Aristotle Certification Training & Assessment

Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Προγραμματισμός ΙΙ

(ενδιάμεσο επίπεδο)

Κατανοώντας τον Προγραμματισμό

Έκδοση 1

ACTA ΑΕ – Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Εγνατίας 1, 54630, Τηλ.: 2310 510 870, Fax: 2310 510 871, email: info@acta.edu.gr , www.acta.edu.gr

ΣΤΟΙΧΕΙΑ ΣΥΓΓΡΑΦΕΑ

Ονοματεπώνυμο Συγγραφέα: ΜΙΧΑΗΛ Ι. ΣΤΑΜΑΤΟΠΟΥΛΟΣ (Πληροφορικός ΠΕ19/ΠΕ86).

Είναι πτυχιούχος Πληροφορικός, του Τμήματος Πληροφορικής, της Σχολής Θετικών Επιστημών & Τεχνολογίας, Ε.Α.Π., Πατρών. Κατέχει παιδαγωγική και διδακτική επάρκεια στην εκπαίδευση της Πληροφορικής και έχει μεταπτυχιακή επιμόρφωση, σαν εκπαιδευτικός ειδικής αγωγής όπως επίσης και σαν εκπαιδευτής ενηλίκων, από το Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (Ε.Κ.Π.Α.). Είναι επιστημονικός συνεργάτης – ερευνητής, στο Εργαστήριο Ευφυών Συστημάτων (i-Lab), του Τμήματος Πολιτισμικής Τεχνολογίας και Επικοινωνίας (Τ.Π.Τ.Ε.), της Σχολής Κοινωνικών Επιστημών, του Πανεπιστημίου Αιγαίου. Εργάζεται σαν μηχανικός λογισμικού και σαν εκπαιδευτικός Πληροφορικής για περισσότερα από 25 έτη και υπήρξε εκπρόσωπος στην Ελλάδα, κορυφαίων διεθνώς εταιρειών τεχνολογίας (FARO Technologies, CADZone, Trancite, A-T Solution).

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 2^{ο}

«ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΗΝ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 3ο

«ΜΕΤΑΒΛΗΤΕΣ, ΣΤΑΘΕΡΕΣ ΚΑΙ ΤΕΛΕΣΤΕΣ»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 4ο

«Η ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΗΝ ΜΗΧΑΝΗ»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

ΚΕΦΑΛΑΙΟ 5ο

«ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ»

ΚΕΦΑΛΑΙΟ 6^{ο}

«ΕΝΑΣ ΑΛΓΟΡΙΘΜΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ»

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Γλωσσάριο σημαντικών όρων

Βιβλιογραφία

Οδηγίες για περαιτέρω μελέτη

ΕΙΣΑΓΩΓΗ

Είναι πλέον γεγονός ότι οι γλώσσες προγραμματισμού και οι κάθε είδους εφαρμογές που δημιουργούνται από αυτές, διαδραματίζουν έναν σημαντικό ρόλο στην καθημερινότητα των ανθρώπων του σύγχρονου κόσμου. Οι εφαρμογές απασχολούν έντονα τους ανθρώπους και καταναλώνουν σημαντικό μέρος από τον χρόνο τους. Αυτή η σύγχρονη *κοινωνία της πληροφορίας*, (Information Society) είναι μια νέα κατάσταση στην οποία οι υπολογιστές, διασυνδεδεμένοι μέσω του διαδικτύου (internet), προσφέρουν υπηρεσίες για κάθε πλευρά του πολιτισμού, όπως, οι μεταφορές, οι επικοινωνίες, η εκπαίδευση, το εμπόριο, η ασφάλεια αλλά και η ίδια η λειτουργία των κυβερνήσεων. Τα σπουδαία αυτά τεχνολογικά επιτεύγματα γίνονται δυνατά, χάρη στην ύπαρξη και την λειτουργία των εφαρμογών του λογισμικού. Παρόλη όμως αυτή την καθολική χρήση των εφαρμογών λογισμικού, δεν είναι αυτονόητο ότι όλοι οι χρήστες αντιλαμβάνονται πραγματικά το πώς λειτουργεί ή το πώς κατασκευάζεται μια εφαρμογή λογισμικού.

Η γνώση των βασικών στοιχείων προγραμματισμού των υπολογιστών είναι πια απαραίτητη για κάθε νέο άνθρωπο, ο οποίος δεν θέλει να μένει πίσω από τις τεχνολογικές εξελίξεις. Στα πλαίσια αυτά, το σύγγραμμα, *Προγραμματισμός ΙΙ (ενδιάμεσο επίπεδο), Κατανοώντας τον Προγραμματισμό,* επιχειρεί να δώσει στους εκπαιδευόμενους όλες εκείνες τις απαραίτητες γνώσεις για να αποκτήσουν ικανότητες να κατασκευάζουν προγράμματα χρησιμοποιώντας μια σύγχρονη γλώσσα προγραμματισμού.

Το σύγγραμμα, *Προγραμματισμός ΙΙ (ενδιάμεσο επίπεδο), Κατανοώντας τον Προγραμματισμό*, δεν είναι ένα εισαγωγικό σύγγραμμα προγραμματισμού. Προϋποθέτει την εξοικείωση με κάποιες βασικές γνώσεις προγραμματισμού όπως οι έννοιες, αλγόριθμος, μεταβλητή, δεδομένα εισόδου, δομές επανάληψης, δομές επιλογής, δομημένος προγραμματισμός, κ.λπ. Παρόλα αυτά ένας νέος και άπειρος προγραμματιστής μπορεί να το διαβάσει εάν, έχει ολοκληρώσει την μελέτη του συγγράμματος, *Προγραμματισμός Ι (βασικό επίπεδο), Ανακαλύπτοντας τον Προγραμματισμό*, των συγγραμμάτων της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ΑΣΤΑ.

1. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Σκοπός και επιμέρους στόχοι

Σκοπός του πρώτου κεφαλαίου είναι να εισάγει τους εκπαιδευόμενους στον κόσμο του προγραμματισμού των ηλεκτρονικών υπολογιστών, με την γλώσσα προγραμματισμού C. Το κεφάλαιο χωρίζεται σε δυο μέρη. Το πρώτο μέρος περιλαμβάνει μια σύντομη αλλά σημαντική αναφορά στην ιστορία της γλώσσας προγραμματισμού C και το δεύτερο μέρος περιέχει όλο το απαραίτητο εκπαιδευτικό και καθοδηγητικό υλικό, για την εγκατάσταση, παραμετροποίηση και χρήση ενός μεταγλωττιστή γλώσσας C++.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του πρώτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να περιγράφουν σημαντικά ιστορικά στοιχεία για την γλώσσα προγραμματισμού C,
- να αντιλαμβάνονται την σημασία της γλώσσας προγραμματισμού C και των σύγχρονων «απογόνων» της,
- να επιλέγουν κατάλληλο περιβάλλον ανάπτυξης για τα προγράμματα τους,
- να εγκαθιστούν στον υπολογιστή τους ένα ολοκληρωμένο περιβάλλον ανάπτυξης μιας γλώσσας και να ρυθμίζουν την λειτουργία του,
- να αντιλαμβάνονται ποια είναι τα βασικά στοιχεία ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης (I.D.E.) σε μια γλώσσα προγραμματισμού,
- να χρησιμοποιούν τα κατάλληλα εργαλεία για να δημιουργούν προγράμματα σε πηγαία μορφή και να τα μετατρέπουν σε εκτελέσιμη μορφή,
- να αντιλαμβάνονται τα βασικά στοιχεία λειτουργίας του I.D.E. περιβάλλοντος, Dev-C++, της BloodShed.

