψης, διάγραμμα ροής
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα με χρήση της εντολής while, • να αναπαραστήσουν τα παραπάνω παραδείγματα με διαγράμματα ροής.

Η while ανήκει στην κατηγορία των υπό συνθήκη προτάσεων επανάληψης και αποτελεί την πιο συχνά χρησιμοποιούμενη πρόταση αυτής της κατηγορίας. Κυρίως χρησιμοποιείται όταν δεν γνωρίζουμε εξ αρχής τον αριθμό των επαναλήψεων. Η εντολή while έχει την παρακάτω σύνταξη:

while (<ΣΥΝΘΗΚΗ>)

{ ΕΝΤΟΛΕΣ }

Η <ΣΥΝΘΗΚΗ> πρέπει υποχρεωτικά να είναι μέσα σε παρενθέσεις και μπορεί να είναι λογική έκφραση, έκφραση σύγκρισης, αποτέλεσμα κάποιας πράξης, είτε ακόμα και κάποια μεταβλητή. Ο βρόχος επανάληψης μπορεί να είναι μία ή περισσότερες εντολές (block) μέσα σε άγκιστρα {…}. Σε μια δομή επανάληψης, ο αριθμός των επαναλήψεων ελέγχεται συνήθως από μια μεταβλητή ελέγχου (control variable). Η μεταβλητή συνήθως παίρνει μια αρχική τιμή, σε κάθε επανάληψη παίρνει την επόμενη από μια προκαθορισμένη ακολουθία τιμών (control sequence) και αυτό επαναλαμβάνεται έως ότου η μεταβλητή ελέγχου φτάσει στο όριο της ακολουθίας τιμών. Στη συνέχεια, ο έλεγχος μεταφέρεται πάλι στην αρχή της while,

δηλαδή στον υπολογισμό της τιμής της έκφρασης. Αυτό συνεχίζεται έως ότου η έκφραση τελικά δώσει ψευδή τιμή, οπότε ο έλεγχος μεταφέρεται στην εντολή που ακολουθεί τη while.

Η ανάπτυξη της εντολής while είναι: «Συνέχισε να εκτελείς τις εντολές του βρόχου επανάληψης, όσο η έκφραση είναι αληθής»

Παράδειγμα με την εντολή while

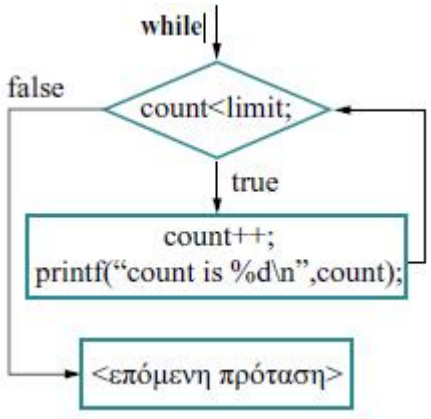
Το παρακάτω πρόγραμμα ζητά επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και το μέσο όρο τους.

```
#include <stdio.h>
float bathmos;
float total = 0.0;
int counter;
main()
{
        counter = 1;
        while (counter <= 10)
        {
                printf("Enter the score of student %d: ", counter);
                scanf("%f", &bathmos);
                total += bathmos;       /* Add the current score to the total */
                counter++;
        }
        printf("The total score is %lf", total);
        printf("The average score is %lf", total / 10);
}
```

Παρατήρηση: η μεταβλητή counter στον παραπάνω κώδικα λειτουργεί ως μετρητής των επαναλήψεων και επειδή βρίσκεται μέσα στη συνθήκη που τερματίζει την επανάληψη, πρέπει να μεταβάλει την αρχική τιμή της έτσι ώστε κάποτε να τερματιστεί η επανάληψη.

Το παρακάτω σχήμα δίνει το διάγραμμα ροής ενός παραδείγματος χρήσης της εντολής while.

παράδειγμα χρήσης της while	**διάγραμμα ροής**

while (count < limit) {
count++;
printf(“count is %d\n”, count);
}
< επόμενη πρόταση>
	

Παρατήρηση: Εάν η τιμή της limit είναι 12 και η τιμή της count μεγαλύτερη η ίση του 12, το σώμα του βρόχου δεν εκτελείται ούτε μια φορά.

Ερωτήσεις αντικειμενικού τύπου - S5_LA15LO135

Να απαντήσετε αν οι παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες.

1. Μία βοηθητική μεταβλητή (μετρητής) μέσα στο σώμα των εντολών της εντολής while πρέπει να μεταβάλει κάποτε την αρχική τιμή της συνθήκης για να τερματιστεί η επανάληψη.
2. Στην εντολή while, αν η συνθήκη είναι ΨΕΥΔΗΣ, θα εκτελεστεί η εντολή που βρίσκεται μετά την εντολή while (δηλαδή, η επόμενη εντολή).
3. Η μορφή επανάληψης while χρησιμοποιείται όταν γνωρίζουμε από την αρχή τον αριθμό των επαναλήψεων.
4. Οι εντολές που βρίσκονται σε μια εντολή while εκτελούνται τουλάχιστον μία φορά.
5. Στην εντολή while, οι μεταβλητές που συμμετέχουν στη συνθήκη πρέπει να αρχικοποιούνται πριν το βρόχο.
6. Με χρήση της εντολής while επιτυγχάνεται η επανάληψη μίας διαδικασίας με βάση κάποια συνθήκη.
7. Δεν μπορούμε να έχουμε μετρητή επαναλήψεων μέσα στο βρόχο της εντολής while.
8. Δεν γίνεται να βρίσκεται μια εντολή while μέσα σε μια άλλη εντολή while.
9. Μπορεί ο βρόχος της εντολής while να εκτελείται άπειρες φορές, όταν η μεταβλητή ελέγχου της συνθήκης δεν αλλάζει τιμή κατά τη διάρκεια της επανάληψης.
10. Όταν το πρόγραμμα δεν διαθέτει τρόπο τερματισμού της επανάληψης χαρακτηρίζεται ως ατέρμων βρόχος.

Άσκηση αυτοαξιολόγησης - S5_LA15LO136

Πόσες φορές θα εκτελεστούν οι παρακάτω επαναληπτικές δομές;

Α. int x=3; while (x>0) x = x-1;	**Β.** int x=1; while (x>3) x = x-1;
Γ. int x=3; while (x>0) x = x+1;	**Δ.** int x=9; while (x>=0) x = x-3;

Άσκηση αυτοαξιολόγησης - S5_LA15LO137

Στο παρακάτω πρόγραμμα, η εντολή εκτύπωσης θα εκτελεστεί 5 φορές. Γράψτε τι θα εκτυπώσει.

```
#include <stdio.h>
void main ()
{
	char a='Z';
	while (a > 40)
	{
		printf("a=%c ASCII value=%d\n", a, a);
		a = a - 10;
	}
}
```

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA15LO138**
Τίτλος	**Πρόταση επανάληψης do - while**
Τίτλος Δραστηριότητας	Προτάσεις ελέγχου ροής - επανάληψη
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται ο βρόχος do - while, ο οποίος είναι κατάλληλος στις περιπτώσεις που δεν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων. Εκτελείται καθόσον η συνθήκη παραμένει αληθής. Όταν η συνθήκη καταστεί ψευδής, ο έλεγχος του προγράμματος παρακάμπτει το περιεχόμενο του βρόχου και προχωρά στην επόμενη εντολή. Σε αντίθεση με την εντολή while, ο βρόχος επανάληψης θα εκτελεστεί οπωσδήποτε μία φορά, γιατί ο έλεγχος της συνθήκης γίνεται στο τέλος δηλαδή μετά την πρώτη εκτέλεση του βρόχου επανάληψης.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Εντολή do - while, διάγραμμα ροής, βρόχος επανάληψης
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα με χρήση της εντολής do - while, • να αναπαραστήσουν τα παραπάνω παραδείγματα με διαγράμματα ροής.

Σύνταξη της εντολής:

do

{ ΕΝΤΟΛΕΣ }

while (<ΣΥΝΘΗΚΗ>)

Η <ΣΥΝΘΗΚΗ> πρέπει να περικλείεται σε παρενθέσεις και μπορεί να είναι μια λογική έκφραση, μια έκφραση συσχετισμού, ένα αποτέλεσμα κάποιας πράξης, είτε ακόμα και κάποια μεταβλητή. Η ανάπτυξη της εντολής do - while είναι:

Εκτέλεση του βρόχου επανάληψης μέχρι η ΣΥΝΘΗΚΗ να μην είναι αληθής (ο έλεγχος εδώ γίνεται στο τέλος του βρόγχου).

Παράδειγμα με την εντολή do-while

Το παρακάτω πρόγραμμα ζητά επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και το μέσο όρο τους:

```
#include <stdio.h>
float bathmos;
float total = 0.0;
int counter;

main()
{
	counter = 1;
	do
	{
		printf("Enter the score of student %d: ", counter);
		scanf("%f", &bathmos);
		total += bathmos; /* Add the current score to the total */
		counter++;
	}
	while (counter <= 10);
	printf("The total score is %lf", total);
	printf("The average score is %lf", total / 10);
}
```

Η C επίσης υποστηρίζει τις παρακάτω εντολές ελέγχου, που χρησιμοποιούνται για τον τερματισμό ή τη συνέχιση μιας επανάληψης:

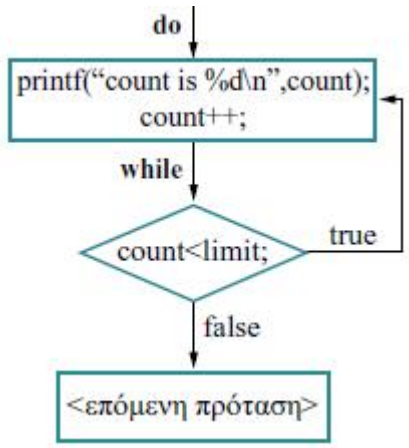
break: Τερματισμός εκτέλεσης μιας επανάληψης και συνέχιση του προγράμματος μετά από αυτή (δείτε την προηγούμενη ενότητα - Εντολές Ελέγχου).

continue: Διακοπή εκτέλεσης των εντολών της επανάληψης και συνέχιση της επανάληψης αν συνεχίζει να είναι αληθής η ΣΥΝΘΗΚΗ από την αρχή του βρόχου.

Παρατήρηση: Η πρόταση do - while, σε αντίθεση με τη while, υπολογίζει την έκφραση και αποφασίζει για την επανάληψη ή όχι του βρόχου μετά την εκτέλεση του βρόχου επανάληψης. Αυτό σημαίνει πως έχουμε μία τουλάχιστον εκτέλεση του βρόχου, ανεξάρτητα από την τιμή της συνθήκης. Το στοιχείο αυτό αποτελεί και τη μόνη διαφορά από την εντολή while.

Το παρακάτω σχήμα δίνει το διάγραμμα ροής ενός παραδείγματος χρήσης της do - while.

παράδειγμα χρήσης της do - while	**διάγραμμα ροής**

do {
count++;
printf("count is %d\n", count);
} while (count < limit);
< επόμενη πρόταση >
	

Παρατήρηση: Αν η τιμή της limit είναι 12, το σώμα του βρόχου εκτελείται μία φορά, έστω και αν η τιμή της count είναι μεγαλύτερη ή ίση του 12.

Ερωτήσεις αντικειμενικού τύπου - S5_LA15LO139

Να απαντήσετε αν οι παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες.

1. Η εντολή do - while χρησιμοποιείται όταν ο αριθμός επαναλήψεων είναι εκ των προτέρων γνωστός.
2. Στην εντολή do - while, αν η συνθήκη είναι ΨΕΥΔΗΣ θα εκτελεστεί η εντολή που βρίσκεται μετά την εντολή do - while (δηλαδή, η επόμενη εντολή).
3. Στην εντολή do - while, οι εντολές εκτελούνται όσο η συνθήκη είναι ΨΕΥΔΗΣ.
4. Μια επαναληπτική διαδικασία μπορεί να γραφεί σωστά είτε με την εντολή while είτε με την εντολή do - while.
5. Με την εντολή do - while υπάρχει ένας βρόχος που εκτελείται τουλάχιστον μία φορά.
6. Στις εντολές do - while και while που χρησιμοποιούνται για το ίδιο ζήτημα, οι συνθήκες είναι μεταξύ τους αντίθετες.
7. Εντός μιας δομής επιλογής δεν μπορεί να περιέχεται δομή επανάληψης.
8. Στην εντολή do - while, οι μεταβλητές που συμμετέχουν στη συνθήκη πρέπει να αρχικοποιούνται πριν το βρόχο.
9. Μια δομή επανάληψης πρέπει να φροντίζει για μεταβολή της τιμής της συνθήκης ώστε κάποτε να τερματίζεται.
10. Η εντολή break τερματίζει την εκτέλεση μιας επανάληψης και συνεχίζει το πρόγραμμα μετά από αυτήν.

Άσκηση αυτοαξιολόγησης - S5_LA15LO140

Πόσες φορές θα εκτελεστούν οι παρακάτω επαναληπτικές δομές;

Α.
x=3;
do x=x-1;
while (x<=0);
	Β.
	x=3;
	do x=x-1;
	while (x>0);

Γ.
x=3;
do x=x+1;
while (x<0);
	Δ.
	x=3; i=1;
	do {
	x=x+1;
	i-i+1;
	} while (x>=10);

Άσκηση αυτοαξιολόγησης - S5_LA15LO141

Δίνεται το παρακάτω πρόγραμμα:

```
#include <stdio.h>
main() {
    int x=0;
    int i=1;
    int y=0;
    do {
        if (i % 2 == 1) then
            x = x+1;
        else
            y=y+1;
        i=i+3;
    }
    while ((x>3) || (y>2));
    printf(" %d, %d, %d \n", x, y, i);
    getchar();
    return 0;
}
```

α) Πόσες φορές θα εκτελεστούν οι εντολές του βρόχου;

β) Να βρείτε τις τιμές των μεταβλητών X, Y και I που θα εμφανίσει το παραπάνω τμήμα προγράμματος.

Μαθησιακό Αντικείμενο	
Όνομα	**S5_LA15LO142**
Τίτλος	**Πρόταση επανάληψης for**
Τίτλος Δραστηριότητας	Προτάσεις ελέγχου ροής - επανάληψη
Περιγραφή	Στο συγκεκριμένο ΜΑ, περιγράφεται η εντολή for**.** Πρώτα αρχικοποιείται η μεταβλητή ελέγχου της επανάληψης και στη συνέχεια δοκιμάζεται η συνθήκη. Εάν δεν είναι αληθής ο έλεγχος μεταφέρεται εκτός της εντολής for. Διαφορετικά εκτελούνται οι εντολές του βρόχου επανάληψης, αυξάνεται ή μειώνεται η μεταβλητή ελέγχου κατά ένα βήμα και ο έλεγχος επιστρέφει στη δοκιμή της συνθήκης. Η εκτέλεση του βρόχου επανάληψης και η μεταβολή της μεταβλητής ελέγχου συνεχίζεται μέχρις ότου η συνθήκη να γίνει ψευδής.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Εντολή for, διάγραμμα ροής, βρόχος επανάληψης
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δώσουν τουλάχιστον δύο παραδείγματα με χρήση της εντολής for, • να αναπαραστήσουν τα παραπάνω παραδείγματα με διαγράμματα ροής, • να εφαρμόσουν τη χρήση της εντολής για την προσπέλαση ή την εισαγωγή στοιχείων σε πίνακα.

Η πρόταση for αποτελεί αναπαράσταση της ειδικής εκείνης περίπτωσης επανάληψης, στην οποία απαιτείται αρχικοποίηση μιας ή περισσότερων μεταβλητών πριν την είσοδο στο βρόχο και, επιπλέον, αλλαγή της τιμής αυτών των μεταβλητών μετά από κάθε εκτέλεση του σώματος του βρόχου.

Η γενικευμένη σύνταξη της εντολής είναι:

for (Ε1; <ΣΥΝΘΗΚΗ>; Ε3)
ΕΝΤΟΛΕΣ

ή

for (initialize; test; update)
ΕΝΤΟΛΕΣ

Ο βρόχος επανάληψης μπορεί να είναι μία ή περισσότερες εντολές (block) μέσα σε άγκιστρα {…}. Η εντολή Ε1 μπορεί να είναι μία ή και περισσότερες εντολές που συνήθως καθορίζουν κάποιες αρχικές τιμές. Η <ΣΥΝΘΗΚΗ> είναι συνήθως ο έλεγχος μιας έκφρασης συσχετισμού (ή και κάποιας λογικής έκφρασης), ενώ η εντολή Ε3 είναι συνήθως μια παράσταση αύξησης. Η ανάπτυξη της εντολής for είναι η εξής:

Εκτέλεση της εντολής Ε1 και στη συνέχεια όσο η ΣΥΝΘΗΚΗ είναι αληθής εκτέλεση του βρόχου επανάληψης και της εντολής Ε2.

Η εκτέλεσή της εντολής for περιγράφεται από τα παρακάτω βήματα:

1. Υπολογίζεται η τιμή της έκφρασης Ε1. Η έκφραση, που είναι συνήθως μια έκφραση ανάθεσης, αποδίδει αρχικές τιμές σε μία ή περισσότερες μεταβλητές, απ' όπου και το όνομα έκφραση αρχικοποίησης.

2. Υπολογίζεται η τιμή της έκφρασης που αποτελεί τη συνθήκη της πρότασης. Εάν η τιμή της είναι ψευδής, ο έλεγχος μεταφέρεται εκτός του βρόχου, διαφορετικά εκτελείται ο βρόχος επανάληψης.

3. Μετά την εκτέλεση του βρόχου, υπολογίζεται η έκφραση Ε3. Η έκφραση αυτή συνήθως μεταβάλει τις τιμές μιας ή περισσοτέρων μεταβλητών, απ' όπου και το όνομα έκφραση ενημέρωσης (update). Ο έλεγχος μεταφέρεται πάλι στον υπολογισμό της συνθήκης (βήμα 2).

Παράδειγμα της εντολής for:

```
for (i = 0; i < 10; i++)
    printf("%d\n", i);
```

Εκτελείται η εντολή Ε1 (i=0) και όσο η συνθήκη i<10 είναι αληθής τυπώνει το i (ΕΝΟΤΗΤΑ - η εντολή printf) και αυξάνει το i κατά 1 (εντολή Ε3).

Παράδειγμα της εντολής for

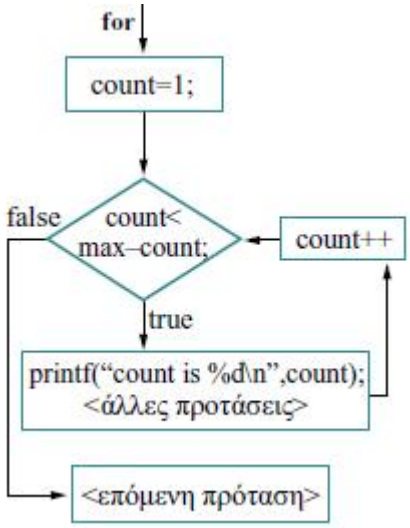
Το παρακάτω πρόγραμμα ζητά επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και το μέσο όρο τους:

```
#include <stdio.h>
float bathmos;
float total = 0.0;
int counter;
main()
{
    for (counter = 1; counter <= 10; counter++)
    {
        printf("Enter the score of student %d: ", counter);
        scanf("%f", &bathmos);
        /* Προσθέτω το βαθμό στη μεταβλητή total */
        total += bathmos;
```

```
    }
    printf("The total score is %lf", total);
    printf("The average score is %lf", total / 10);
}
```

Το παρακάτω σχήμα δίνει το διάγραμμα ροής ενός παραδείγματος χρήσης της for.

παράδειγμα χρήσης της for	**διάγραμμα ροής**

for (count=1; count<max_count; count++)
{
printf("count is %d\n", count);
}
< επόμενη πρόταση >
	

Παρατηρήσεις:

Με τον παρακάτω κώδικα τυπώνονται τα κεφαλαία γράμματα της λατινικής αλφαβήτου.

```
for (c = 'A'; c <= 'Z'; ++c)
    putchar(c);
```

Επίσης παρακάτω βλέπετε ένα βρόγχο που δεν σταματά ποτέ. Είναι άραγε χρήσιμοι στον προγραμματισμό;

```
for ( ; ; )
    { ΕΝΤΟΛΕΣ }
```

Χρήσιμες εφαρμογές της εντολής for

Η εντολή for σε μονοδιάστατους πίνακες

- **Διάβασμα πίνακα:**

```
int i, x[N]; /* Δηλώσεις. Το N είναι συμβολική τιμή */
for (i=0; i<=N-1; i++) /* Διάβασμα πίνακα από το χρήστη*/
{
    printf("Dose to %d stoixeio: ", i+1);
    scanf("%d", &x[i]);
}
```

- **Εκτύπωση πίνακα:**

```
for (i=0; i<=N-1; i++) /* Εκτύπωση πίνακα */
    printf("%d\t", x[i]);
```

- **Εύρεση μέγιστου στοιχείου πίνακα:**

```
int max=x[0]; /*Αρχικοποίηση μεταβλητής max */
for (i=1; i<=N-1; i++)
{
    if (x[i]>max)
        max=x[i];
}
```

Η εντολή for σε δισδιάστατους πίνακες

- **Διάβασμα πίνακα:**

```
int i, j, x[N][M]; /* Δηλώσεις. Τα N, M είναι συμβολικές τιμές */
for (i=0; i<=N-1; i++) /* Διάβασμα πίνακα (γραμμές) */
    for (j=0; j<=M-1; j++)  /* (στήλες) */
    {
        printf("Dose to stoixeio x[%d][%d]: ", i+1, j+1);
        scanf("%d", &x[i][j]);
    }
```

- **Εκτύπωση πίνακα:**

```
for (i=0; i<=N-1; i++)
{
    for (j=0; j<=M-1; j++)
        printf("%d\t", x[i][j]);
    printf("\n");       /*Αλλαγή γραμμής*/
}
```

Η εντολή for με χαρακτήρες

Το παρακάτω πρόγραμμα εκτυπώνει όλο το αγγλικό αλφάβητο σε πεζά και κατόπιν σε κεφαλαία.

```
#include <stdio.h>
main ()
{
    char i;
```

```
        printf("The alphabet, in lower-case:\n");
        for (i='a'; i<='z'; i++)
        {
                printf("%c", i);
        }
        printf("\nThe alphabet in upper-case:\n");
        for (i='A'; i<='Z'; i++)
        {
                printf("%c", i);
        }
}
```

Ερωτήσεις αντικειμενικού τύπου - S5_LA15LO143

Να απαντήσετε αν οι παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες.

1. Στη for, μία εντολή της ομάδας εντολών πρέπει να αλλάζει την τιμή του μετρητή, ώστε η επανάληψη να τερματιστεί.
2. Κάθε βρόχος που υλοποιείται με την εντολή while μπορεί να γραφεί και με χρήση της εντολής for.
3. Οι εντολές που βρίσκονται σε μία επανάληψη for εκτελούνται τουλάχιστον μία φορά.
4. Όταν ένας βρόχος είναι μέσα σε άλλο βρόχο, ο βρόχος που ξεκινάει τελευταίος πρέπει να ολοκληρώνεται πρώτος.
5. Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσότερων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.
6. Με τον παρακάτω κώδικα τυπώνονται τα κεφαλαία γράμματα της λατινικής αλφαβήτου: for (c='A'; c<='Z'; ++c) putchar(c);
7. Ο παρακάτω κώδικας for (i=1; i<=15; i++) printf("%d", i); εκτελείται άπειρες φορές.
8. Στην εντολή for δεν είναι δυνατόν η αρχική τιμή να είναι μεγαλύτερη από την τελική.
9. Η εντολή for πρέπει να περιλαμβάνει για βήμα πάντοτε ένα θετικό αριθμό.
10. Στην εντολή for το βήμα δεν μπορεί να είναι μηδέν.

Άσκηση αυτοαξιολόγησης - S5_LA15LO144

Δίνεται το παρακάτω τμήμα προγράμματος:

```
int k=50;
int i;
for (i=0; i<=20; i=i+5)
        k=k-1;
        printf("%d", k);
```

α) Πόσες φορές θα εκτελεστεί η επαναληπτική εντολή for;

β) Τι θα εμφανίσει το παραπάνω τμήμα κώδικα;

Άσκηση αυτοαξιολόγησης - S5_LA15LO145

Τι θα εμφανίσουν τα παρακάτω τμήματα προγράμματος;

α)

```
int i, j;
for (i=3; i<=5; i=i+1)
    for (j=2; j<=10; j=i+j)
        printf("%d,", 2*i+j);
```

β)

```
int i, j;
for (i=5; i>=2; i=i-2)
    for (j=1; j<=i; j=i+j)
        printf("%d,", i+2*j);
```

γ)

```
int i, j;
for (i=1; i<=3; i=i+1)
    for (j=4; j>=i; j=i-1)
        printf("%d,", 2*j-i);
```

Άσκηση αυτοαξιολόγησης - S5_LA15LO146

Έστω ένα πρόγραμμα με το οποίο υπολογίζεται η παράσταση:

$1^2+3^2+5^2+...+11^2$

Να συμπληρώσετε τα κενά στον παρακάτω κώδικα ώστε να υλοποιείται η παραπάνω διεργασία.

```
#include <stdio.h>
void main()
{
    int i, sum=0;
    for (i=…; i<=…; i+=…)
        sum+=…;
    printf("Parastasi = %d\n", sum);
}
```

3.6 Ενότητα 6 - Συναρτήσεις

Τίτλος Ενότητας	**S6** - Συναρτήσεις
Τίτλος Μαθήματος	Βασικές αρχές προγραμματισμού με τη γλώσσα προγραμματισμού C
Περιγραφή	Στην ενότητα αυτή θα παρουσιαστούν οι συναρτήσεις, οι οποίες αναπαριστούν ξεχωριστές διεργασίες σαν αυτόνομα τμήματα κώδικα που μπορούν να κληθούν επανειλημμένα σε ένα πρόγραμμα, δεχόμενες κάθε φορά διαφορετικές τιμές στις εισόδους τους. Θα γίνει περιγραφή του σχεδιασμού μιας συνάρτησης, καθώς και της κλήσης της. Τέλος, θα τεθούν ειδικά θέματα που αφορούν στις συναρτήσεις, όπως γιατί χρειάζεται να δείχνουμε στις διευθύνσεις των μεταβλητών των συναρτήσεων ή πώς χειρίζεται η γλώσσα C τους πίνακες μέσα στις συναρτήσεις.
Εκπαιδευτικοί Στόχοι	Στόχοι για την ενότητα αυτή είναι να μπορούν οι εκπαιδευόμενοι: • να δίνουν τον ορισμό της συνάρτησης, • να αναφέρουν 4 τουλάχιστον χαρακτηριστικά που θέλουμε να έχει κάθε συνάρτηση, • να αναφέρουν 4 τουλάχιστον πλεονεκτήματα που απορρέουν από τη χρήση συναρτήσεων.
Μαθησιακά Αποτελέσματα **(ΜΑπ)**	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση συναρτήσεων, • να δίνουν παραδείγματα υλοποίησης συναρτήσεων, • να ξεχωρίζουν τις τοπικές από τις πραγματικές μεταβλητές, • να χρησιμοποιούν πίνακες ως ορίσματα συναρτήσεων.
Εκπαιδευτικές Δραστηριότητες	Οι εκπαιδευτικές δραστηριότητες της συγκεκριμένης ενότητας είναι οι παρακάτω: • Στην πρώτη, παρουσιάζουμε μια εισαγωγή για την έννοια της συνάρτησης και τα βασικά της χαρακτηριστικά. • Στη δεύτερη, παρουσιάζουμε την υλοποίηση συναρτήσεων με την συγγραφή τους και την κλήση τους. • Στην τρίτη, παρουσιάζουμε ειδικά θέματα συναρτήσεων.
Εμπλεκόμενοι Ρόλοι	Κατά την διεξαγωγή του μαθήματος εμπλέκονται οι ρόλοι του σχεδιαστή της ενότητας και των εκπαιδευόμενων.

Αξιολόγηση	Η ενότητα του μαθήματος αυτού χρησιμοποιεί σε κάθε δραστηριότητα και σε κάθε μαθησιακό αντικείμενο ασκήσεις αυτοαξιολόγησης. Τέτοιες ασκήσεις μπορεί να είναι ερωτήσεις επίλυσης προβλημάτων ή ερωτήσεις αντικειμενικού τύπου. Για τη διεκπεραίωση αυτών των ασκήσεων μπορεί να χρησιμοποιηθούν διάφορα εργαλεία λογισμικού, όπως προγράμματα σχεδιασμού ερωτήσεων συμπλήρωσης κενού, σταυρόλεξα κλπ. ή λογισμικό σχεδιασμού ανάπτυξης προγραμμάτων στη γλώσσα C.
Συνολικός χρόνος Ενότητας	1 εβδομάδα

3.6.1 Δραστηριότητα 1 - Εισαγωγή στις Συναρτήσεις

Τίτλος Δραστηριότητας	***S6_LA16*** - Εισαγωγή στις Συναρτήσεις
Τίτλος Ενότητας	Συναρτήσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Η συνάρτηση είναι μία αυτόνομη μονάδα κώδικα σχεδιασμένη να επιτελεί συγκεκριμένο έργο. Βοηθά στην αποφυγή της επανάληψης, στην αύξηση της επαναχρησιμοποίησης, στη βελτίωση της αναγνωσιμότητας του κώδικα, καθώς και στη βελτίωση της διαδικασίας συντήρησης. Κάθε πρόγραμμα σε γλώσσα C αποτελείται από μία ή περισσότερες συναρτήσεις και περιέχει οπωσδήποτε τη συνάρτηση main από την οποία, όπως ήδη γνωρίζετε, αρχίζει η εκτέλεση του προγράμματος.
Γλώσσα	***Ελληνικά***
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση μιας συνάρτησης, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C που είδη γνωρίζετε, • να αναφέρουν χαρακτηριστικά των συναρτήσεων και του προγραμματισμού που τις υποστηρίζει.
Μαθησιακά Αντικείμενα	Αφαιρετικότητα στις διεργασίες Εισαγωγή στις συναρτήσεις
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 5 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για την θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα **S6_LA16LO147** και **S6_LA16LO150**.
Λέξεις Κλειδιά	Συνάρτηση, διεργασία, διαδικασία, δήλωση και κλήση συνάρτησης

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA16LO147**
Τίτλος	**Αφαιρετικότητα στις διεργασίες**
Τίτλος δραστηριότητας	Εισαγωγή στις Συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι διεργασίες, οι οποίες υλοποιούνται ως αυτόνομα τμήματα κώδικα, με αποτέλεσμα να σχηματίζουμε διαδικασίες ή συναρτήσεις. Η γλώσσα προγραμματισμού C υποστηρίζει μόνο συναρτήσεις που χαρακτηρίζονται από ένα όνομα, είσοδο, έξοδο και σώμα, δηλαδή ένα σύνολο ενεργειών, οι οποίες έχουν ομαδοποιηθεί με στόχο την εκτέλεση συγκεκριμένου έργου. Το σώμα της συνάρτησης το ονομάζουμε υλοποίηση (implementation), ενώ τα υπόλοιπα (όνομα, είσοδοι, έξοδοι) αποτελούν τον τρόπο αναφοράς στη συνάρτηση, τη διεπαφή (interface) όπως λέμε της συνάρτησης.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Διεργασία, συνάρτηση, διαδικασία, διεπαφή της διεργασίας, υπορουτίνα
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να κατανοήσουν την έννοια της αφαιρετικότητας στις διεργασίες με την χρήση κατασκευών όπως οι συναρτήσεις, • να γνωρίζουν την αναγκαιότητα της χρήσης των συναρτήσεων, • να δώσουν τουλάχιστον τρία παραδείγματα διεργασιών που μπορούν να υλοποιηθούν με συναρτήσεις.