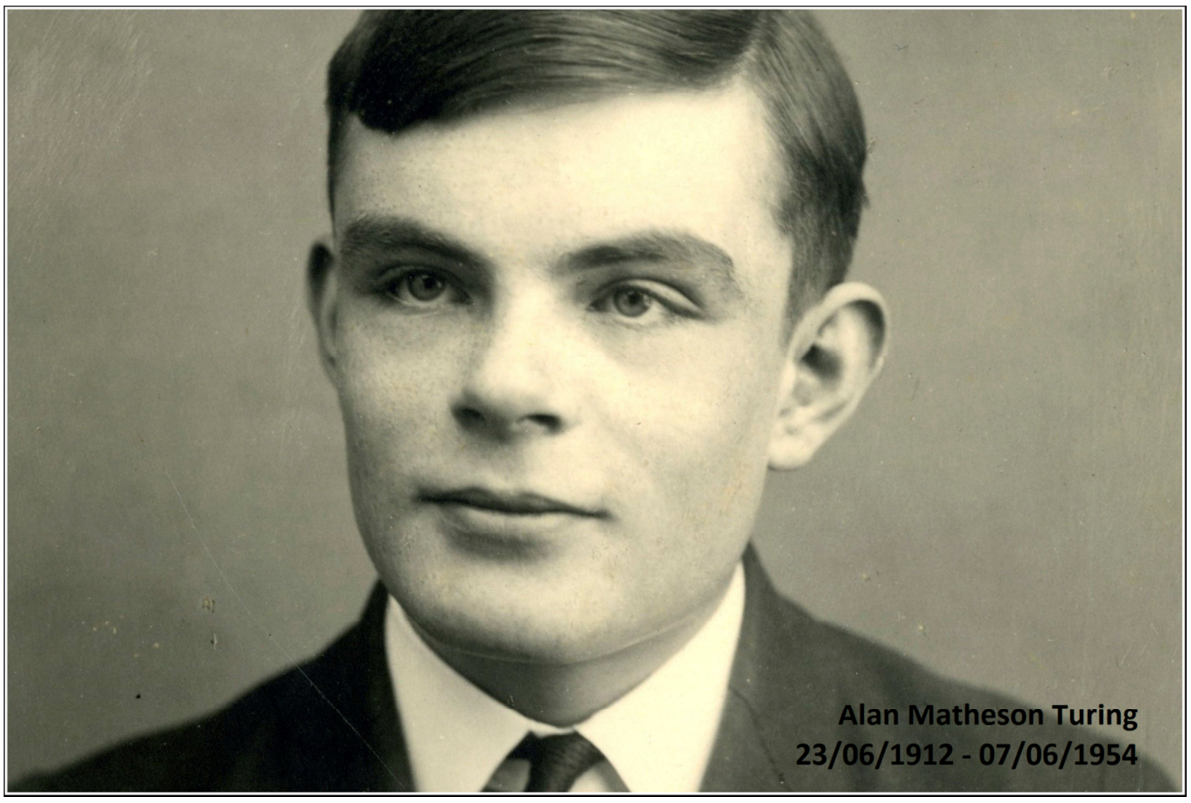
Έννοιες – Λέξεις Κλειδιά

Ομοιάζουσες γλώσσες, Μεταφερσιμότητα, Πρότυπο C++20, Κυβερνοχώρος, Παγκόσμιος ψηφιακός μετασχηματισμός, Τεχνολογίες αιχμής, Εγκατάσταση γλώσσας, Ολοκληρωμένο περιβάλλον ανάπτυξης, Μεταγλώττιση κώδικα, Εκτέλεση κώδικα.

Εισαγωγικές Παρατηρήσεις

Για την επιστήμη της *Πληροφορικής*, η ύψιστη σε παγκόσμιο επίπεδο επιστημονική διάκριση, είναι το βραβείο, ***Alan Turing***, ή βραβείο *A.C.M. Alan Matheson Turing*, ή βραβείο "*Nobel Prize of Computing,*".

Το βραβείο *Alan Turing*, θεσμοθετήθηκε το έτος 1966 και απονέμεται κάθε χρόνο σε αυτόν ή σε αυτούς, που με την εργασία τους σε κάποιο τομέα της κοινωνίας των πληροφοριών, καταφέρνουν να εξελίσσουν την επιστήμη της *Πληροφορικής*. Από το έτος 2014 το βραβείο *Alan Turing*, συνοδεύεται με χρηματικό έπαθλο, 1.000.000 δολαρίων.



Το βραβείο, πήρε το όνομα του, προς τιμήν του Alan Turing (23 Ιουνίου 1912 - 7 Ιουνίου 1954), του ανθρώπου, ο οποίος με την εργασία του, τις μηχανές Turing, οδήγησε στο «σπάσιμο» του Enigma, της συσκευής κρυπτογράφησης που είχαν κατασκευάσει και χρησιμοποιούσαν κατά την διάρκεια του καταστροφικού, Β' Παγκοσμίου Πολέμου (WW2), τα ναζιστικά γερμανικά στρατεύματα. Θεωρείται δεδομένο ότι μια από τις βασικές αιτίες (και ίσως η βασικότερη) που

οι γερμανοί έχασαν τον πόλεμο, ήταν οι υπολογιστές της εποχής, η ομάδα του Άγγλου μαθηματικού, Alan Turing και το σπάσιμο του Enigma (από την ελληνική λέξη Αίνιγμα).

Το έτος 1983, το βραβείο *Alan Turing*, απονεμήθηκε στους Ken Thompson και Dennis M. Ritchie για την συνεισφορά τους στην θεωρεία των λειτουργικών συστημάτων των υπολογιστών και του έργου τους, την κατασκευή του λειτουργικού συστήματος UNIX.

Οι δυο αυτοί άνθρωποι που είχαν την τύχη και την εξαιρετική ικανότητα να λάβουν μια τέτοια διάκριση, είναι οι δημιουργοί της γλώσσας C, της γλώσσας προγραμματισμού με την οποία εκπαιδεύονται και έχουν εκπαιδευτεί, όλοι σχεδόν οι μηχανικοί λογισμικού και οι προγραμματιστές, όλων των εποχών. Οι μηχανισμοί λειτουργίας της γλώσσας C, έχουν αφομοιωθεί από πολλές άλλες γλώσσες και δεν είναι υπερβολικό να ειπωθεί ότι οι περισσότερες σύγχρονες γλώσσες προγραμματισμού είναι C-like ή C-family γλώσσες (ομοιάζουσες με την C). Η εκμάθηση της γλώσσας C, από κάθε νέο προγραμματιστή θεωρείται απαραίτητη, καθώς είναι ικανή να δημιουργήσει το κατάλληλο τεχνικό υπόβαθρο για την εύκολη εκμάθηση και κάθε άλλης γλώσσας προγραμματισμού.

Τα συγγράμματα της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ACTA, χρησιμοποιούν και περιγράφουν την γλώσσα προγραμματισμού C.

1.1. Μια Ιστορία για την Γλώσσα Προγραμματισμού C

Η γλώσσα προγραμματισμού C είναι μια γλώσσα γενικής χρήσης. Έχει συνδεθεί άμεσα με το λειτουργικό σύστημα UNIX, ένα από τα ικανότερα και σπουδαιότερα λειτουργικά συστήματα στην βιομηχανία των ηλεκτρονικών υπολογιστών. Η γλώσσα C, έργο των Ken Thompson και Dennis Ritchie, δημιουργήθηκε το 1973, για να χρησιμοποιηθεί στην κατασκευή, μιας έκδοσης λειτουργικού συστήματος UNIX, το οποίο θα υποστήριζε τη λειτουργία ενός υπολογιστή DEC PDP-11. Πολλά από τα τεχνικά χαρακτηριστικά της νέας τότε γλώσσας C, προέρχονταν από την παλαιότερη γλώσσα B, η οποία είχε δημιουργηθεί το 1970, από τον Ken Thompson και είχε χρησιμοποιηθεί για την κατασκευή της πρώτης έκδοσης του λειτουργικού συστήματος UNIX, το οποίο θα υποστήριζε για πρώτη φορά έναν υπολογιστή (ένα DEC PDP-7). Κάποια από τα χαρακτηριστικά της γλώσσας B, προέρχονταν και αυτά, από τη γλώσσα BCPL, που είχε δημιουργηθεί παλαιότερα από τον Martin Richards.

Η C δεν θεωρείται μια γλώσσα ιδιαίτερα «υψηλού επιπέδου», στην C, ταιριάζει περισσότερο ο χαρακτηρισμός, γλώσσα «χαμηλού επιπέδου». Μια αιτιολόγηση για τον χαρακτηρισμό αυτό, είναι, το «συμπυκνωμένο» λεξιλόγιο της γλώσσας και η ύπαρξη μεγάλου αριθμού «διακοπτών», για την ενεργοποίηση των δυνατοτήτων της. Αυτό μπορεί ίσως να αποδοθεί στην τεχνική ιδιοσυγκρασία, που είχαν αναπτύξει οι δημιουργοί της, εκείνη την εποχή, καθώς μεγάλωσαν μέσα σε περιβάλλοντα όπου επικρατούσαν γλώσσες όπως, οι Assembly, η FORTRAN και η ALGOL. Παρόλο που η γλώσσα C είχε αρχικά συνδεθεί με το λειτουργικό σύστημα UNIX, δεν θα πρέπει σήμερα να συνδέεται απαραίτητα με κάποιο συγκεκριμένο λειτουργικό σύστημα. Επίσης, αν και η C μπορεί να ταιριάξει με τις ιδιαιτερότητες στο χαμηλότερο επίπεδο συγκεκριμένων υπολογιστών, είναι ανεξάρτητη από οποιαδήποτε αρχιτεκτονική υπολογιστών και διατηρεί την μεταφερσιμότητα της. Με λίγη προσπάθεια και προσοχή από τον προγραμματιστή, είναι δυνατό να δημιουργηθούν προγράμματα που να μπορούν να εκτελεστούν χωρίς καμία αλλαγή σε οποιονδήποτε υπολογιστή ανεξαρτήτου αρχιτεκτονικής. Γενικότερα, παρόλο που η C αρχικά σχεδιάστηκε σαν μια γλώσσα προγραμματισμού συστημάτων (μεταγλωττιστών, λειτουργικών συστημάτων, επεξεργαστών κειμένου, κ.λπ.) σήμερα έχει καθιερωθεί παντού και χρησιμοποιείται σε πολλά και διαφορετικά πεδία εφαρμογών, χαμηλού αλλά και υψηλού επιπέδου.

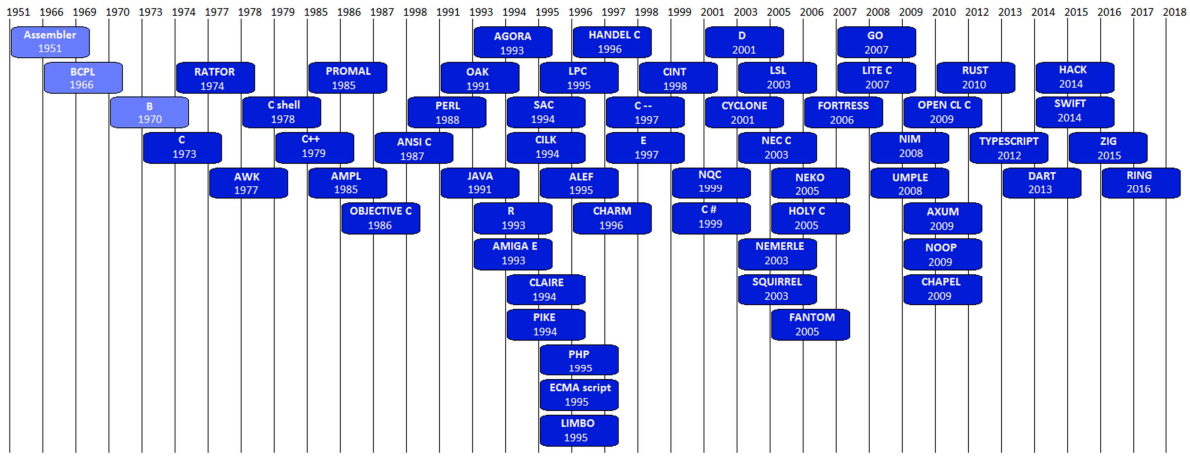
Από το 1973 και στη διάρκεια των 50 χρόνων ζωής της, η γλώσσα C, πέρασε μέσα από διάφορες εκδόσεις πολλών διαφορετικών κατασκευαστών και εξελίχθηκε σε πολύ μεγάλο βαθμό με επίσημες καθιερώσεις και διεθνή πρότυπα (ANSI, ISO). Το 1979 εμφανίστηκε η έκδοση C++, μια προέκταση, προς την νέα τεχνοτροπία του αντικειμενοστραφούς προγραμματισμού (Object Oriented Programming, O.O.P.), ενώ αυτό συνεχίστηκε και το 1999 με την εμφάνιση της έκδοσης C#. Οι δημιουργοί των C++ και C#, προσπάθησαν και προσπαθούν με τις εκδόσεις τους, να εισάγουν κατάλληλες τεχνικές ώστε να καταστήσουν την συγγραφή ενός προγράμματος από τους νέους προγραμματιστές ευκολότερη διαδικασία από ότι η κλασσική C. Οι γλώσσες C, C++, C# αλλά και κάθε άλλη έκδοση της C, δεν παράγονται από έναν και μόνο κατασκευαστή. Οι γλώσσες παράγονται από πολλούς και διαφορετικούς κατασκευαστές μεταγλωττιστών, με κάποιες από αυτές να πωλούνται εμπορικά και κάποιες άλλες να διατίθενται δωρεάν. Η γλώσσα προγραμματισμού C, όπως και κάθε άλλη γλώσσα, έχει τα ελαττώματά της με κάποια στοιχεία να παρουσιάζουν μια σχετική δυσκολία στην αρχική τους αφομοίωση, όμως είναι σίγουρο ότι η C, είναι μια ικανή πλατφόρμα ανάπτυξης, με πλήθος δομικών στοιχείων απομακρυσμένων από το χαμηλό επίπεδο των μηχανών και μπορεί να αποδειχθεί εξαιρετικά αποτελεσματική για ένα ευρύ κοινό προγραμματιστών και μια μεγάλη ποικιλία εφαρμογών προγραμματισμού υψηλού επιπέδου.

Στο σύγγραμμα αυτό και για την συνέχεια της εκπαιδευτικής διαδικασίας, επιλέχθηκε να χρησιμοποιηθεί η έκδοση C++, κατά το νεώτερο πρότυπο, ISO 2020, (ή πρότυπο C++20), το οποίο εκδόθηκε το έτος 2020 από τον παγκόσμιο οργανισμό ISO, για την γλώσσα C++.

Οι έννοιες του αντικειμενοστραφούς προγραμματισμού (Object Oriented Programming, O.O.P.), μέσω των εκδόσεων C++ και C#, περιγράφονται στο τελευταίο σύγγραμμα της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ACTA.