Στην πρώτη ενότητα του μαθήματος αναφερθήκαμε στη διεργασία που περιγράφει το στιγμιότυπο ενός προγράμματος που εκτελείται σε έναν υπολογιστή. Σε αντιδιαστολή με την έννοια του προγράμματος, το οποίο είναι ένα στατικό σύνολο εντολών, μια διεργασία συνιστά την εκτέλεση αυτών των εντολών. Επομένως, ένα πρόγραμμα γενικώς συσχετίζεται με περισσότερες από μία διεργασίες, μία για κάθε φορά που εκτελείται.

Μια διεργασία μπορεί να θεωρηθεί αποτελούμενη από τρία επί μέρους τμήματα:

1. το όνομα της διεργασίας

2. τις εισόδους και εξόδους της διεργασίας

3. το σώμα της διεργασίας

Το όνομα χρησιμοποιείται για να προσδιορίσει τη διεργασία. Οι είσοδοι προσδιορίζουν το σύνολο των στοιχείων εισόδου της διεργασίας και οι έξοδοι το αποτέλεσμα της διεργασίας. Τέλος, το σώμα της διεργασίας είναι η περιγραφή ενός συνόλου ενεργειών, οι οποίες έχουν ομαδοποιηθεί με στόχο την εκτέλεση συγκεκριμένου έργου.

Τι κάνετε όταν θέλετε ο υπολογιστής να εκτελέσει τη διεργασία της εύρεσης του μέγιστου μεταξύ δύο αριθμών; Αναφέρεστε στη διεργασία με το όνομά της (όνομα προγράμματος), δίνετε τις εισόδους (τους δύο αριθμούς) και ο υπολογιστής, έχοντας το σώμα της διεργασίας, την εκτελεί και σας δίνει το αποτέλεσμα. Το σώμα της διεργασίας το ονομάζουμε υλοποίηση (implementation) της διεργασίας, ενώ τα υπόλοιπα (όνομα, είσοδοι, έξοδοι) αποτελούν τον τρόπο αναφοράς στη διεργασία, τη διεπαφή (interface) όπως λέμε της διεργασίας.

Η συνάρτηση είναι για τη C η γλωσσική κατασκευή που αναπαριστά τη διεργασία. Άλλες κατασκευές που χρησιμοποιούνται για αναπαράσταση διεργασιών είναι η διαδικασία ή υπορουτίνα (subroutine) και η λειτουργία (operation). Κάποιες γλώσσες προγραμματισμού, όπως η Pascal, υποστηρίζουν και τις συναρτήσεις και τις υπορουτίνες, ενώ κάποιες γλώσσες, όπως η C, υποστηρίζουν μόνο τις συναρτήσεις. Η διαφορά μεταξύ διαδικασιών και συναρτήσεων είναι ότι οι η διαδικασία περιγράφει κάθε διεργασία, ενώ η συνάρτηση περιγράφει μόνο τις διεργασίες που επιστρέφουν μια τιμή. Όσο για τη λειτουργία, αυτή αποτελεί βασικό χαρακτηριστικό του αντικειμενοστρεφούς προγραμματισμού.

Οι συναρτήσεις αποτελούν αυτόνομες, επώνυμες μονάδες κώδικα, σχεδιασμένες να επιτελούν συγκεκριμένο έργο. Μπορούν να κληθούν επανειλημμένα σε ένα πρόγραμμα, δεχόμενες κάθε φορά διαφορετικές τιμές στις εισόδους τους.

Οφέλη από τη χρήση συναρτήσεων για τη C είναι η αποφυγή επαναλήψεων, η αύξηση επαναχρησιμοποίησης κώδικα, η βελτίωση αναγνωσιμότητας κώδικα και η καλύτερη συντήρηση των προγραμμάτων που περιέχουν συναρτήσεις.

Η εφαρμογή της αφαιρετικότητας στις διεργασίες έχει να κάνει με το ότι επιτρέπεται στον προγραμματιστή να εστιάσει την προσοχή του στη διεργασία που εκτελείται στο σημείο κλήσης της συνάρτησης και όχι στον τρόπο με τον οποίο αυτή επιτελείται. Δηλαδή η αφαιρετικότητα (abstraction) είναι η έννοια με βάση την οποία γίνεται διαχωρισμός ανάμεσα στο τι και στο πώς. Αυτό επιτυγχάνεται διακρίνοντας τα δύο τμήματα, της διεπαφής και της υλοποίησης. Η **διεπαφή**, γνωστή στη C ως **πρωτότυπο της συνάρτησης**, επιτρέπει στο μεταγλωττιστή να διασφαλίσει ότι κάθε κλήση της συνάρτησης είναι σύμφωνη με τον ορισμό της, προφυλάσσοντας τον προγραμματιστή από δύσκολα στην ανεύρεση λάθη.

Κάθε επιμέρους διεργασία του προγράμματος αναπτύσσεται σαν μια αυτόνομη συνάρτηση και, τελικά, όλες οι συναρτήσεις ενοποιούνται για να αποτελέσουν το πρόγραμμα.

Κάθε συνάρτηση πρέπει να έχει τα παρακάτω χαρακτηριστικά:

1. να είναι σχετικά μικρό πρόγραμμα,

2. να έχει αν είναι δυνατό μια είσοδο και μια έξοδο,

3. να αναπτύσσεται και να ελέγχεται σαν αυτόνομη μονάδα,

4. να εκτελεί ένα σαφώς ορισμένο έργο, και

5. να έχει μια καλώς προσδιορισμένη διεπαφή (interface).

Έχουμε συναρτήσεις Βιβλιοθήκης και εγγενείς συναρτήσεις. Οι πρώτες δεν ορίζονται από τον χρήστη, π.χ. printf(), scanf(), rand() και ενσωματώνονται με την εντολή #include της κατάλληλης βιβλιοθήκης όπου είναι ορισμένη η συνάρτηση (π.χ. <stdio.h>, <stdlib.h>). Οι δεύτερες συναρτήσεις ορίζονται από τον προγραμματιστή (user defined functions).

Ας αναφερθούμε σε μια διεργασία, η οποία είναι πολύ βασική και χρησιμοποιείται πολύ συχνά. Τι κάνουμε όταν θέλουμε να ζητήσουμε από τον υπολογιστή να μας εμφανίσει ένα αποτέλεσμα στην οθόνη; Αναφερόμαστε στη συνάρτηση printf της βασικής βιβλιοθήκης. Το μεν σώμα της printf βρίσκεται σε ένα από τα αρχεία .lib της βασικής βιβλιοθήκης της C, ο δε τρόπος αναφοράς της στο αρχείο επικεφαλίδας stdio.h, το οποίο και φροντίσαμε να συμπεριλάβουμε στο πρόγραμμά μας χρησιμοποιώντας την εντολή include του προεπεξεργαστή.

Άσκηση αυτοαξιολόγησης - S6_LA16LO148

Διαβάστε την περιγραφή του παρακάτω προγράμματος και καταγράψτε τις βασικές διεργασίες που θα πρέπει να εκτελεί το σύστημα στα πλαίσια της άσκησης. Εντοπίστε ποιες από αυτές υλοποιούνται από τη βασική βιβλιοθήκη.

```
#include <stdio.h>

main()

{

    int a, b, c, max;

    clrscr();

    printf("\nΔώσε τον πρώτο αριθμό: ");

    scanf("%d", &a);

    printf("\nΔώσε τον δεύτερο αριθμό: ");

    scanf("%d", &b);

    printf("\nΔώσε τον τρίτο αριθμό: ");

    scanf("%d", &c);

    max = a;

    if (b > max) max = b;

    if (c > max) max = c;

    printf("\nO μέγιστος είναι ο: %d", max);

} /* end of main */
```

Ερωτήσεις αντικειμενικού τύπου - S6_LA16LO149

Να απαντήσετε ποιες από τις παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες.

1. Οι συναρτήσεις αποτελούν αυτόνομες, επώνυμες μονάδες κώδικα, σχεδιασμένες να επιτελούν συγκεκριμένο έργο.

2. Ένα πρόγραμμα γενικώς δεν μπορεί να συσχετίζεται με περισσότερες από μία διεργασίες, κάθε φορά που εκτελείται.
3. Κάθε συνάρτηση πρέπει να αναπτύσσεται και να ελέγχεται σαν αυτόνομη μονάδα.
4. Οι συναρτήσεις δεν μπορεί να ορίζονται από τον χρήστη, παρά μόνο από τις βιβλιοθήκες που συμπεριλαμβάνει στο περιβάλλον της κάθε γλώσσα προγραμματισμού.
5. Η αφαιρετικότητα (abstraction) είναι ή έννοια με βάση την οποία γίνεται διαχωρισμός ανάμεσα στο τι και το πότε.
6. Οι συναρτήσεις βιβλιοθήκης ορίζονται από τον χρήστη και ενσωματώνονται με την εντολή #include της κατάλληλης βιβλιοθήκης όπου η συνάρτηση είναι ορισμένη.
7. Η διεπαφή της συνάρτησης, γνωστή στη C ως πρωτότυπο της συνάρτησης, επιτρέπει στο μεταγλωττιστή να διασφαλίσει ότι κάθε κλήση της συνάρτησης είναι σύμφωνη με τον ορισμό της, προφυλάσσοντας τον προγραμματιστή από δύσκολα στην ανεύρεση λάθη.
8. Κάθε συνάρτηση πρέπει να έχει μια καλώς προσδιορισμένη διεπαφή (interface).
9. Η διαφορά σε μια υπορουτίνα ή διαδικασία και σε μια συνάρτηση είναι ότι οι διαδικασίες περιγράφουν μόνο τις διεργασίες που επιστρέφουν μια τιμή.
10. Κάθε συνάρτηση πρέπει να έχει αν είναι δυνατό μία είσοδο και μία έξοδο.

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA16LO150**
Τίτλος	**Εισαγωγή στις συναρτήσεις της C**
Τίτλος δραστηριότητας	Εισαγωγή στις Συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι συναρτήσεις. Σύνθετα προβλήματα επιλύονται αποτελεσματικότερα με τη μέθοδο της αποσύνθεσης στην οποία τα επιμέρους υποπροβλήματα στα οποία διασπάται το βασικό πρόβλημα μπορεί να υλοποιηθούν από συναρτήσεις που μπορεί να ορίσει ο προγραμματιστής. Η συνάρτηση μπορεί να θεωρηθεί ως μια αυτόνομη μονάδα προγράμματος, η οποία δέχεται ως είσοδο ένα σύνολο δεδομένων, εκτελεί ένα σύνολο εντολών και επιστρέφει στη συνάρτηση που την κάλεσε μια τιμή. Στη C, μια συνάρτηση αποτελείται από δύο μέρη: την επικεφαλίδα και το σώμα της, το οποίο δεν είναι τίποτε άλλο παρά μια ομάδα εντολών.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Παρουσίαση, βιντεοδιάλεξη
Λέξεις Κλειδιά	Συνάρτηση, διαδικασία ή υπορουτίνα, επιστρεφόμενη τιμή, δήλωση συνάρτησης, κλήση συνάρτησης
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν την ενότητα αυτή, μελετήσει οι εκπαιδευόμενοι θα μπορούν: • να αναφέρονται σε συγκεκριμένα χαρακτηριστικά των συναρτήσεων, • να γνωρίζουν την αναγκαιότητα της χρήσης των συναρτήσεων, • να δώσουν τουλάχιστον τρία παραδείγματα χρήσης συναρτήσεων.

Η φιλοσοφία σχεδίασης της C βασίζεται στη χρήση των συναρτήσεων. Έχουμε ήδη δει και χρησιμοποιήσει πολλές συναρτήσεις που έχει το σύστημα, όπως είναι οι printf(), scanf(), strlen() και άλλες, αλλά έχουμε δημιουργήσει και δικές μας συναρτήσεις. Οι πιο πολλές είχαν το όνομα main(). Τα προγράμματα της C πάντα αρχίζουν με την εκτέλεση των εντολών της συνάρτησης main() και μετά η main() καλεί (πιθανώς) άλλες συναρτήσεις. Ήρθε βέβαια ο καιρός για να δούμε τις συναρτήσεις αναλυτικότερα.

Συνάρτηση αποκαλείται μια αυτοδύναμη μονάδα κώδικα προγράμματος που έχει σχεδιαστεί για να εκτελεί μια συγκεκριμένη εργασία. Μια συνάρτηση της C κάνει την ίδια δουλειά με τις συναρτήσεις, τις υπορουτίνες και τις διαδικασίες των άλλων γλωσσών. Μερικές συναρτήσεις, όπως η printf(), έχουν ως αποτέλεσμα να γίνει μια ενέργεια. Μερικές άλλες συναρτήσεις υπολογίζουν μια τιμή, που χρησιμοποιείται από το πρόγραμμα, όπως η strlen(). Γενικά, μια συνάρτηση μπορεί να παράγει ενέργειες ή και να παράγει τιμές.

Χρησιμοποιούμε τις συναρτήσεις για να γλυτώσουμε από τον επαναλαμβανόμενο προγραμματισμό. Δηλαδή, αν πρέπει να γίνει μια συγκεκριμένη εργασία πολλές φορές μέσα σ' ένα πρόγραμμα, τότε γράφουμε μια φορά μια κατάλληλη συνάρτηση και μετά τη χρησιμοποιούμε από το πρόγραμμα όταν και όπου τη χρειαζόμαστε.

Μπορούμε να χρησιμοποιήσουμε την ίδια συνάρτηση σε διαφορετικά προγράμματα, αλλά, ακόμα και αν κάνουμε μια εργασία μόνο μία φορά μέσα στο πρόγραμμα, θα είναι πιο σωστό να χρησιμοποιήσουμε μια συνάρτηση, επειδή έτσι το πρόγραμμα γίνεται πιο εύκολο στην ανάγνωση και στην τροποποίησή του και είμαστε πιο κοντά σ' αυτό που λέγεται δομημένος προγραμματισμός. Για παράδειγμα, αν θέλουμε να κάνουμε ένα πρόγραμμα που να διαβάζει μια λίστα αριθμών, να τους ταξινομεί, να βρίσκει το μέσο όρο τους και να τυπώνει ένα ραβδόγραμμα, θα μπορούσαμε να δημιουργήσουμε το εξής πρόγραμμα:

```
#include <stdio.h>

#define SIZE 50

main()

{

	float list[SIZE];


	readlist(list, SIZE);

	sort(list, SIZE);

	average(list, SIZE);

	bargraph(list, SIZE);

}
```

Όταν χρησιμοποιούμε περιγραφικά ονόματα για τις συναρτήσεις, είναι ευκολότερο να καταλάβουμε τι κάνει ένα πρόγραμμα και πώς οργανώνεται. Μπορούμε ακόμα να επεξεργαστούμε ξεχωριστά κάθε συνάρτηση μέχρι να πετύχουμε τη σωστή εκτέλεση μιας εργασίας. Ένα ακόμη όφελος είναι ότι αν δημιουργήσουμε τις συναρτήσεις κατά γενικό τρόπο, θα μπορούμε να τις χρησιμοποιήσουμε και σ' άλλα προγράμματα.

Οι συναρτήσεις έχουν ως σκοπό να μας βοηθήσουν να επικεντρώνουμε την προσοχή μας στη συνολική σχεδίαση ενός προγράμματος και όχι στις λεπτομέρειές του. Πριν γράψουμε τον κώδικα μιας συνάρτησης, πρέπει να σκεφτούμε πρώτα τι δουλειά πρέπει να κάνει αυτή η συνάρτηση και πώς θα συσχετίζεται με το όλο πρόγραμμα.