1.2. Οι Ομοιάζουσες Γλώσσες Προγραμματισμού της C

Οι γλώσσες προγραμματισμού των ηλεκτρονικών υπολογιστών, εκτείνονται γύρω από ένα ευρύ φάσμα επιλογών και δυνατοτήτων. Πολλές από τις γλώσσες προγραμματισμού, χρησιμοποιούνται για την ανάπτυξη εμπορικών εφαρμογών, άλλες για εφαρμογές internet, για επιστημονικές εφαρμογές, για την εκπαίδευση, για γενικούς σκοπούς και άλλες για ειδικές χρήσεις. Η γλώσσα C, αποτελεί μια ειδική περίπτωση στην ιστορία των γλωσσών προγραμματισμού, καθώς, πέρα από το ότι είναι ιδιαίτερα δημοφιλής, έχει καταφέρει να δημιουργήσει και να δημιουργεί ακόμα και σήμερα σημαντικούς «απογόνους», C-like γλώσσες ή C-family γλώσσες.



Αυτές οι ομοιάζουσες γλώσσες, «απόγονοι» της C, είναι γλώσσες οι οποίες έχουν επηρεαστεί από τον τρόπο κατασκευής της αρχικής C και έχουν υιοθετήσει πολλά στοιχεία από τους μηχανισμούς και την λειτουργικότητα της. Το γενεαλογικό δέντρο της γλώσσας C, των, Ken Thompson και Dennis Ritchie, αποτυπώνεται σχηματικά στην ανωτέρω εικόνα.

1.3. Ο Παγκόσμιος Ψηφιακός Μετασχηματισμός και η C

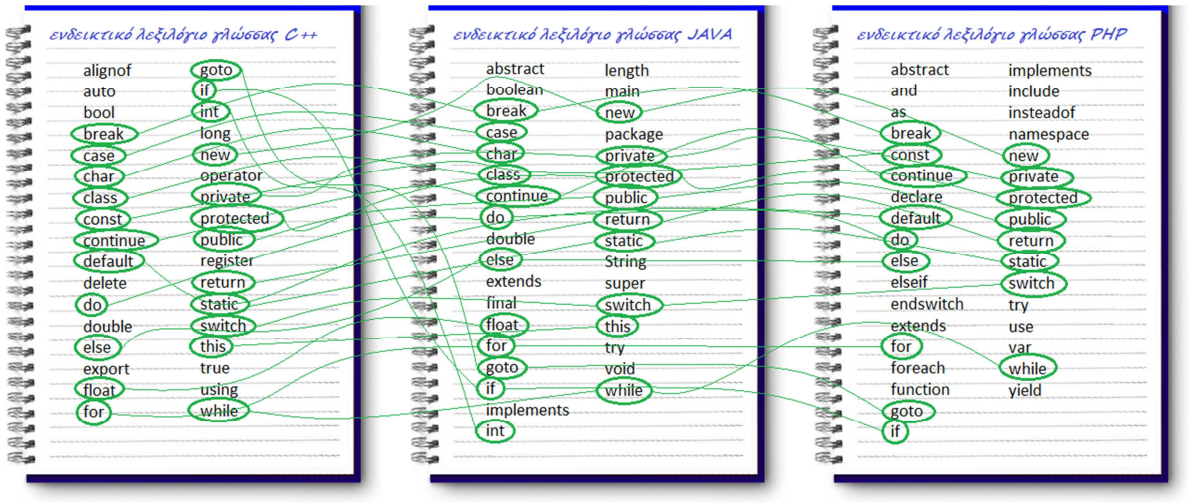
Από τις αρχές της δεκαετίας του 2020 το ψηφιακό περιβάλλον του διαδικτύου, ο κυβερνοχώρος ή το «σύννεφο» (cloud) όπως ονομάζεται, άρχισε να επιδρά με καταλυτικό τρόπο στις ζωές των ανθρώπων σε όλες τις χώρες του κόσμου. Τα γεγονότα που άλλαξαν δραματικά την ιστορία του κόσμου, ο ιός SARS-CoV-2 και η νόσος COVID-19, οδήγησαν τις σύγχρονες κοινωνίες στην επιτακτική ανάγκη δημιουργίας νέων μηχανισμών άμυνας, για το ανθρώπινο είδος.

Η επιστήμη της *Πληροφορικής*, ανέλαβε ένα μεγάλο βάρος της ευθύνης προς αυτό το σκοπό. Ο σύγχρονος πολιτισμός έπρεπε, γρήγορα, μαζικά και με ασφάλεια να συνεχίσει να λειτουργεί, online. Η εκπαίδευση, η εργασία, οι επικοινωνίες, οι μεταφορές, η τροφοδοσία, η ανοσοποιητική θωράκιση του πληθυσμού, οι παροχές των δημοσίων υπηρεσιών, έπρεπε να συνεχίσουν να λειτουργούν «ψηφιακά».

Αυτός ο μαζικός παγκόσμιος ψηφιακός μετασχηματισμός, υλοποιήθηκε και στην Ελλάδα η οποία δικαίως κατατάσσεται στις χώρες που πρωτοποριακά κατάφεραν μέσα σε σύντομο χρονικό διάστημα και με μεγάλα βήματα να φύγουν προς το «σύννεφο». Η σημαντική αυτή

μεταστροφή των πραγμάτων, υλοποιήθηκε με τεχνολογίες που προσφέρουν οι σύγχρονοι «απόγονοι» της γλώσσας C.

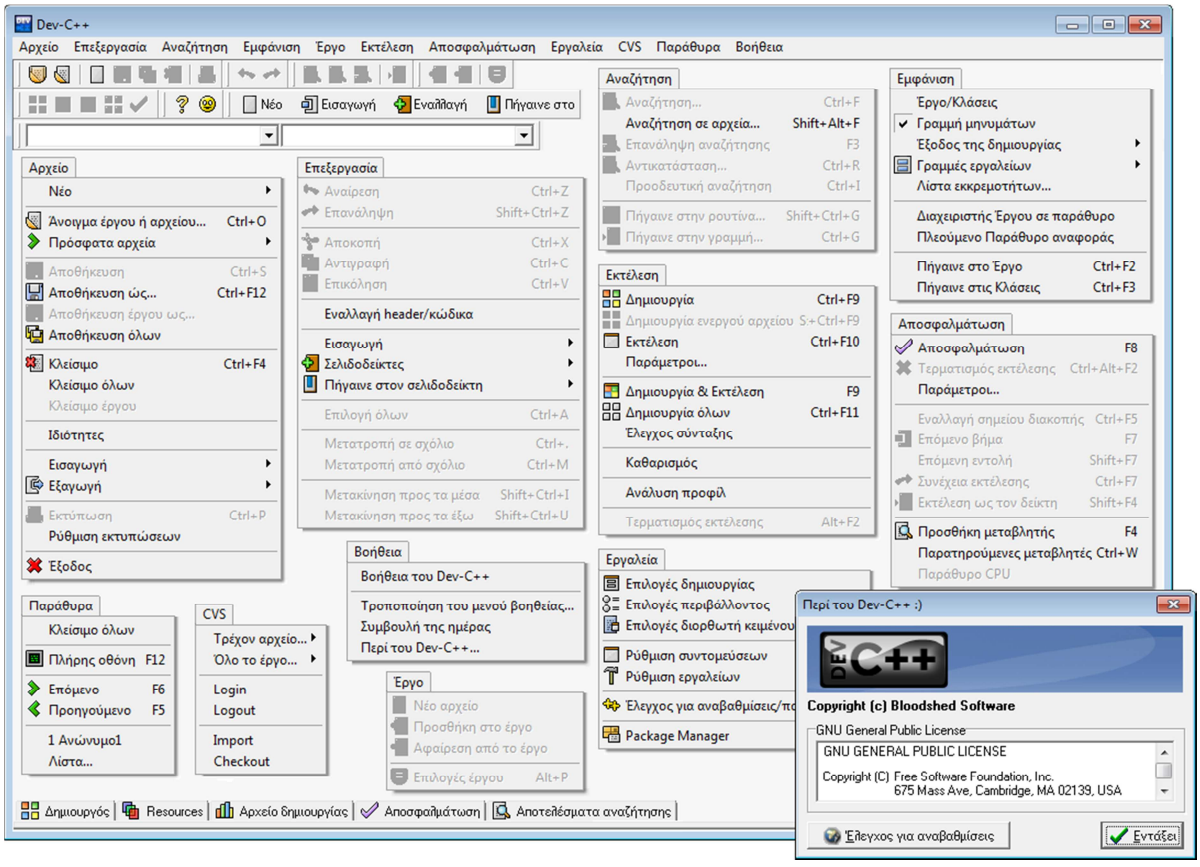
Γλώσσες «απόγονοι» όπως η JAVA και η PHP, βρίσκονται πίσω από όλες σχεδόν τις πρόσφατα δημιουργημένες διαδικτυακές πλατφόρμες, που ευρέως χρησιμοποιούμε σήμερα όλοι, κρατώντας στο χέρι μας μια μικρή φορητή υπολογιστική μηχανή.