Συνοψίζοντας, θα μπορούσαμε να πούμε ότι η χρήση των συναρτήσεων συνιστάται διότι παρέχει τη δυνατότητα: α) κατάτμησης ενός προβλήματος σε υποπροβλήματα τα οποία αντιμετωπίζονται ευκολότερα (τεχνική «διαίρει και βασίλευε»), β) κατανομής του συνολικού προγραμματιστικού έργου σε διάφορες ομάδες, γ) επαναχρησιμοποίησης τμημάτων κώδικα και δ) μείωσης του μεγέθους του κώδικα.

Η προσθήκη μιας νέας συνάρτησης σε ένα πρόγραμμα C περιλαμβάνει τα εξής βήματα:

1. Δήλωση της συνάρτησης (προσδιορισμός του πρωτοτύπου της συνάρτησης) πριν από τη συνάρτηση main του προγράμματος (συνήθως μετά τις οδηγίες #include).
2. Ορισμός της συνάρτησης (προσδιορισμός των εντολών του σώματος της συνάρτησης) σε κάποιο σημείο μετά τη συνάρτηση main του προγράμματος.
3. Κλήση της συνάρτησης χρησιμοποιώντας το όνομα της και «περνώντας» τις αναγκαίες παραμέτρους, σε όποιο σημείο του προγράμματος απαιτείται.

Μία δήλωση συνάρτησης έχει τη μορφή:

επιστρεφόμενος_τύπος όνομα_συνάρτησης(λίστα παραμέτρων);

Παράδειγμα: συνάρτηση υπολογισμού του μέγιστου μεταξύ δύο ακεραίων

int max(int m, int n)

Ο ορισμός της συνάρτησης έχει τη μορφή:

επιστρεφόμενος_τύπος όνομα_συνάρτησης(λίστα παραμέτρων)
{
δηλώσεις μεταβλητών
εντολές συνάρτησης
return επιστρεφόμενη_τιμή
}

Παράδειγμα:

```
int max(int m, int n)
{
    int max
    if (m >= n)
        max = m;
    else
        max = n;
    return max;
}
```

Η εντολή return επιστρέφει στην καλούσα συνάρτηση μία τιμή, η οποία πρέπει να συμφωνεί με τον επιστρεφόμενο τύπο της συνάρτησης. Ωστόσο, μία συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει τιμή, οπότε αντί για επιστρεφόμενο τύπο θα έχει τη λέξη - κλειδί void.

Ερωτήσεις αντικειμενικού τύπου - S6_LA16LO151

Να απαντήσετε ποιες από τις παρακάτω προτάσεις είναι Σωστές ή Λανθασμένες.

1. Κάθε συνάρτηση πρέπει να έχει μόνο μία είσοδο και μία έξοδο.
2. Μία συνάρτηση μπορεί να καλέσει τη συνάρτηση main.

3. Η εντολή return επιστρέφει στην καλούσα συνάρτηση μία τιμή, η οποία πρέπει να συμφωνεί με τον επιστρεφόμενο τύπο της συνάρτησης.
4. Οι συναρτήσεις μπορούν να υπολογίζουν μια τιμή και να την επιστρέφουν.
5. Μια συνάρτηση μπορεί να καλέσει μια άλλη συνάρτηση.
6. Όφελος είναι ότι αν δημιουργήσουμε τις συναρτήσεις κατά γενικό τρόπο, θα μπορούμε να τις χρησιμοποιήσουμε και σ' άλλα προγράμματα.
7. Η συνάρτηση main πρέπει να είναι πολύ μεγαλύτερη από τις άλλες συναρτήσεις.
8. Η χρήση των συναρτήσεων παρέχει τη δυνατότητα αύξησης του όγκου των προγραμμάτων με τη γλώσσα C.
9. Τα προγράμματα της C πάντα αρχίζουν με την εκτέλεση των εντολών της συνάρτησης main() και μετά η main() καλεί άλλες συναρτήσεις.
10. Οι συναρτήσεις έχουν ως σκοπό να μας βοηθήσουν να επικεντρώνουμε την προσοχή μας στη συνολική σχεδίαση ενός προγράμματος και όχι στις λεπτομέρειές του.

Άσκηση αυτοαξιολόγησης - S6_LA16LO152

Τι θα τυπώσουν οι παρακάτω εντολές;

```
#include <stdio.h>
int function1(int, int);
main()
{
    int x, a=5, b=10, c=0;
    x = function1(a, b);
    printf("\n %d %d %d\n", x, b, c);
    getchar() ;
}

int function1(int m, int n)
{
    int k;
    m=m-n;
    k=m;
    return k;
}
```

Α. 5, 10, 0 Β. 5, 10, -5 Γ. -5, 10, 0 Δ. -5, 10, -5

Άσκηση αυτοαξιολόγησης - S6_LA16LO153

Τι θα τυπώσουν οι παρακάτω εντολές;

```
#include <stdio.h>
int function2(int, int);
main()
{
        int x, a=5, b=10;
        x = function2(b, a);
        printf(“\n %d %d \n”, a, b);
        getchar();
}

int function2(int m, int n)
{
        int k;
        printf("\n %d %d \n", m, n);
        m=m-n;
        return k;
}
```

A. 5, 10 B. 10, 5 Γ. 5, 10 Δ. 10, 5
5, 10 5, 5 -5, 10 5, 10

Άσκηση αυτοαξιολόγησης - S6_LA16LO154

Τι θα τυπώσουν οι παρακάτω εντολές;

```
#include <stdio.h>
int function3(int, int);
main()
{
        int x, a=10, b=5;
        x = function3(a, b);
        printf("\n %d %d \n", a, b);
}
int function3(int m, int n)
{
```

```
    int a, b;
    a=0;
    b=0;
    printf("\n %d %d \n", a, b);
}
```

Α. 10, 5
0, 0

Β. 10, 5
10, 5

Γ. 0, 0
0, 0

Δ. 0, 0
10, 5

3.6.2 Δραστηριότητα 2 - Δημιουργία και Χρήση μιας Απλής Συνάρτησης

Τίτλος Δραστηριότητας	***S6_LA17*** **-** Δημιουργία και Χρήση μιας Απλής Συνάρτησης
Τίτλος Ενότητας	Συναρτήσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Στη δραστηριότητα αυτή περιγράφεται η υλοποίηση της χρήσης των συναρτήσεων στη γλώσσα C. Ένα τμήμα κώδικα που είναι σε μεγάλο βαθμό ανεξάρτητο από το υπόλοιπο πρόγραμμα μπορεί να αποτελέσει μια συνάρτηση. Το τμήμα αυτό περιλαμβάνει δηλώσεις ποσοτήτων και εκτελέσιμες εντολές και μπορεί να παραμετροποιείται από κάποιες σταθερές ή μεταβλητές ποσότητες, τα ορίσματα της συνάρτησης. Μια συνάρτηση μπορεί να μην επιστρέφει τίποτε ή να επιστρέφει μία απλή ή σύνθετη ποσότητα. Η κλήση μιας συνάρτησης γίνεται παραθέτοντας το όνομά της, ακολουθούμενο σε παρενθέσεις από ποσότητες κατάλληλου τύπου ώστε να αντιστοιχούν στα ορίσματά της.
Γλώσσα	**Ελληνικά**
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση μιας συνάρτησης, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C που είδη γνωρίζετε, • να αναφέρουν χαρακτηριστικά των συναρτήσεων και του προγραμματισμού που τις υποστηρίζει, • να δηλώνουν μια συνάρτηση, • να καλούν μια συνάρτηση από το κυρίως πρόγραμμα.
Μαθησιακά Αντικείμενα	Δημιουργία και Χρήση μιας Απλής Συνάρτησης
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 4 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στο μαθησιακό αντικείμενο **S6_LA17LO155**.
Λέξεις Κλειδιά	Συνάρτηση, διαδικασία, επιστροφή τιμής, ορίσματα συνάρτησης, πρωτότυπο συνάρτησης, κλήση συνάρτησης, δήλωση συνάρτησης

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA17LO155**
Τίτλος	**Δημιουργία και χρήση μιας απλής συνάρτησης**
Τίτλος δραστηριότητας	Εισαγωγή στις συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι συναρτήσεις. Η συνάρτηση είναι ένα υποπρόγραμμα που εκτελεί συγκεκριμένες λειτουργίες και ανάλογα με τα δεδομένα εισόδου (παράμετροι συνάρτησης) υπολογίζει τα δεδομένα εξόδου (επιστρεφόμενος τύπος). Όλες οι συναρτήσεις πρέπει αρχικά να δηλωθούν. Η δήλωση γίνεται πάντα πριν την συνάρτηση main. Η υλοποίηση μπορεί να γίνει ταυτόχρονα με τη δήλωση ή ξεχωριστά πριν ή μετά τη main. Όταν κληθεί μια συνάρτηση, ο έλεγχος του προγράμματος μεταφέρεται στην καλούμενη συνάρτηση.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Συνάρτηση, διαδικασία, επιστροφή τιμής, ορίσματα συνάρτησης, πρωτότυπο συνάρτησης, κλήση συνάρτησης, δήλωση συνάρτησης
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρονται στη διαδικασία κλήσης μιας συνάρτησης με παραδείγματα, • να γνωρίζουν τον τρόπο συγγραφής μιας συνάρτησης, • να γνωρίζουν τη διαφορά μεταξύ πραγματικών και τυπικών παραμέτρων, • να δώσουν τουλάχιστον τρία παραδείγματα δήλωσης συναρτήσεων.

Διαδικασία κλήσης συνάρτησης

Μια **συνάρτηση** είναι ένα σύνολο εντολών που έχουν ομαδοποιηθεί και τους έχει αποδοθεί ένα όνομα.

Η πράξη του συνόλου των εντολών που σχετίζονται με μια συνάρτηση ονομάζεται **κλήση** της συνάρτησης. Κατά την κλήση μιας συνάρτησης γράφουμε το όνομα της συνάρτησης

ακολουθούμενο από παραστάσεις μέσα σε παρενθέσεις, που χωρίζονται από κόμμα, που ονομάζονται **ορίσματα** της συνάρτησης ή παράμετροι.

Όταν κληθεί μια συνάρτηση:

- Παίρνει τα δεδομένα που της παρέχονται ως ορίσματα,

- Εκτελεί τις εντολές της,

- Επιστρέφει (return) στο σημείο από το οποίο κλήθηκε.

Κατά την επιστροφή της μια συνάρτηση είναι δυνατόν να «στείλει» μια τιμή πίσω στη συνάρτηση που την κάλεσε. Αυτή η λειτουργία ονομάζεται **επιστροφή τιμής**.

n = GetInteger();

Παράδειγμα κλήσης συνάρτησης

Κλήση της συνάρτησης υπολογισμού της τετραγωνικής ρίζας της μαθηματικής βιβλιοθήκης math της C.

#include <math.h> /* Συμπερίληψη του αρχείου επικεφαλίδας math.h */

...

x = sqrt(2.0);

distance = sqrt(x*x + y*y);

Γενικά

Δήλωση συναρτήσεων

Στην πρότυπη (ANSI) C, για να χρησιμοποιηθεί μια συνάρτηση σε ένα πρόγραμμα πρέπει πρώτα να έχει δηλωθεί. Υπενθυμίζεται ότι το ίδιο ισχύει και για τις μεταβλητές στη C. Η δήλωση μιας συνάρτησης ονομάζεται **πρωτότυπο συνάρτησης**. Σε αυτήν ορίζονται:

- Το όνομα της συνάρτησης,

- Ο τύπος κάθε ορίσματος και, τις περισσότερες φορές, ένα όνομα για το όρισμα,

- Ο τύπος της τιμής που επιστρέφει η συνάρτηση.

Μορφή Πρωτοτύπου Συνάρτησης

Επιστρεφόμενος_τύπος όνομα_συνάρτησης (τύπος_ορίσματος όνομα_ορίσματος,...);

• Παράδειγμα:

int sum (int x, int y);

ή

int sum(int, int);

double square(double x);

Πρωτότυπα συνάρτησης

Κάθε αρχείο επικεφαλίδας (header file) περιέχει τα πρωτότυπα των συναρτήσεων της αντίστοιχης βιβλιοθήκης.

Η επικεφαλίδα μιας συνάρτησης είναι η διεπαφή της συνάρτησης με το υπόλοιπο πρόγραμμα και είναι αυτό το οποίο χρειάζεται ένας προγραμματιστής για να τη χρησιμοποιήσει. Με την επικεφαλίδα ορίζεται το όνομα της συνάρτησης, ο τύπος της τιμής την οποία επιστρέφει όταν κληθεί, καθώς και μια λίστα παραμέτρων που καλούνται **τυπικές παράμετροι** της συνάρτησης, μέσω των οποίων η συνάρτηση δέχεται ως είσοδο δεδομένα, επιστρέφει δεδομένα ή και τα δύο. Αν παραληφθεί αυτό, τότε υπονοείται ο int. Η κλήση μιας συνάρτησης από το κυρίως πρόγραμμα απαιτεί και τη χρήση **πραγματικών παραμέτρων**, οι οποίες θα αντικαταστήσουν (κατά την κλήση της και μόνο) τις τυπικές παραμέτρους της συνάρτησης.

• Ένα πρωτότυπο προδιαγράφει τη μορφή μιας συνάρτησης αλλά δεν ορίζει τη λειτουργία της.

• Μια συνάρτηση που δεν έχει ορίσματα χρησιμοποιεί την ειδική λέξη void ως προδιαγραφή του ορίσματος.

Στην ANSI C χρησιμοποιείται η λέξη κλειδί void για να δηλωθεί ότι μια συνάρτηση δεν επιστρέφει τιμή, αλλά και για να δηλώσει την απουσία τυπικών παραμέτρων στην συνάρτηση. Στην περίπτωση που η συνάρτηση δεν επιστρέφει τιμή, ο ορισμός της έχει ως εξής:

void <όνομα συνάρτησης>(<λίστα τυπικών παραμέτρων>)

Συγγραφή μιας συνάρτησης

Η συγγραφή μιας συνάρτησης απαιτεί:

1. Τον ορισμό του πρωτοτύπου της συνάρτησης στην αρχή, συνήθως, του προγράμματος.

2. Την υλοποίηση της συνάρτησης στην οποία ορίζονται τα βήματα που θα εκτελεστούν από τη συνάρτηση.