Αυτές οι τεχνολογίες των γλωσσών προγραμματισμού, «απόγονοι» της γλώσσας C, θεωρούνται σήμερα οι τεχνολογίες αιχμής (state_of_the_art) στον προγραμματισμό και γενικότερα στην τεχνολογία λογισμικού (Software Engineering).

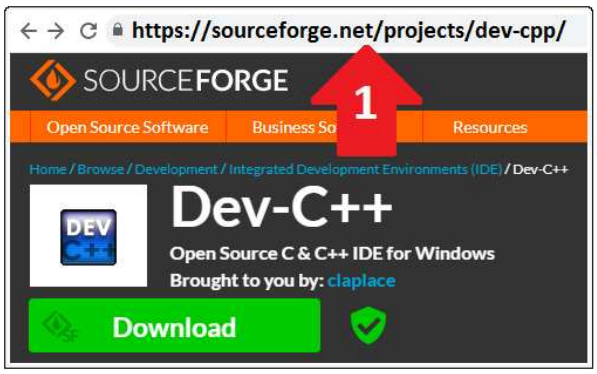
1.4. Εγκατάσταση της Γλώσσας Προγραμματισμού C++

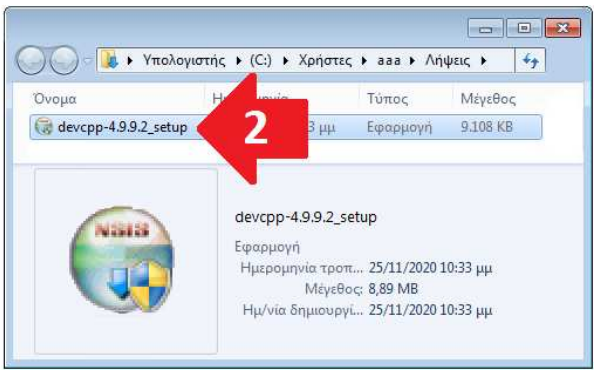
Τα κεφάλαια που ακολουθούν, περιλαμβάνουν αναλυτικές παρουσιάσεις πολλών διαφορετικών δυνατοτήτων της γλώσσας προγραμματισμού C++. Οι παρουσιάσεις αυτές, υλοποιούνται μέσα από το ολοκληρωμένο περιβάλλον Dev-C++, της Bloodshed. Για την μελέτη, «στην πράξη», των ενοτήτων αυτών, θα πρέπει οι εκπαιδευόμενοι να προχωρήσουν στην εγκατάσταση του συγκεκριμένου μεταγλωττιστή στον προσωπικό τους υπολογιστή.



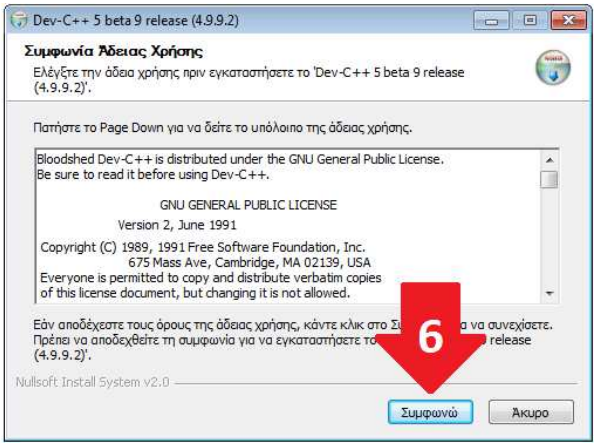
Ο μεταγλωττιστής *Dev-C++*, όπως ονομάζεται, είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (I.D.E. - Integrated Development Environment) το οποίο παρέχει όλα εκείνα τα απαραίτητα εργαλεία για την κατασκευή, δοκιμή και εκτέλεση προγραμμάτων C++. Ειδικότερα, το I.D.E. της BloodShed περιλαμβάνει, επεξεργαστή κειμένου, πολλές χρήσιμες βιβλιοθήκες (libraries), μεταφραστή (compiler), προ-μεταφραστή (pre-compiler) και έναν debugger για τον έλεγχο και την αποσφαλμάτωση προγραμμάτων. Ο compiler διατίθεται δωρεάν και μπορεί να χρησιμοποιηθεί ελεύθερα με την άδεια, G.N.U., (General Public License), της Bloodshed (βλ. βέλος, 1).

Το αρχείο για την μεταφόρτωση και εγκατάσταση του, μπορεί να ανευρεθεί στην διεύθυνση, http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe

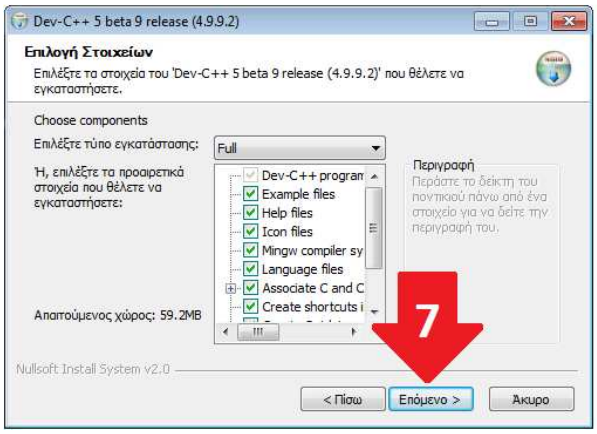


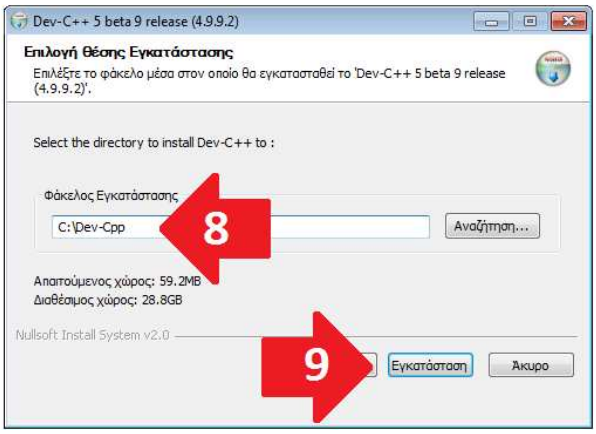


Η διαδικασία απαιτεί ελεύθερο χώρο 60 MB στον σκληρό δίσκο του υπολογιστή εγκατάστασης. Με το τέλος της διαδικασίας μεταφόρτωσης, θα πρέπει να αναζητηθεί το αρχείο, **devcpp-4.9.9.2_setup.exe**, μέσα από τον φάκελο λήψεων του υπολογιστή εγκατάστασης (βλ. βέλος, 2) και να εκτελεστεί, με δικαιώματα διαχειριστή.

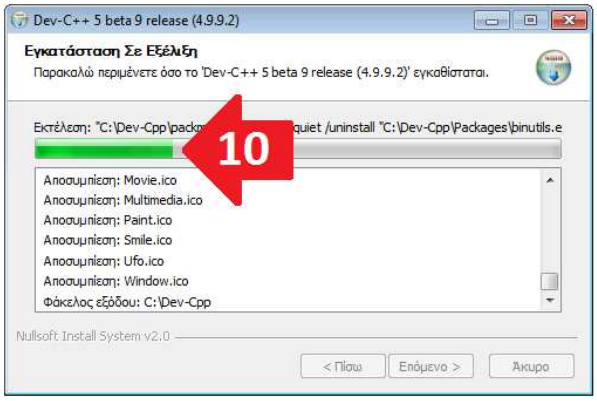


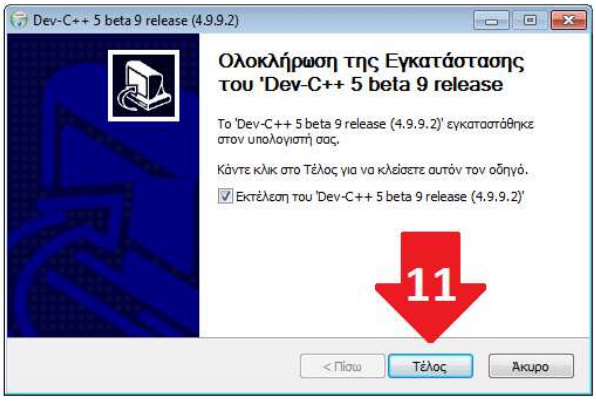
Η εγκατάσταση του μεταγλωττιστή της BloodShed ολοκληρώνεται σε βήματα (βλ. βέλη, 3, 4, 5, 6, 7, 8, 9, 10, 11), με το τελικό στάδιο να περιλαμβάνει μια δήλωση ρυθμίσεων σχετικών με την εν συνέχεια λειτουργία του.





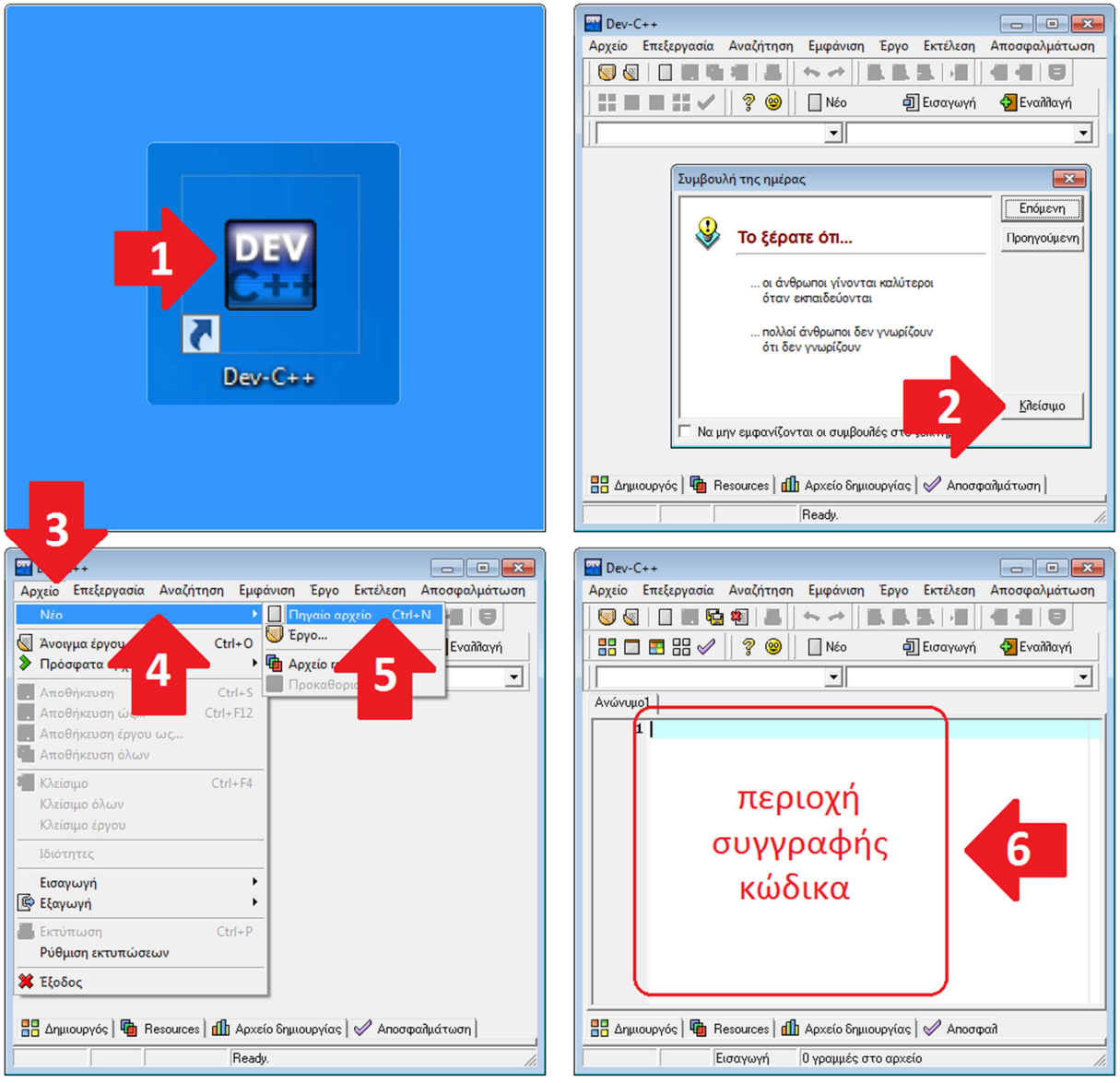
Οι αρχικές αυτές ρυθμίσεις αφορούν, την λειτουργία του κεντρικού μενού στα ελληνικά και την δυνατότητα παραγωγής αρχείων με επέκταση (extension), **ccc** (**Code Completion Cache**). Οι απαντήσεις και στις δυο αυτές ερωτήσεις είναι θετικές.