Παράδειγμα:

Συνάρτηση μετατροπής θερμοκρασίας από βαθμούς Κελσίου σε Fahrenheit

Η μετατροπή γίνεται σύμφωνα με τη σχέση:

-F = 9/5 C + 32

Δεδομένα εισόδου της συνάρτησης: Η θερμοκρασία σε βαθμούς Κελσίου. Τύπος:

double

Δεδομένα εξόδου της συνάρτησης: Η θερμοκρασία σε βαθμούς Fahrenheit. Τύπος:

double

Ορισμός του πρωτοτύπου. Το πρωτότυπο της συνάρτησης είναι το παρακάτω:

double CelciusToFahrenheit(double c);

Η υλοποίηση μιας συνάρτησης ορίζεται ως εξής:

Επικεφαλίδα_συνάρτησης (χωρίς ερωτηματικό)

Σώμα συνάρτησης

Στο παράδειγμα:

```
double CelciusToFahrenheit(double c)
{
double f;
f = 9.0 /5.0 * c + 32.0;
return f;
}
```

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Η εντολή return καθορίζει την τιμή που πρόκειται να επιστραφεί. Στη C, οι συναρτήσεις μπορεί και να μην επιστρέφουν τιμή και η εντολή return να παραλείπεται.
- Η συνάρτηση CelciusToFahrenheit μπορεί να κληθεί από τη συνάρτηση main() ενός προγράμματος ή και από μια άλλη συνάρτηση χρησιμοποιώντας απλά το όνομά της. Μετά το όνομα της συνάρτησης και τις παρενθέσεις πρέπει να ακολουθεί το σύμβολο (;) ως εξής: CelciusToFahrenheit (double);
- Όταν το πρόγραμμα, καθώς εκτελείται, φθάσει σε μια κλήση συνάρτησης, αναζητά τη συνάρτηση αυτή και ακολουθεί τις εντολές που βρίσκονται σ' αυτήν. Όταν τελειώσει, επιστρέφει στην επόμενη γραμμή του καλούντος προγράμματος.
- Όταν γράφουμε μια συνάρτηση ακολουθούμε την ίδια μορφή που ξέρουμε από τη main(), δηλαδή πρώτα το όνομα, μετά το σύμβολο {, μετά η δήλωση των μεταβλητών, μετά οι προτάσεις ορισμού της συνάρτησης και τέλος το σύμβολο }.
- Οι παρενθέσεις στη CelciusToFahrenheit(double) λένε στο μεταγλωττιστή ότι η CelciusToFahrenheit(double) είναι μια συνάρτηση. Η CelciusToFahrenheit() δεν ακολουθείται από το σύμβολο ; πράγμα που σημαίνει ότι η συνάρτηση αυτή ορίζεται παρά ότι χρησιμοποιείται.
- Η μεταβλητή f στη CelciusToFahrenheit() είναι μια τοπική μεταβλητή, δηλαδή μια μεταβλητή γνωστή μόνο για τη CelciusToFahrenheit(). Μπορούμε να χρησιμοποιήσουμε το όνομα f και σ’ άλλες συναρτήσεις χωρίς κανένα πρόβλημα, ακόμα και μέσα στην ίδια την main() του προγράμματος.
- Οι συναρτήσεις έχουν κι αυτές τύπους όπως οι μεταβλητές. Ο τύπος void χρησιμοποιείται για συναρτήσεις χωρίς τιμές επιστροφής και ο τύπος αυτός εμφανίζεται στον ορισμό της συνάρτησης.
- Προγράμματα που χρησιμοποιούν μια συνάρτηση πρέπει να περιέχουν τη δήλωση του τύπου πριν η συνάρτηση χρησιμοποιηθεί. Γι' αυτό το λόγο, η main() θα πρέπει να περιέχει την εξής δήλωση:

  double CelciusToFahrenheit(double);

Έτσι δηλώνουμε ότι το πρόγραμμα χρησιμοποιεί μια συνάρτηση τύπου double που ονομάζεται CelciusToFahrenheit() και ότι ο μεταγλωττιστής θα βρει τον ορισμό αυτής της συνάρτησης κάπου αλλού.

- Η συνάρτηση double CelciusToFahrenheit() ή μια άλλη συνάρτηση θα μπορούσε να μην έχει είσοδο και να μη χρειάζεται καμία πληροφορία από το καλούν πρόγραμμα.

Άσκηση αυτοαξιολόγησης - S6_LA17LO156

Με βάση το παρακάτω πρόγραμμα, να απαντήσετε τα εξής:

Α. Τι θα εκτυπώσει η printf του κυρίως προγράμματος;

Β. Ποιο είναι το όνομα της συνάρτησης;

Γ. Πόσες είναι οι τυπικές παράμετροι;

Δ. Να δώσετε τα ονόματα των πραγματικών παραμέτρων.

```
#include <stdio.h>
int megalytero(int, int);
main()
{
    int x, p=5, q=4;
    x = megalytero(p, q);
    printf("\nΜεγαλύτερο των %d και %d είναι ο αριθμός %d\n", p, q, x);
}

int megalytero(int a, int b)
{
    if (a>b)
        return a;
    else
        return b;
}
```

Άσκηση αυτοαξιολόγησης - S6_LA17LO157

Με βάση το παρακάτω πρόγραμμα να απαντήσετε τα εξής:

Α. Ποιο είναι το όνομα της συνάρτησης;

Β. Να δώσετε τα ονόματα των τυπικών παραμέτρων.

Γ. Να δώσετε τα ονόματα των πραγματικών παραμέτρων.

Δ. Σε ποια γραμμή του κώδικα υπάρχει το πρωτότυπο της συνάρτησης;

Ε. Σε ποιες γραμμές του κώδικα βρίσκεται ο ορισμός (υλοποίηση) της συνάρτησης;

ΣΤ. Σε ποια γραμμή του κώδικα βρίσκεται η κλήση της συνάρτησης;

```
1.  #include <stdio.h>
2.  float compute_area(float x, float y);
3.  float compute_area(float a, float b)
4.  {
5.      return (a * b);
6.  }
7.  int main()
8.  {
9.      float length, width;
10.     float area;
11.     printf("Enter length and width: ");
12.     scanf("%f%f", &length, &width);
13.     area = compute_area(length, width);
14.     printf("The area of a rectangle with dim %f by %f m is %f sq. m\n",length, width,
        area);
15.     return(0);
16. }
```

Άσκηση αυτοαξιολόγησης - S6_LA17LO158

Να γράψετε τον κώδικα για μια συνάρτηση με όνομα akeraio_mi που θα επιστρέφει τον μικρότερο από δύο ακεραίους a και b.

Α. Ποιο το πρωτότυπο της συνάρτησης;

Β. Συμπληρώστε τα κενά:

```
if (……..)
    return….;
else
    return….;
```

Άσκηση αυτοαξιολόγησης - S6_LA17LO159

Να γράψετε μια συνάρτηση, η οποία παίρνει ως τιμή εισόδου την ακτίνα ενός κύκλου (R) και στη συνέχεια να εκτυπώνει το εμβαδόν του κύκλου με τον τύπο εμβαδό = 3.14 * R^2

ΠΑΡΑΤΗΡΗΣΗ

Πρόκειται για συνάρτηση με μία παράμετρο χωρίς τιμή εξόδου.

3.6.3 Δραστηριότητα 3 - Ειδικά θέματα συναρτήσεων

Τίτλος Δραστηριότητας	***S6_LA18*** **-** Ειδικά θέματα συναρτήσεων
Τίτλος Ενότητας	Συναρτήσεις
Εκπαιδευτική Στρατηγική	Θα χρησιμοποιηθούν θεωρία και πρακτική εφαρμογή με ασκήσεις αυτοαξιολόγησης**.**
Περιγραφή	Στη δραστηριότητα αυτή περιγράφεται η υλοποίηση της χρήσης των συναρτήσεων στη γλώσσα C σε ειδικές περιπτώσεις. Χρησιμοποιώντας δείκτες και τον τελεστή * η συνάρτηση μπορεί να εξετάσει τις τιμές που βρίσκονται σ' αυτές τις θέσεις μνήμης και να τις αλλάξει. Με τους δείκτες, δηλαδή, δίνεται η δυνατότητα σε κάθε συνάρτηση να μπορεί να αλλάξει ότι είναι αποθηκευμένο εκεί. Επίσης η C δεν επιτρέπει ολόκληροι πίνακες να είναι ορίσματα μιας συνάρτησης, αλλά για να γίνει αυτό μπορούμε να περάσουμε (μεταβιβάσουμε) τη διεύθυνση ενός πίνακα σε μια συνάρτηση. Η συνάρτηση μπορεί μετά να χρησιμοποιήσει αυτή την τιμή για το χειρισμό του πρωτότυπου πίνακα.
Γλώσσα	Ελληνικά
Μαθησιακά Αποτελέσματα	Όταν θα έχουν μελετήσει τη δραστηριότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναγνωρίζουν πότε είναι απαραίτητη η χρήση καθολικών και τοπικών μεταβλητών, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με τη χρήση καθολικών μεταβλητών, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με τη χρήση τοπικών μεταβλητών, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με ορίσματα πίνακες, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με τη χρήση δεικτών στα ορίσματά τους.
Μαθησιακά Αντικείμενα	Εμβέλεια μεταβλητών Οι δείκτες και οι συναρτήσεις Συναρτήσεις, Πίνακες και Δείκτες
Αξιολόγηση Εκπαιδευομένων	Η αξιολόγηση της δραστηριότητας γίνεται με 4 μαθησιακά αντικείμενα που είναι ασκήσεις αυτοαξιολόγησης για τη θεωρία που παρουσιάζεται στα μαθησιακά αντικείμενα S6_LA18LO160, S6_LA18LO163 και S6_LA18LO165.
Λέξεις Κλειδιά	Εμβέλεια μεταβλητών, τοπικές μεταβλητές, καθολικές μεταβλητές

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA18LO160**
Τίτλος	**Εμβέλεια μεταβλητών**
Τίτλος δραστηριότητας	Εισαγωγή στις συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι μεταβλητές και η εμβέλειά τους μέσα στις συναρτήσεις. Όταν μία μεταβλητή δηλώνεται μέσα σε μια συνάρτηση τότε ονομάζεται τοπική μεταβλητή, γιατί μπορούμε να τη χρησιμοποιήσουμε τοπικά, μόνο μέσα στο σώμα της συνάρτησης και όχι έξω από αυτή. Εάν δηλώνεται στην αρχή της συνάρτησης, τότε μπορούμε να τη χρησιμοποιήσουμε οπουδήποτε μέσα στο σώμα αυτής. Όταν μία μεταβλητή δηλώνεται έξω από τις συναρτήσεις στην αρχή του προγράμματος τότε ονομάζεται καθολική μεταβλητή γιατί μπορούμε να τη χρησιμοποιήσουμε οπουδήποτε και μέσα σε όλες τις συναρτήσεις.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Εμβέλεια μεταβλητών, τοπικές μεταβλητές, καθολικές μεταβλητές
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με τη χρήση καθολικών μεταβλητών, • να δίνουν παραδείγματα συναρτήσεων της γλώσσας C με τη χρήση τοπικών μεταβλητών.

Σε κάθε συνάρτηση χρησιμοποιούνται δύο ειδών μεταβλητές: οι τυπικές παράμετροι και οι εσωτερικές ή τοπικές μεταβλητές (local variables). Για παράδειγμα στην επόμενη συνάρτηση:

```
#include <stdio.h>
int mikrotero (int, int);
main()
{
    int x, p=5, q=4;
    x = mikrotero (p, q);
```

```
    printf("\nΜικρότερο των %d και %d είναι ο αριθμός %d\n", p, q, x);
}

int mikrotero(int a, int b)
{
    int t;
    if (a< b)
        t = a;
    else
        t = b;
    return t;
}
```

Οι **τυπικές παράμετροι** είναι οι ακέραιες μεταβλητές a και b, ενώ **τοπική μεταβλητή** της συνάρτησης είναι η ακέραια μεταβλητή t. Ότι ισχύει με όλες τις συναρτήσεις ισχύει και για τη main. Δηλαδή, στο προηγούμενο παράδειγμα οι μεταβλητές x, p, και q αποτελούν το σύνολο των τοπικών μεταβλητών της main. Η εμβέλεια (scope) μίας μεταβλητής περιορίζεται στο σώμα της συνάρτησης που θα δηλωθεί ή, αν αποτελεί παράμετρο, τότε πάλι στο σώμα της συνάρτησης στην οποία είναι παράμετρος. Με τον όρο **εμβέλεια** εννοείται πού ακριβώς η μεταβλητή είναι ορατή, δηλαδή πού μπορεί να χρησιμοποιηθεί. Στην περίπτωση που δύο μεταβλητές ακόμη και με ίδιο όνομα και τύπο έχουν δηλωθεί σαν τοπικές δύο συναρτήσεων τότε η C αντιμετωπίζει σαν εντελώς διαφορετικές και ξένες μεταξύ τους. Βέβαια εδώ τίθεται το εξής ερώτημα: τι θα γινόταν αν μία μεταβλητή έπρεπε να είναι ορατή σε όλο το πρόγραμμα και σε όλες τις συναρτήσεις του; Για να συμβεί αυτό θα πρέπει η μεταβλητή να δηλωθεί στην αρχή του προγράμματος (αμέσως μετά τις εντολές προεπεξεργαστή include και define). Οι μεταβλητές αυτού του τύπου ονομάζονται **καθολικές μεταβλητές** (global variables). Για παράδειγμα, μια μεταβλητή k, ακριβώς επειδή είναι καθολική μεταβλητή μπορεί να χρησιμοποιηθεί σε όλες τις συναρτήσεις του προγράμματος.

Άσκηση αυτοαξιολόγησης - S6_LA18LO161

Με βάση τον παρακάτω κώδικα να απαντήσετε στα ερωτήματα:

Α. Ποιες είναι οι τοπικές μεταβλητές στην κύρια συνάρτηση;

Β. Ποιες είναι οι τοπικές μεταβλητές στην συνάρτηση piliko();

Γ. Υπάρχουν και ποιες είναι καθολικές μεταβλητές;

Δ. Ποιες είναι οι παράμετροι της συνάρτησης piliko();

```
#include <stdio.h>
int N = 10;
int megalytero (int, int);
main()
{
    int x, p, i;
    for (i=1; i<=N; i++)
    {
        printf("\nΕισαγωγή αριθμού: ");
```

```
        scanf("%d", &p);
        x = piliko(p) ;
        printf("\nΤο πηλίκο του %d με %d είναι ο αριθμός %d\n", p, N, x);
    }
}

int piliko(int a)
{
    int t;
    t=b / N;
    return t;
}
```

Άσκηση αυτοαξιολόγησης - S6_LA18LO162

Με βάση τον παρακάτω κώδικα να απαντήσετε στα ερωτήματα:

Α. Ποιες είναι οι τοπικές μεταβλητές στην κύρια συνάρτηση;

Β. Ποιες είναι οι τοπικές μεταβλητές στη συνάρτηση sum();

Γ. Υπάρχουν και ποιες είναι καθολικές μεταβλητές;

Δ. Υπάρχουν και ποιες είναι καθολικές μεταβλητές σταθερές;

Ε. Ποιες είναι οι τυπικές παράμετροι;

ΣΤ. Ποιες είναι οι πραγματικές παράμετροι;

Ζ. Τι τυπώνει το πρόγραμμα;

```
#include <stdio.h>
#defined        5
int c=5;
int sum(intx, inty)
{
    int c = 1;
    return(x + y + c);
}
int main()
{
    int a = 5, b=6, c=11;
    printf("%d+%d=%d", a, b, sum(a, b));
    return 0;
}
```

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA18LO163**
Τίτλος	**Οι Δείκτες και οι Συναρτήσεις**
Τίτλος δραστηριότητας	Εισαγωγή στις Συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι συναρτήσεις.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Κλήση με τιμή, κλήση με αναφορά
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρονται σε συγκεκριμένο στοιχείο πίνακα, • να γνωρίζουν την αναγκαιότητα της χρήσης των πινάκων, • να δώσουν τουλάχιστον τρία παραδείγματα δήλωσης πινάκων διαφορετικού τύπου και μεγέθους.

Εκείνο που μας ενδιαφέρει είναι η μετάδοση των τιμών που επιστρέφει μια συνάρτηση πίσω στη main(). Όπως ξέρουμε, με την πρόταση return μπορούμε να μεταδώσουμε πίσω στην καλούσα συνάρτηση μόνο μία τιμή. Για να μεταδώσουμε δύο τιμές μιας συνάρτησης στη συνάρτηση που την κάλεσε, πρέπει να χρησιμοποιήσουμε τους δείκτες, που είναι συμβολικές παραστάσεις διευθύνσεων.