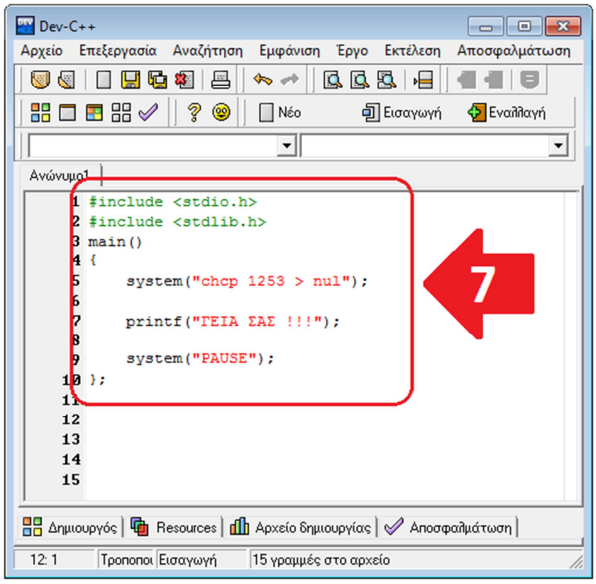


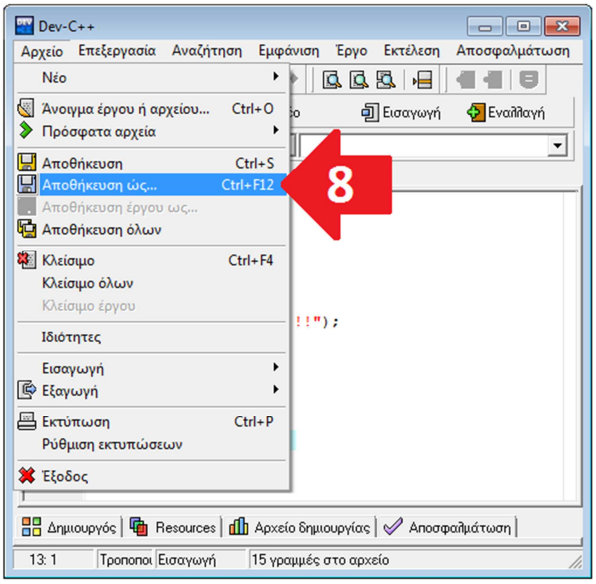


1.5. Ενεργοποίηση της Γλώσσας Προγραμματισμού C++

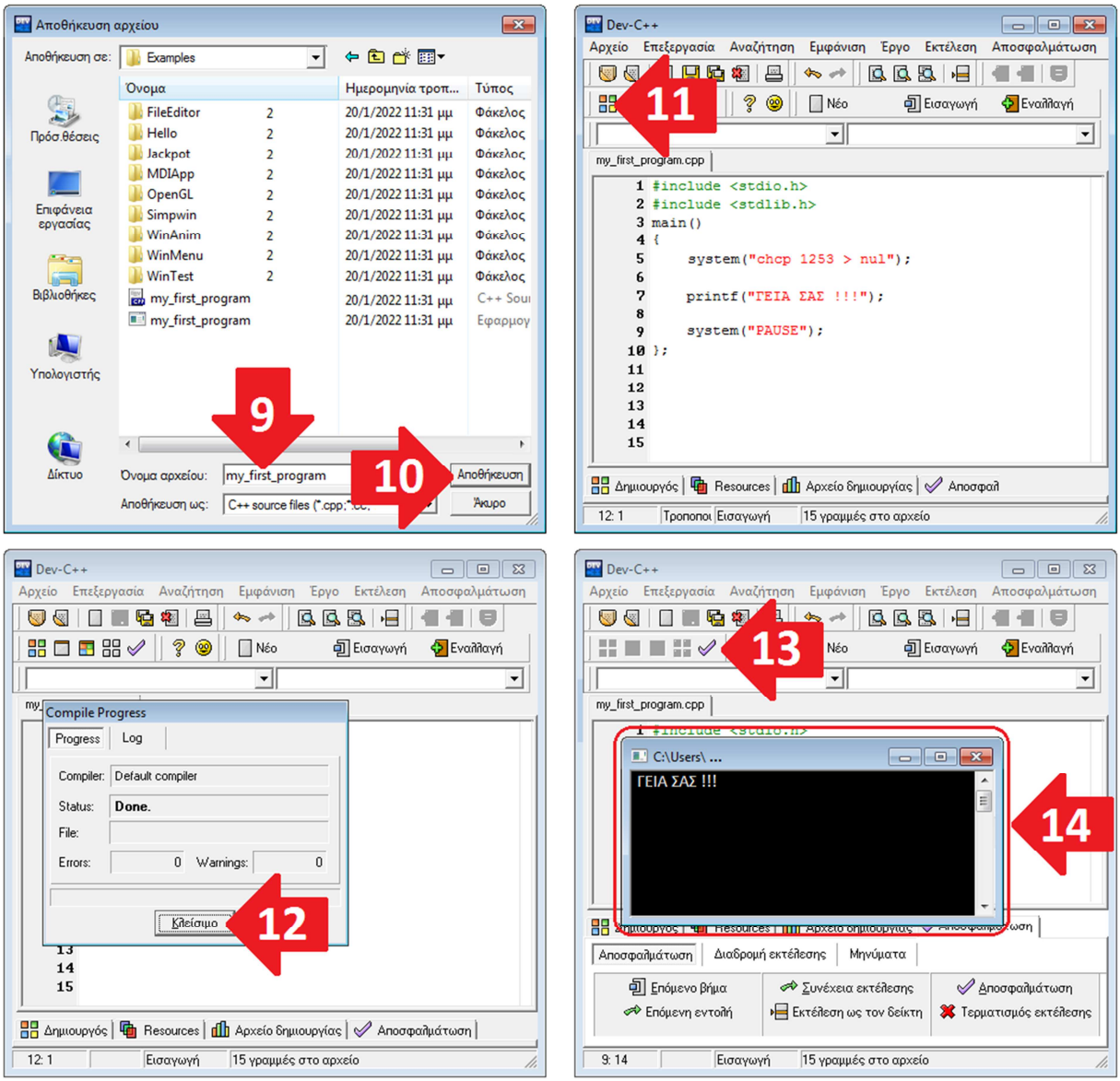
Με την ολοκλήρωση της εγκατάστασης του μεταγλωττιστή Dev-C++, έχει προστεθεί στην επιφάνεια εργασίας του υπολογιστή ένα σχετικό εικονίδιο ενεργοποίησης. Από το εικονίδιο αυτό (βλ. βέλος, _1_), ο χρήστης εισάγεται στο περιβάλλον I.D.E. της BloodShed και στο πλούσιο σε επιλογές κεντρικό του μενού (βλ. βέλος, _2_). Η συγγραφή του πηγαίου κώδικα (source code) κάθε νέου προγράμματος, μπορεί να γίνει με τον ενσωματωμένο επεξεργαστή κειμένου (text editor). Για να γίνει αυτό θα πρέπει, από το κεντρικό μενού του Dev-C++, και την ομάδα επιλογών, «*Αρχείο*», να δοθεί η επιλογή «*Νέο*» και ακολούθως η υπο-επιλογή «*Πηγαίο αρχείο*» (βλ. βέλη, _3_, _4_ και _5_).