Ακολουθεί ένα πρόγραμμα όπου γίνεται επιστροφή τιμών στη συνάρτηση που καλεί.

```
/* prog40.c - χρήση δεικτών για την ανταλλαγή τιμών */
#include <stdio.h>
void interchange();
main()
{
    int x = 5, y = 10;
    printf("Αρχικά x = %d και y = %d. \n", x, y);
    interchange(&x, &y); /* στείλε τις διευθύνσεις στη συνάρτηση */
    printf("Τώρα x = %d και y = %d. \n", x, y);
}
```

```
void interchange(u, v)
int *u, *v; /* οι u και v είναι τώρα δείκτες */
{
    int temp;
    temp = *u; /* η temp παίρνει την τιμή που δείχνει η u */
    *u = *v;
    *v = temp;
}
```

Το αποτέλεσμα θα είναι:

Αρχικά x = 5 και y = 10

Τώρα x = 10 και y = 5

Ας προσέξουμε τα εξής σημεία:

1. Η κλήση της συνάρτησης γίνεται ως εξής: interchange(&x, &y) δηλαδή, αντί να στέλνονται οι τιμές των x και y, στέλνονται οι διευθύνσεις τους. Έτσι, τα τυπικά ορίσματα u και v της συνάρτησης interchange() έχουν διευθύνσεις ως τιμές και πρέπει να δηλωθούν ως δείκτες.

2. Εφόσον οι x και y είναι ακέραιοι, δηλώνουμε τις u και v ως δείκτες σε ακέραιους.

3. Θυμηθείτε ότι η u έχει την τιμή &x, έτσι η u δείχνει τη x, δηλαδή η *u μας δίνει την τιμή του x.

4. Δεν πρέπει να γράψουμε temp = u; γιατί έτσι θα αποθηκευτεί η διεύθυνση της x και όχι η τιμή της.

Θέλουμε μια συνάρτηση να ανταλλάσσει τις τιμές δύο μεταβλητών. Δίνοντας στη συνάρτηση τις διευθύνσεις των μεταβλητών αυτών, αποκτάμε πρόσβαση σ' αυτές τις μεταβλητές. Χρησιμοποιώντας δείκτες και τον τελεστή * η συνάρτηση μπορεί να εξετάσει τις τιμές που βρίσκονται σ' αυτές τις θέσεις και να τις αλλάξει.

Με τους δείκτες, δηλαδή, μπορούμε να μπούμε στη main() και να αλλάξουμε ότι είναι αποθηκευμένο εκεί.

Άσκηση αυτοαξιολόγησης - S6_LA18LO164

Όταν περάσουμε μια μεταβλητή (π.χ. int, char, float) σε μια συνάρτηση, τότε δημιουργείται ένα αντίτυπο της μεταβλητής, το οποίο δεν έχει καμία σχέση με την αρχική μεταβλητή. Αυτός ο τρόπος περάσματος παραμέτρων ονομάζεται «κλήση με τιμή». Αντίθετα στην «κλήση με αναφορά» δεν δημιουργείται αντίγραφο αλλά οι αλλαγές γίνονται απευθείας στη διεύθυνση μνήμης της μεταβλητής.

Με βάση όσα αναφέρθηκαν να απαντήσετε τι εκτυπώνει η παρακάτω συνάρτηση:

Α. Όταν γίνεται κλήση με τιμή

Β. Όταν γίνεται κλήση με αναφορά

Κλήση με Τιμή	Κλήση με Αναφορά

int change(int a)
{
a = 5;
}
 	
int main()
{
int a = 1;
change(a);
printf("%d", a);
}
	int change(int *a)
	{
	(*a) = 5;
	}
	
	int main()
	{
	int a = 1;
	change(&a);
	printf("%d", a);
	}

Μαθησιακό Αντικείμενο	
Όνομα	**S6_LA18LO165**
Τίτλος	**Συναρτήσεις, Πίνακες και Δείκτες**
Τίτλος δραστηριότητας	Εισαγωγή στις Συναρτήσεις
Περιγραφή	Στο συγκεκριμένο ΜΑ περιγράφονται οι συναρτήσεις σε σχέση με ορίσματα πίνακες. Μέχρι τώρα είδαμε παραδείγματα με επιστροφή μόνο μιας τιμής από συνάρτηση. Τι γίνεται όμως αν θέλουμε να αλλάξουμε τα περιεχόμενα ενός πίνακα μέσα από συνάρτηση; Μία από τις σημαντικές χρήσεις των δεικτών είναι ότι επιτρέπουν τη μεταβίβαση ορισμάτων σε συναρτήσεις με αναφορά (by reference). Η μέθοδος μεταβίβασης παραμέτρων με αναφορά (by reference) βασίζεται στην ιδέα ότι ο κώδικας (σώμα) της συνάρτησης λειτουργεί σαν να λαμβάνει και χρησιμοποιεί την ίδια την μεταβλητή του ορίσματος. Αυτό σημαίνει ότι η τιμή της μεταβλητής του ορίσματος στο σημείο κλήσης μπορεί να είναι διαφορετική μετά την επιστροφή της συνάρτησης, εάν η συνάρτηση στο σώμα της μεταβάλλει την τιμή της παραμέτρου που αντιστοιχεί στο όρισμα που μεταβιβάζεται με αναφορά.
Γλώσσα	Ελληνικά
Μαθησιακός Τύπος	Θεωρία
Τεχνικός Τύπος	Υπερκείμενο, βιντεοδιάλεξη
Λέξεις Κλειδιά	Μεταβίβαση ορισμάτων σε συναρτήσεις με αναφορά, μεταβίβαση ορισμάτων σε συναρτήσεις με τιμή
Μαθησιακά Αποτελέσματα (ΜΑπ)	Όταν θα έχουν μελετήσει την ενότητα αυτή, οι εκπαιδευόμενοι θα μπορούν: • να αναφέρονται σε συγκεκριμένο στοιχείο πίνακα, • να γνωρίζουν την αναγκαιότητα της χρήσης των πινάκων, • να δώσουν τουλάχιστον τρία παραδείγματα δήλωσης πινάκων διαφορετικού τύπου και μεγέθους.

Υπάρχουν δύο μέθοδοι μεταβίβασης ορισμάτων στις συναρτήσεις

- Μεταβίβαση τιμής - ανάθεση (by value)
- Μεταβίβαση με αναφορά - απ’ ευθείας χρήση (by reference)

Η μέθοδος με μετάδοση τιμής (by value) βασίζεται στην ιδέα ότι ό κώδικας (σώμα) της συνάρτησης λαμβάνει και χρησιμοποιεί ένα αντίγραφο του ορίσματος, το οποίο ισχύει όσο η συνάρτηση εκτελείται. Αυτό σημαίνει ότι η τιμή της μεταβλητής του ορίσματος στο σημείο κλήσης είναι η ίδια και μετά την επιστροφή της συνάρτησης μιας και μόνο ένα αντίγραφό της χρησιμοποιείται. Χαρακτηριστικό παράδειγμα είδαμε με την προηγούμενη άσκηση αυτοαξιολόγησης.

Η μέθοδος μεταβίβασης με αναφορά (by reference) βασίζεται στην ιδέα ότι ο κώδικας (σώμα) της συνάρτησης λειτουργεί σαν να λαμβάνει και χρησιμοποιεί την ίδια τη μεταβλητή του ορίσματος. Αυτό σημαίνει ότι η τιμή της μεταβλητής του ορίσματος στο σημείο κλήσης μπορεί να είναι διαφορετική μετά την επιστροφή της συνάρτησης, εάν η συνάρτηση στο σώμα της μεταβάλλει την τιμή της παραμέτρου που αντιστοιχεί στο όρισμα που μεταβιβάζεται με αναφορά.

Μία από τις σημαντικές χρήσεις των δεικτών είναι ότι επιτρέπουν τη μεταβίβαση ορισμάτων σε συναρτήσεις με αναφορά (by reference).

Αυτό σημαίνει ότι, αντί να περνάμε ένα αντίγραφο του ορίσματος στην αντίστοιχη παράμετρο, περνάμε τη διεύθυνση του ορίσματος. Οπότε, κάθε αλλαγή στην παράμετρο (στο σώμα της συνάρτησης) ουσιαστικά μεταβάλλει τα περιεχόμενα στη θέση μνήμης του ορίσματος!

Η C δεν επιτρέπει ολόκληροι πίνακες να είναι ορίσματα μιας συνάρτησης, αλλά για να το επιτύχουμε αυτό μπορούμε να περάσουμε (μεταβιβάσουμε) τη διεύθυνση ενός πίνακα σε μια συνάρτηση. Η συνάρτηση μπορεί μετά να χρησιμοποιήσει αυτή την τιμή για το χειρισμό του πρωτότυπου πίνακα.

Είναι ένα κλασσικό παράδειγμα όπου πρέπει να χρησιμοποιήσουμε δείκτες για το χειρισμό ενός πίνακα. Μπορούμε να χρησιμοποιήσουμε είτε συμβολισμό πίνακα είτε συμβολισμό δείκτη μέσα στη συνάρτηση.

Ακολουθεί ένα παράδειγμα με μια συνάρτηση που επιστρέφει το άθροισμα των στοιχείων ενός πίνακα.

```
#include <stdio.h>

#defineSIZE    10

long sum();

main()

{

    int pina[SIZE] = {20, 10, 5, 39, 4, 16, 19, 26, 31, 20};

    long answer;


    answer = sum(pina, SIZE);

    printf("Το άθροισμα των στοιχείων είναι %d.\n", answer);

}
```

```
long sum(ar, n)          /* χρήση δείκτη */
int *ar;                 /* ο ar είναι ένας δείκτης */
int n;
{
    int i;
    long total = 0;
    for (i=0; i<n; i++)
    {
        total += *ar;    /* πρόσθεση τιμής στο total */
        ar++;            /* ο δείκτης στο επόμενο στοιχείο */
    }
    return total;
}
```

Ο ar ξεκινάει δείχνοντας το πρώτο στοιχείο του πίνακα pina και με την πρόταση καταχώρησης total += *ar; προστίθεται η τιμή του πρώτου στοιχείου, δηλαδή η 20, στη μεταβλητή total. Μετά, η έκφραση ar++ αυξάνει τη μεταβλητή δείκτη ar έτσι ώστε αυτή να δείχνει στο επόμενο στοιχείο του πίνακα.

Μπορούμε να γράψουμε τις προηγούμενες δύο εντολές και έτσι:

```
total += *ar++;
```

Δηλαδή, αυξάνεται κατά ένα ο δείκτης και όχι η τιμή στην οποία δείχνει και μάλιστα αυξάνεται μετά τη χρήση της τιμής στην οποία δείχνει. Εάν χρησιμοποιήσουμε *++ar, τότε η σειρά θα ήταν: αύξηση του δείκτη και μετά χρήση της τιμής στην οποία δείχνει. Αλλά αν χρησιμοποιήσουμε την έκφραση (*ar)++ τότε το πρόγραμμα θα κάνει χρήση της τιμής που δείχνει η ar και μετά θα αυξήσει την τιμή αυτή κατά ένα και όχι το δείκτη.

Άσκηση αυτοαξιολόγησης - S6_LA18LO166

Έστω ο παρακάτω κώδικας, ο οποίος επιστρέφει με τη συνάρτηση max_min το μέγιστο και το ελάχιστο στοιχείο ενός πίνακα.

Να συμπληρωθούν σωστά τα κενά σε 11 αριθμημένα σημεία του προγράμματος, ώστε να γίνει επιστροφή των δύο τιμών από τη συνάρτηση.

```
#include <stdio.h>
#define N 5
void max_min(int ……1……., int n, int *max, int *min);
main()
{
    int b[N], i, high, low;
    printf("Enter %d numbers: ", N);
```

```
    for (i=0; ……..2…….; i++)
    scanf("%d", …….3……);
    max_min(b, N, ……4……, ……..5…….);
    printf("The largest number is: %d \n", …….6……..);
    printf("The smallest number is: %d ", …….7…….);
    getchar();
    getchar();
    return 0;
}

void max_min(int a[], int size, int …….8……., int…….9……)
{
    int i;
    *max = *min = a[0];
    for(i = 1; i < size; i++)
    {
        if (a[i] > *max)
            …….10…….= a[i];
        else if (a[i] < *min)
            ……11…….= a[i];
    }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

Αχιλλέας Αχιλλέως, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, Εισαγωγή στον Προγραμματισμό για ΗΜΥ. Ανακτήθηκε από http://www.cs.ucy.ac.cy/courses/EPL034/slides/w4.pdf

Γεωργακόπουλος Γ, Παπαδόγγονας Ι, (2009) "Σύγχρονες Γλώσσες προγραμματισμού"

Γουρζής Σ, (1999) "Ιστορία του προγραμματισμού των Η/Υ". Ανακτήθηκε από http://www.slideshare.net/stathis13061965/ss-9071594

Γκουμόπουλος Χ, Βασικά στοιχεία της C, Πανεπιστήμιο Αιγαίου. Ανακτήθηκε από http://www.icsd.aegean.gr/lessons/intro2prog/lectures/04_i2p_C_basics.pdf

Καρακαπιλίδης, Ν., Top Down Σχεδιασμός. Ανακτήθηκε από http://www.mech.upatras.gr/~nikos/progr/notes/notes-03.pdf

Κέντρο ΠΛΗ.ΝΕ.Τ. Ν. Φλώρινας, "Η Γλώσσα Προγραμματισμού C". Ανακτήθηκε από http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-C-Part-4.html

Θραμπουλίδης Κ, (2000) "Γλώσσες προγραμματισμού", ΕΑΠ

Μαστοροκώστας Π, (2006) "Διαδικαστικός Προγραμματισμός", ΤΕΙ Σερρών

Μπαλαούρας Π, Τσιμπάνης Κ, "Ανοικτά ψηφιακά μαθήματα στα ελληνικά ΑΕΙ", Ανακτήθηκε 16/4/2015 από ***https://learn20.wikispaces.com/file/view/MOOCs-Karakostas-οκ.docx***

Μισυρλής Ν, (2003) "Εισαγωγή στον Προγραμματισμό με την C", Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών.

Παγγέ, Τ. (2009) "Εκπαιδευτική Τεχνολογία". Ιωάννινα, εκδόσεις Θεοδωρίδη.

Πάρρυ Β, Συνοπτικός οδηγός αναφοράς συναρτήσεων της C. Ανακτήθηκε από http://cgi.di.uoa.gr/~ip/Cfunctionsref.html

Πανεπιστήμιο Αιγαίου, "Συναρτήσεις και διαδικασίες". Ανακτήθηκε από http://www.samos.aegean.gr/math/andpapas/courses/ics/presentations/functions-procedures.pdf

Πανεπιστήμιο Αιγαίου, "Αλφαριθμητικά, πίνακες και δείκτες". Ανακτήθηκε από http://www.samos.aegean.gr/math/andpapas/courses/pl/presentations/strings-and-pointers.pdf

Πανεπιστήμιο Αιγαίου, Εισαγωγή στις γλώσσες προγραμματισμού με τη γλώσσα C. Ανακτήθηκε από http://myria.math.aegean.gr/epeaek/pdfs/C.pdf

Πανεπιστήμιο Κρήτης, Τμημα Επιστήμης και Τεχνολογίας Υλικών. Ανακτήθηκε από https://www.materials.uoc.gr/el/undergrad/courses/ETY215/node5.html

Πανεπιστήμιο Κύπρου Τμήμα Πληροφορικής, "Πίνακες και Συναρτήσεις". Ανακτήθηκε από http://www.cs.ucy.ac.cy/courses/EPL032/epl032-1/lectures/lecture%2018.pdf

Παπαδάκης, Στ., & Καλογιαννάκης, Μ. (2014). MOOC «Massive Open Online Courses»: Μια πρώτη επισκόπηση του πεδίου. Νέος Παιδαγωγός, 2(1), 51-58 Ανακτήθηκε 16/4/2015 από http://neospaidagogos.gr/periodiko/writers/pdfs2/04.massive_open_online_courses..pdf

Παπαρρίζου Α, "Εισαγωγή στο Δομημένο Προγραμματισμό". Ανακτήθηκε από http://users.uowm.gr/apaparrizou/Cprogramming.html

Πιερράτος Θ, "Εισαγωγή στον προγραμματισμό". Ανακτήθηκε από http://users.sch.gr/pierratos/anaptixi/theoria6.pdf

Πάρρυ Β, "Συνοπτικός οδηγός αναφοράς συναρτήσεων της C". Ανακτήθηκε από http://cgi.di.uoa.gr/~ip/Cfunctionsref.html

Πολυχρονόπουλος Ε, (2008) "Εισαγωγή στο Διαδικαστικό Προγραμματισμό C". Ανακτήθηκε από http://software.hpclab.ceid.upatras.gr/c_notes.pdf

Σάββας Η, (2005) "ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΙΙ", ΤΕΙ Λάρισας. Ανακτήθηκε από http://www.teilar.gr/dbData/ProfAnn/profann-263766d4.pdf

Σεφερίδης Β, (1995) "C για Αρχάριους", Εκδόσεις Κλειδάριθμος, Ανατύπωση 2001

Σταυρινού Ιωάννου Μ, "Συμβολοσειρές και πίνακες", Πανεπιστήμιο Κύπρου - Τμήμα Πληροφορικής. Ανακτήθηκε από http://www.cs.ucy.ac.cy/~dzeina/courses/epl032/labs/lab09.pdf

Στυλιάδη Κ., (1997) " Έτοιμες Ασκήσεις σε C". Ανακτήθηκε από http://dide.flo.sch.gr/Plinet/Tutorials/C-Exercises.pdf

ΤΕΙ Κεντρικής Μακεδονίας, Τμήμα Πληροφορικής, Συναρτήσεις. Ανακτήθηκε από http://www.teicm.gr/repository/repository/sygk_yliko/didaktiko%20yliko%20gia%20pylh/Tmhma_Plhroforikhs_Epik/Mastorokwstas/Programmatismos_II/ue_1.pdf

Bichelmeyer, Ph.D. The ADDIE Model - A Metaphor for the Lack of Clarity in the field of IDT, Indiana University. Ανακτήθηκε από http://www.indiana.edu/~idt/shortpapers/documents/IDTf_Bic.pdf

Breslow, L., Pritchard, D. E., DeBoer, J., Stump, G. S., Ho, A. D., & Seaton, D. T. (2013). Studying learning in the worldwide classroom: Research into edX's first MOOC. Research & Practice in Assessment, 8, 13-25.

Chen, X., Barnett, D. and Stephens, C. (2013). Fad of Future: The Advantages and Challenges of Massive Open Online Courses (Moocs). Ανακτήθηκε από

http://www.lindenwood.edu/r2p/docs/ChenBarnettStephens.pdf

Interactive C User's Guide. *Ανακτήθηκε από* ***http://www.newtonlabs.com/ic/ic_toc.html***

Liyanagunawardena, T., Adams, A. & Williams, S. (2013). MOOCs: A Systematic Study of the Published Literature 2008-2012. The International Review of Research in Open and Distance Learning, 14(3), 202-227

MacKay, R.F. (April 11, 2013). "Learning analytics at Stanford takes huge leap forward with MOOCs". Stanford Report. Stanford University. Retrieved June 22, 2013.

Marshall, S. (2013). Evaluating the Strategic and Leadership Challenges of MOOCs. MERLOT Journal of Online Learning & Teaching, 9(2), 216-227.

McAuley, A., Stewart, B., Siemens, G., & Cormier, D. (2010). The MOOC model for digital practice, 1-63. Ανακτήθηκε από http://www.elearnspace.org/Articles/MOOC_Final.pdf

McAuley, A., Stewart, B., Siemens, G., & Cormier, D. (2010). The MOOC model for digital practice, 1-63. Ανακτήθηκε από http://www.elearnspace.org/Articles/MOOC_Final.pdf

MOOC (Massive Open Online Courses): "Μια νέα πρόκληση στη σύγχρονη διαδικτυακή εκπαίδευση". Ανακτήθηκε 16/4/2015 από

http://dide-peiraia.att.sch.gr/plinetp/images/stories/files/newsletter/131/MOOC.pdf

Schildt H, (2011). " Οδηγός της C", Third Edition

Schildt H, (2000) “C: The Complete Reference”, Fourth Edition, McGraw Hill

The C Book (1991) Ανακτήθηκε από http://publications.gbdirect.co.uk/c_book/

Yuan, L.,& Powell, S. (2013). MOOCs and Open Education: Implications for Higher Education. A white paper. Ανακτήθηκε από http://publications.cetis.ac.uk/wp-content/uploads/2013/03/MOOCs-and-Open-Education.pdf

Wikipedia, (2015) “Massive Open Online Course”. Ανακτήθηκε 16/4/2015 από http://en.wikipedia.org/wiki/Massive_open_online_course

Wikipedia, (2015) “ADDIE Model”. Ανακτήθηκε 16/4/2015 από http://en.wikipedia.org/wiki/ADDIE_Model

Zhang T., (2000) “Μάθετε τη C σε 24 Ώρες”, Δεύτερη Έκδοση, Γκιούρδας.

ΠΑΡΑΡΤΗΜΑ Ι

Πρότυποι πίνακες

Πίνακας 1. Περιγραφή Μαθήματος (1η φάση: Ανάλυση)

Τίτλος Μαθήματος	Τίτλος Μαθήματος
Περιγραφή	Συνοπτική Περιγραφή του μαθήματος
Γνωστικό Πεδίο	Γνωστικό Πεδίο στο οποίο αναφέρεται το μάθημα
Τύπος Μαθήματος	Καταγραφή του τύπου του μαθήματος (Full time ή Part time)
Συνολικός Χρόνος Μαθήματος	Καταγραφή του προβλεπόμενου συνολικού χρόνου του μαθήματος
Εκπαιδευτικό πρόβλημα	Αναφορά του εκπαιδευτικού προβλήματος που επιδιώκει να επιλύσει
Χαρακτηριστικά και ανάγκες Εκπαιδευομένων	Περιγραφή των χαρακτηριστικών και αναγκών των εκπαιδευομένων
Έννοιες	Περιγραφή των βασικών εννοιών που πρόκειται να διδαχθούν
Ενότητες Μάθησης	Καταγραφή των ενοτήτων μάθησης που θα διδαχθούν
Εκπαιδευτικοί Στόχοι	Περιγραφή των κύριων εκπαιδευτικών στόχων που επιδιώκει να επιλύσει
Εργαλεία	Περιγραφή των εργαλείων που θα χρησιμοποιηθούν στο μάθημα
Απαιτήσεις του Περιβάλλοντος	Αναφορά σε τυχόν ιδιαίτερες απαιτήσεις του περιβάλλοντος που θα παραδοθεί γνώση
Εμπλεκόμενοι Ρόλοι	Περιγραφή των εμπλεκόμενων ρόλων και των ενεργειών τους
Χρονοπρογραμματισμός	Περιγραφή του χρονοπρογραμματισμού στον οποίο θα καθορίζεται η σειρά και ο χρόνος εκτέλεσης των απαιτούμενων εργασιών για την υλοποίηση των μαθημάτων

Πίνακας 2α. Περιγραφή Ενοτήτων Μάθησης (Φάση: Σχεδιασμός)

Τίτλος Ενότητας Μάθησης	Τίτλος Ενότητας Μάθησης
Τίτλος Μαθήματος που ανήκει	Τίτλου μαθήματος που ανήκει
Περιγραφή	Περιγραφή της ενότητας μάθησης
Εκπαιδευτικοί Στόχοι	Περιγραφή των εκπαιδευτικών στόχων της ενότητας
Μαθησιακά Αποτελέσματα (ΜΑπ)	Περιγραφή των Μαθησιακών Αποτελεσμάτων της ενότητας
Εκπαιδευτικές Δραστηριότητες	Περιγραφή των εκπαιδευτικών δραστηριοτήτων που θα πλαισιώνουν τα Μ.Α.
Εμπλεκόμενοι Ρόλοι	Περιγραφή των εμπλεκόμενων ρόλων και των ενεργειών τους
Αξιολόγηση	Περιγραφή των μέσων και εργαλείων αξιολόγησης των εκπαιδευομένων

Πίνακας 2β. Περιγραφή Εκπαιδευτικών Δραστηριοτήτων (Φάση: Σχεδιασμός)

Τίτλος Δραστηριότητας	Τίτλος Δραστηριότητας
Τίτλος Ενότητας που ανήκει	Τίτλος ενότητας που ανήκει
Εκπαιδευτική Στρατηγική	Περιγραφή της εκπαιδευτικής στρατηγικής που θα χρησιμοποιηθεί (π.χ. Παρουσίαση, Παιχνίδι Ρόλων - role playing, Πρακτική Εφαρμογή κλπ.)
Περιγραφή	Περιγραφή της εκπαιδευτικής Δραστηριότητας
Γλώσσα	Αναφορά της Γλώσσας της εκπαιδευτικής δραστηριότητας
Μαθησιακά Αποτελέσματα	Περιγραφή των Μαθησιακών Αποτελεσμάτων που ικανοποιεί η εκπαιδευτική δραστηριότητα
Μαθησιακά Αντικείμενα	Αναφορά των Μ.Α. που απαιτούνται
Αξιολόγηση Εκπαιδευομένων	Περιγραφή των μέσων και εργαλείων αξιολόγησης εκπαιδευομένων
Λέξεις Κλειδιά	Λέξεις - Κλειδιά

Πίνακας 2γ. Μαθησιακό Αντικείμενο (Φάση: Σχεδιασμός)

Μαθησιακό Αντικείμενο	
Όνομα	
Τίτλος	**Τίτλος Μαθησιακού Αντικειμένου**
Τίτλος Εκπαιδευτικής Δραστηριότητας που ανήκει	**Τίτλος Εκπαιδευτικής Δραστηριότητας**
Περιγραφή	Περιγραφή Μαθησιακού Αντικειμένου
Γλώσσα	Αναφορά της γλώσσας του Μαθησιακού Αντικειμένου
Μαθησιακός Τύπος	Αναφορά του Μαθησιακού Τύπου
Τεχνικός Τύπος	Αναφορά του Τεχνικού Τύπου
Λέξεις Κλειδιά	Λέξεις - κλειδιά
Μαθησιακά Αποτελέσματα (ΜΑπ)	Αναφορά στα μαθησιακά αποτελέσματα που συνεισφέρει

ΠΑΡΑΡΤΗΜΑ ΙΙ

Λύσεις των ασκήσεων

S1_LA1LO2

1	2	3	4	5	6	7	8	9	10
Λ	Λ	Σ	Σ	Λ	Σ	Σ	Σ	Σ	Σ

S1_LA1LO3

Ιδεατό-ανάλυσης-σχεδιασμού-υλοποίησης

S1_LA1LO4

Τα βήματα λοιπόν ανάπτυξης ενός προγράμματος είναι:

1. Διατύπωση του προβλήματος με σαφήνεια. Ένα πρόβλημα δεν μπορεί να επιλυθεί ορθά αν δεν γίνει πρώτα κατανοητό.
2. Ανάλυση του προβλήματος και προσδιορισμός της λύσης. Ένα πρόβλημα μπορεί να έχει πολλές λύσεις.
3. Διατύπωση της λύσης υπό τη μορφή αλγόριθμου.
4. Κωδικοποίηση του αλγόριθμου. Επιλογή γλώσσας προγραμματισμού.
5. Μετάφραση και εκτέλεση του προγράμματος στον υπολογιστή.
6. Έλεγχος και διόρθωση συντακτικών και λογικών λαθών.

S1_LA1LO5

καθορισμένη, εκτελέσιμων, πεπερασμένο, **γλώσσα**, **προγραμματισμού**, διεργασίες

S1_LA2LO7

1	2	3	4	5	6	7	8	9	10	11	12	13
Σ	Λ	Σ	Λ	Λ	Λ	Λ	Σ	Λ	Σ	Λ	Λ	Σ

S1_LA2LO8

μεταγλωττιστής, συντακτικό, λογικά

S1_LA2LO9

γ

S1_LA2L10

5,1,6,3,2,8,4,7

S1_LA2L11

συντακτικό, συντακτικά, αλγόριθμο, λογικό

S1_LA3LO13

1	2	3	4	5	6	7	8	9	10	11	12
Σ	Λ	Σ	Λ	Σ	Λ	Λ	Σ	Σ	Σ	Σ	Λ

S1_LA3LO14

1-5 2-1 3-2 4-3

S1_LA3LO16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Λ	Σ	Σ	Σ	Σ	Λ	Σ	Σ	Σ	Σ	Λ	Λ	Σ	Σ	Σ	Σ	Σ	Λ

S1_LA3LO17

Γ

S1_LA3LO18

1- I 2- II 3- III 4- VII 5- VII 6- VI 7- VI 8- IV 9- IV 10- V 11- III

S1_LA3LO20

70, δομημένου, χαμηλού, υψηλού, χαμηλού, ANSI

S1_LA3LO21

Γ

S1_LA3LO22

Δ

S2_LA4LO21

1-α 2-α 3-α 4-α 5-α 6-β 7-β 8-β

S2_LA4LO22

1	2	3	4	5	6	7	8	9	10
ακέραιος	-	κινητής υποδιαστολής απλής ακρίβειας	κινητής υποδιαστολής απλής ακρίβειας	ακέραιος	χαρακτήρας	κινητής υποδιαστολής διπλής ακρίβειας	-	λογικός	λογικός

S2_LA4LO24

1-α 2-α 3-α 4-β 5-β

S2_LA4LO25

```
unsigned int PORTA ;
int up_down;
float heat ;
double max_read ;
```

S2_LA5LO27

1	2	3	4	5	6	7	8	9	10
Λ	1	Σ	Λ	1	Σ	5	Σ	Σ	4

S2_LA5LO28

3

S2_LA5LO29

1-2, 2-1, 3-5, 4-3, 5-4

S2_LA5LO30

1-α 2-α 3-β 4-α 5- β 6-α

S2_LA5LO32

1	2	3	4	5	6	7	8	9	10
Σ	Σ	Λ	Λ	Σ	Σ	Σ	Σ	Σ	Λ

S2_LA5LO34

1	2	3	4	5	6	7	8	9	10	11	12	13
Σ	Λ	Σ	Σ	Σ	Σ	Λ	Λ	Σ	Λ	Λ	Λ	Λ

S2_LA5LO35

```
// Δήλωση σταθερής τιμής με define
#define FPA_PERCENT 0.21
#include<stdio.h>
int main()
{
// Δήλωση σταθερής τιμής με const
const float fpa_percent = 0.21;
}
```

S2_LA5LO37

Οι 14 εντολές εκτύπωσης θα είναι:

1. i=123	8. i=+0000123
2. i=173	9. f=3.141593
3. i=7b	10. f=3.142
4. i=7B	11. f=+3.142
5. i=+123	12. f=3.14159
6. i=123	13. f=3.14159
7. i=00000123	14. f=3.141593e+000

S2_LA5LO38

1- Χαρακτήρας 2- Χαρακτήρας 3- Χαρακτήρας 4- οκταδικός ακέραιος

5- δεκαεξαδικός ακέραιος 6- ακέραιος 7- κινητής υποδιαστολής 8- κινητής υποδιαστολής

S2_LA5LO39

```
The value of index is: 13
The value of num is: 27.566999
Both index and num:
Index -> 13 and num -> 27.566999
```

S2_LA5LO40

```
1. 5
2. The value of variable index is:5
3. 5+3=8
```

4. The value of variable index is:
   5
5. x=5
   y=3
   x+y=8

S2_LA5LO41

ΠΡΟΓΡΑΜΜΑ

```
#include <stdio.h>
void main() {
int x=5;
float y=5;
float z=5.01234567890123456789;
char w='N';
printf("O akeraios arithmos x exei timh: %d\n", x);
printf("O pragmatikos arithmos y exei timh: %f\n", y);
printf("O pragmatikos arithmos z exei timh: %f\n", z);
printf("O pragmatikos arithmos z exei timh: %.2f\n", z);
printf("O xaraktiras einai o %c\n", w);
}
```

S2_LA5LO42

1	2	3	4	5
1	2	4	1	4

S2_LA6LO44

Έστω ότι οι τιμές που δίνει ο χρήστης είναι 5 και 7

```
Give me the integer index: 5
Give me the integer index: 7
The value of index is 7
The value of num is 27.566999
```

S2_LA6LO45

```
#include <stdio.h>
main () {
float index1;
float index2;
printf ("Give me the first number: ");
scanf("%f", &index1);
printf (" Give me the second number: ");
scanf("%f", &index2);
printf("The value of sum is %f\n",index1+index2);
getchar();
getchar();
```

```
return 0;
}
```

S2_LA6LO47

1-β 2-δ 3-ε 4-α 5-γ

S2_LA6LO48

```
#include <stdio.h>
main()
{
	int num;
	printf(“Δώσε ένα αριθμό: \t”);
	scanf(“%d”, &num);
	printf(“\n Χαρακτήρας: %c\t ASCII κωδικός: %d”, num, num);
}
```

S2_LA6LO49

```
#include <stdio.h>
void main() {
int a,b;
a = 2;
b = 1;
printf("%d\n",a);
printf("%d\n",b);
scanf("%d",&a);
scanf("%d",&b);
printf("%d\n",a);
printf("%d\n",b);
}
```

S2_LA6LO50

```
#include <stdio.h>
void main() {
float a,b;
a = 2;
b = 1;
printf("%.5f\n",a);
printf("%.5f\n",b);
scanf("%f",&a);
scanf("%f",&b);
printf("%.2f %.3f",a,b);
}
```

S3_LA7LO54
32
θα παρουσιαστεί λάθος κατά την εκτέλεση με το μήνυμα ‘non-value in assignment ‘

S3_LA7LO56
Το αποτέλεσμα θα είναι:

```
ακέραια διαίρεση:              5/3 είναι 1
ακέραια διαίρεση:              8/4 είναι 2
ακέραια διαίρεση:              7/5 είναι 1
διαίρεση κινητής υποδιαστολής: 7./4. είναι 1.75
μικτή διαίρεση:                7./4 είναι 1.75
```

S3_LA7LO57
Το αποτέλεσμα θα είναι:

```
Μετατροπή δευτερολέπτων σε λεπτά και δευτερόλεπτα
Δώστε τον αριθμό των δευτερολέπτων:
152
152 δευτερόλεπτα είναι 2 λεπτά και 32 δευτερόλεπτα.
```

S3_LA7LO58
1. 0 2. (-2) 3. 8 4. (-6)

S3_LA7LO60

```
printf("%d\n", 1 + 1 == 2); /* Τυπώνει 1 */
printf("%d\n", 1 > 2); /* Τυπώνει 0 */
printf("%d\n", 5 != 5); /* Τυπώνει 0 */
printf("%d\n", 1 <= 5); /* Τυπώνει 1 */
printf("%d\n", 1 <= 1); /* Τυπώνει 1 */
printf("%d\n", 1 <= 0); /* Τυπώνει 0 */
```

S3_LA7LO61
1. 0 2. 0 3. 0 4. 0

S3_LA7LO62
Βλέπε παρακάτω δραστηριότητα

S3_LA7LO63
Έστω για παράδειγμα 2 ακέραιοι που δίνει ο χρήστης είναι:12, 10

```
Dwse ton prwto akeraio:
12
Dwse ton deytero akeraio:
10
12!=0
10!=0
```

Kapoios arithmos einai arnhtikos -> 0
Kai oi dyo arithmoi einai arnhtikoi -> 0

S3_LA7LO65
Θα πάρει την τιμή 1, διότι ο τελεστής της ισότητας (λογικό ίσον = =) έχει μεγαλύτερη προτεραιότητα από τον τελεστή καταχώρησης (ένα απλό ίσον =). Άρα λοιπόν, καθώς οι μεταβλητές j και k είναι ίσες το αποτέλεσμα της σύγκρισης αυτών (j = = k) θα είναι αληθές, δηλαδή 1, τιμή που θα πάρει η μεταβλητή i

S3_LA7LO66
1. 0 2. 1 3. 0 4. 1

S3_LA7LO67
Αν το έτος λοιπόν είναι δεν είναι δίσεκτο τότε η λογική έκφραση επιστρέφει την τιμή 0 (ψευδής).

S3_LA7LO70
x =179

S3_LA7LO71
x=20

S3_LA7LO72
(10010011) & (00111101) == (00010001)
(10010011) | (00111101) == (10111111)
(10010011) □ (00111101) == (10101110)
(10001010) << 2 == (00101000)
(10001010) >> 2 == (00100010)
~(10011010) == (01100101)

S3_LA7LO73
~val = 11111101 = 253
printf("%d", ~val);

S3_LA8LO75
% a.out
a=2, b=3, x=0, y=0.666667

S3_LA8LO76
0110010
1000101
1111101
0111000
0110100
1100110

0010110

S3_LA8LO80

```
/* prog17.c - προγενέστεροι και μεταγενέστεροι */
#include <stdio.h>
main()
{
      int a =1, b = 1;
      int aplus, plusb;
      aplus = a++; /* μεταγενέστερος */
      plusb = ++b; /* προγενέστερος */
      printf("a aplus b plusb \n");
      printf("%1d %5d %5d %5d\n", a, aplus, b, plusb);
}
```

S3_LA8LO81

Εξήγηση	Εκχωρεί το
c = c + 7	10 στο c
d = d - 4	1 στο d
e = e * 5	20 στο e
f = f / 3	2 στο f
g = g % 9	3 στο g

S3_LA8LO82

x =14 y =6 z =8
x =6 y =6 z =8
x =14 y =6 z =8
x =13 y =5 z =7

S3_LA8LO83

a=4, syna=3, b=4, synb=4

και αυτό διότι στην εντολή syna=a++; γίνεται χρήση της τιμής της μεταβλητής a, πριν γίνει η αύξηση κατά μία μονάδα, ενώ στην synb=++b; μετά την αύξηση της μεταβλητής b κατά μία μονάδα. Επίσης η διαφορά μεταξύ των εντολών:

```
      alpha=beta*delta++; (I)
```

και

```
      alpha=beta*++delta; (II)
```

είναι ότι στην (I) πρώτα πολλαπλασιάζεται η beta με την delta και μετά αυξάνεται η delta κατά μία μονάδα, ενώ στην (II) πρώτα αυξάνεται η delta κατά μία μονάδα και μετά πολλαπλασιάζεται με την beta.

S3_LA8LO84

x	y
11	20
10	10
9	20
10	11
x	y
2	1
3	3
2	3

S3_LA8LO86

```
#include<stdio.h>
int main()
{
int x=5,y=10,max;
max = (x > y)? x : y; // στην μεταβλητή max θα εκχωρηθεί το y αφού είναι μεγαλύτερο του x
printf("The maximum number is %d.\n", max);
}
```

S3_LA9LO89

1. 4 2. 25 3. 10 4. 1 5. 2 6. 11 7. 17 8. 1 9. 1

S3_LA9LO91

-1

S3_LA9LO92

1. 3 2. 7 3. 4 4. 6 5. 9 6. 1 7. 8 8. 2 9. 5

ΠΑΡΑΤΗΡΗΣΗ: η συνάρτηση sqrt υπολογίζει την τετραγωνική ρίζα ενός αριθμού

S3_LA9LO94

Έκφραση στη γλώσσα C
m*m -n*n ή (m*m)-(n*n)
a*x*x + b*x + c
-b + 4 * a* c
(2*a*b)/(c+d)
-a*b + -c*d

S3_LA9LO96

Η πρώτη έκφραση δίνει αποτέλεσμα 3,56 και η δεύτερη 4,06

```
#include <stdio.h>
int main(void)
{
```

```
    printf("%f\n", 2.56 + 3/2);
    printf("%f\n", 2.56 + 3.0 /2);
    getchar();
    return 0;
}
```

S3_LA9LO97

Γ

S3_LA9LO98

Το αποτέλεσμα θα είναι:

```
    ch = A, i = 65, fl = 65.00
    ch = B, i = 197, fl = 329.00
```

S3_LA9LO99

2.5 3.0 2.5

S4_LA10LO101

1	2	3	4	5	6	7	8	9	10
Λ	Λ	Σ	Λ	Σ	Λ	Λ	Σ	Σ	Σ

S4_LA10LO102

β

S4_LA10LO103

```
#include <stdio.h>
main()
{
    int x;
    int y[34];
    char z ;
    char w[ ] ={ ‘a’, ‘b’, ‘c’};
}
```

S4_LA10LO104

100,400,800 (συνήθως)

S4_LA10LO106

1	2	3	4	5	6	7	8	9	10
Λ	Σ	Λ	Σ	Λ	Σ	Σ	Λ	Λ	Σ

S4_LA10LO107

```
int NUMBERS[6]={10, 12, 14, 16, 18, 20};
printf ("το πρώτο στοιχείο του πίνακα είναι το %d \n", NUMBERS[0]);
printf ("το δεύτερο στοιχείο του πίνακα είναι το %d \n", NUMBERS[5]);
```

S4_LA10LO108
B

S4_LA11LO110

1	2	3	4	5
int x, y; int *p1, *p2; x=-42 ; y=163 ;	p1=&x ; P2=&y ;	*p1=17 ;	p1=p2 ;	*p1=*p2 ;

S4_LA11LO111

```
int x=1; /* Αρχικοποίηση x */
int y=2; /* Αρχικοποίηση y */
int *p,*q; /* Δήλωση p, q */
p = &x; /* Ο p δείχνει στην x */
y = *p; /* y=x άρα y=1 */
*p = 0; /* x=0 */
*p+=1; /* x=x+1 άρα x=1 */
++*p; /* x=x+1 άρα x=2 */
(*p)++; /* x=x+1 άρα x=3 */
*p++; /* Το p θα δείχνει στο επόμενο στοιχείο */
q=p; /* Τα περιεχόμενα του p αντιγράφονται στον q */
```

S4_LA11LO112

```
ip=&x;        /*ο ip δείχνει τώρα την x*/
*ip=*ip+1;    /*η μεταβλητή που δείχνει ο ip (η x) αυξάνεται κατά 1*/
y=*ip+3;      /*η x παίρνει την τιμή 5*/
*ip=0;        /*η x παίρνει την τιμή 0*/
ip=&z[6];     /*ο ip δείχνει τώρα το z[6] */
iq=ip;        /*ο iq δείχνει τώρα το z[6] δηλαδή εκεί που δείχνει και ο ip */
```

S4_LA11LO113

```
    p1 = &a;
    *p1 = 4;
    p2 = p3;
    pd2 = &da;
    dc = *pd3;
```

S4_LA11LO115

```
int t[10], i = 5, *p;
p = t;
++p;
*p = 14;
*(t+i) = 33;
```

S4_LA11LO116

	a	b	c	ip1	ip2	ip3
Αρχή	-	-	-	-	-	-
a = 3	3	-	-	-	-	-
b = 4	3	4	-	-	-	-
ip1 = &a	3	4	-	&a	-	-
*ip1 = 8	8	4		&a		
ip3 = &c	8	4		&a		&c
ip2 = ip3	8	4		&a	&c	
b = *ip1	8	8		&a	&c	
*ip2 = *ip1	8	8	8	&a	&c	

S4_LA11LO117

3 2.000000
0 0.000000
10 2.000000

S4_LA11LO118

p1 is 1245064
(after ++) p1 is 1245068
(after --) p1 is 1245064

S4_LA12LO120

7. char msg[5]={'H','e','l','l','o','\0'}; Δεν δεσμεύουμε αρκετό χώρο
8. char msg[ ]={'H','e','l','l','o'}; Πίνακας χαρακτήρων που δεν τελειώνει σε \0.

Επομένως δεν είναι String

S4_LA12LO122

Διεύθυνση του πίνακα χαρακτήρων year	1	1	1821	Διεύθυνση της μεταβλητής year_prt

S4_LA12LO124

char *suit[4] = {"Hearts", "Diamonds", "Clubs", "Spades" };

S5_LA13LO126

1	2	3	4	5	6	7	8
Σ	**Λ**	**Λ**	**Σ**	**Λ**	**Σ**	**Λ**	**Σ**

S5_LA13LO127

ανάπτυξη, ανεξάρτητα, πιο εύκολο, ανεξάρτητα, συντήρησή, δομημένου, τεχνικές

S5_LA14LO129

1	2	3	4	5	6	7	8	9	10
Λ	Σ	Σ	Σ	Σ	Λ	Σ	Λ	Λ	Λ

S5_LA14LO130

(a < b)

printf("%d\n", a);

else printf("%d\n", b);

S5_LA14LO132

1	2	3	4	5	6	7	8	9	10
Λ	Σ	Σ	Λ	Λ	Λ	Σ	Λ	Σ	Σ

S5_LA14LO133

switch (grade)

case 'B':

printf("METRIA ");

case 'F':

S5_LA15LO135

1	2	3	4	5	6	7	8	9	10
Σ	Σ	Λ	Λ	Σ	Σ	Λ	Λ	Σ	Σ

S5_LA15LO136

Α. 3 Β. 0 Γ. Άπειρες Δ. 4

S5_LA15LO137

a=Z ASCII value=90

a=P ASCII value=80

a=F ASCII value=70

a=< ASCII value=60

a=2 ASCII value=50

S5_LA15LO139

1	2	3	4	5	6	7	8	9	10
Λ	Σ	Λ	Σ	Σ	Λ	Λ	Σ	Σ	Σ

S5_LA15LO140

Α. 1 Β. 3 Γ. 1 Δ. 1

S5_LA15LO141

6

3, 19, 3

S5_LA15LO143

1	2	3	4	5	6	7	8	9	10
Σ	Λ	Λ	Σ	Σ	Σ	Λ	Λ	Λ	Σ

S5_LA15LO144

α) 5

β) 45

S5_LA15LO145

α) 8,11,14,10,14,18,12,17

β) 7,5

γ) 7,6,5

S5_LA15LO146

1

11

2

i*2

S6_LA16LO148

printf, scanf, main, clrscr που υλοποιούνται από την βασική βιβλιοθήκη

εύρεση μεγίστου που δεν υλοποιείται από την βασική βιβλιοθήκη

S6_LA16LO149

1	2	3	4	5	6	7	8	9	10
Σ	Λ	Σ	Λ	Λ	Λ	Σ	Σ	Λ	Σ

S6_LA16LO151

1	2	3	4	5	6	7	8	9	10
Σ	Λ	Σ	Σ	Σ	Σ	Λ	Λ	Σ	Σ

S6_LA16LO152

Γ. -5, 10, 0

S6_LA16LO153
Δ. 10, 5
5, 10

S6_LA16LO154
Δ. 0, 0
10, 5

S6_LA17LO156
Α. Μεγαλύτερο των 5 και 4 είναι ο αριθμός 5
Β. megalytero
Γ. 2
Δ. p, q

S6_LA17LO157
Α. compute_area
Β. float a, float b
Γ. length, width
Δ. 2
Ε. 3,4,5,6
ΣΤ. 13

S6_LA17LO158
Α. int akeraio_mi(int a, int b)
Β. a<b
a
b

S6_LA17LO159
```
void display_area(floatR)
{
int result;
result = 3.14 * R * R;
printf(“The area is %f square meters\n”, result);
}
```

S6_LA18LO161
Α. x, p, i
Β. t
Γ. N
Δ. A(τυπική), p(πραγματική)

S6_LA18LO162
Α. a, b, c (μόνο η main τις βλέπει)
Β. c (μόνο η sum την βλέπει)

Γ. c (όλες οι συναρτήσεις την βλέπουν)
Δ. d
Ε. x, y
ΣΤ. a, b
Ζ. 5+6=12

S6_LA18LO164
Α. 1 Β. 5

S6_LA18LO166
1. a[]
2. i<N
3. &b[i]
4. &high
5. &low
6. high
7. low
8. *max
9. *min
10. *max
11. *min