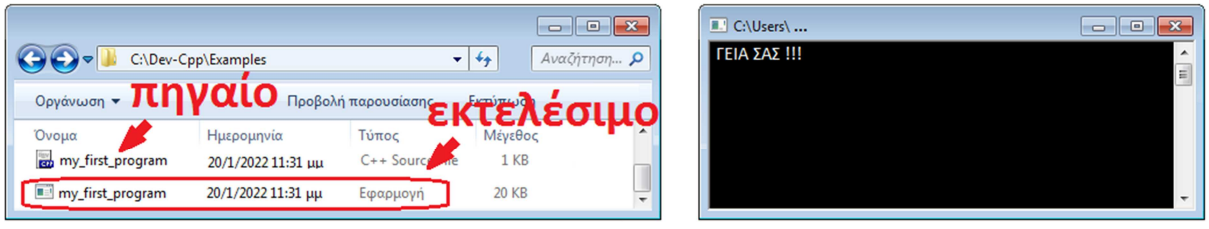




Αμέσως θα ανοίξει μια κενή περιοχή («*περιοχή συγγραφής κώδικα*»), στην οποία μπορεί να τοποθετηθεί ο πηγαίος κώδικας (βλ. βέλη, **6** και **7**). Κάθε νέο πρόγραμμα το οποίο συγγράφεται θα πρέπει να φέρει και ένα κατάλληλο όνομα. Για να δηλωθεί μια ονομασία θα πρέπει, από το κεντρικό μενού του Dev-C++, και την ομάδα επιλογών, «*Αρχείο*», να δοθεί η επιλογή «*Αποθήκευση ως…*» (βλ. βέλος, **8**) και στο παράθυρο αποθήκευσης αρχείων που θα ανοίξει, να οριστεί ένα περιγραφικό όνομα (π.χ. **my_first_program.cpp**). Το απαραίτητο extension **cpp** (**C** **p**lus **p**lus), θα προστεθεί αυτόματα (βλ. βέλη, **9** και **10**). Όταν ολοκληρωθεί η διαδικασία της συγγραφής του προγράμματος θα πρέπει στην συνέχεια να γίνει η μεταγλώττιση του κώδικα (compilation). Το compilation, είναι η διαδικασία του συντακτικού ελέγχου του πηγαίου κώδικα και της μετατροπής του σε εκτελέσιμο πρόγραμμα (executable code). Για να γίνει αυτό θα πρέπει να ενεργοποιηθεί το σχετικό πλήκτρο (βλ. βέλος, **11**), το οποίο βρίσκεται στο κεντρικό μενού των επιλογών. Εάν το πηγαίο πρόγραμμα δεν περιέχει συντακτικά λάθη (βλ. βέλος, **12**), ο compiler θα δημιουργήσει ένα εκτελέσιμο αρχείο απαλλαγμένο από λάθη το οποίο και είναι έτοιμο για να εκτελεστεί (να «τρέξει»). Το «τρέξιμο» αυτό, θα γίνει με το ειδικό πλήκτρο του Dev-C++, το οποίο βρίσκεται επίσης στο κεντρικό μενού επιλογών (βλ. βέλος, **13**). Ο εκτελέσιμος κώδικας που έχει δημιουργηθεί με το όνομα (π.χ. **my_first_program.exe**) θα ενεργοποιηθεί (θα «τρέξει») και θα παράγει σαν αποτέλεσμα, την εμφάνιση του μηνύματος «**ΓΕΙΑ ΣΑΣ !!!**», (βλ. βέλος, **14**).

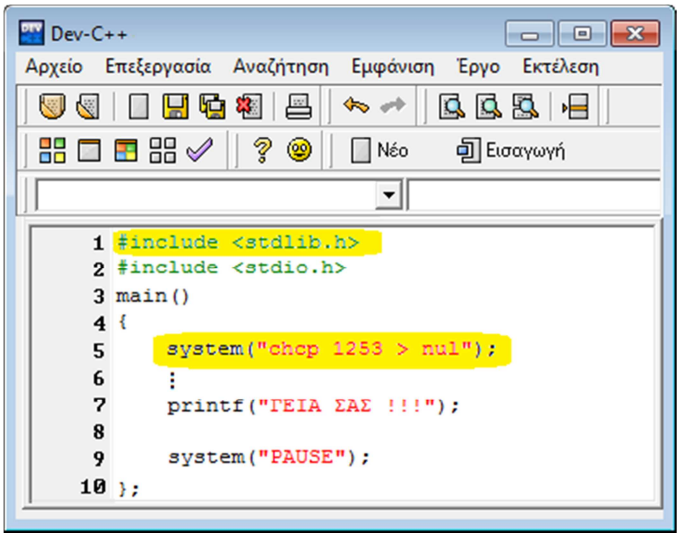


Αυτό το εκτελέσιμο αρχείο με το όνομα **my_first_program.exe** μπορεί να χρησιμοποιηθεί και από οποιονδήποτε άλλον υπολογιστή, ανεξάρτητα εάν διαθέτει το Dev-C++ ή όχι. Εάν για παράδειγμα το αρχείο το επισυνάψουμε σε ένα μήνυμα ηλεκτρονικού ταχυδρομείου και το αποστείλουμε σε έναν άλλο χρήστη, αυτός θα μπορεί να το εκτελέσει και να δει και εκείνος στον δικό του υπολογιστή την εμφάνιση του μηνύματος «**ΓΕΙΑ ΣΑΣ !!!**».

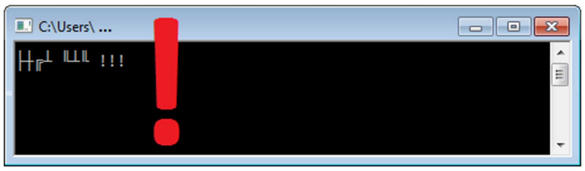


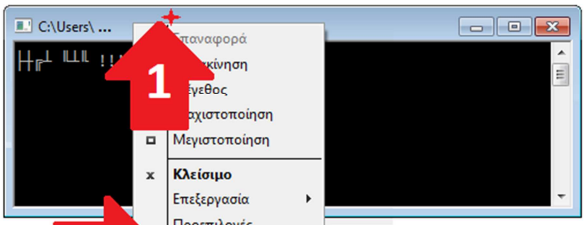
1.6. Ελληνικά και Dev-C++

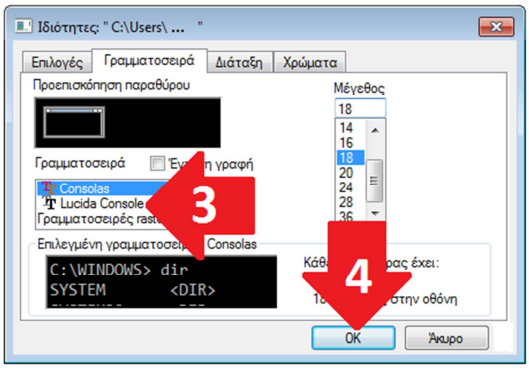
Η διαχείριση των ελληνικών χαρακτήρων από τον μεταγλωττιστή Dev-C++, της BloodShed, δεν αφορά αποκλειστικά και μόνο την προβολή του ελληνικού μενού επιλογών, αλλά συμπεριλαμβάνει και την δυνατότητα της δημιουργίας προγραμμάτων C++, με ελληνικά μενού. Για να καταστεί αυτό δυνατό, θα πρέπει μέσα στα προγράμματα, να καλείται για ενσωμάτωση, η ελληνική γραμματοσειρά με την συνάρτηση, **system ("chcp 1253 > nul")**. Επιπλέον, για να αναγνωριστεί η συνάρτηση **system**, από τον μεταγλωττιστή, θα πρέπει απαραίτητα να προστεθεί στην αρχή κάθε προγράμματος και η βιβλιοθήκη **stdlib.h**, με την εντολή **#include <stdlib.h>**. Οι έννοιες βιβλιοθήκη, συναρτήσεις και οι τρόποι χρήσεως των συναρτήσεων από τις βιβλιοθήκες, επεξηγούνται αναλυτικά σε επόμενο κεφάλαιο του συγγράμματος. Στην ενότητα αυτή, οι έννοιες χρησιμοποιούνται για την επίδειξη του τρόπου με τον οποίο θα πρέπει να ενσωματώνονται μέσα στα προγράμματα C++, ώστε να καθίσταται δυνατή η χρήση των ελληνικών.



Για την ολοκλήρωση της διαδικασίας ενεργοποίησης των ελληνικών, θα πρέπει επίσης, αμέσως μετά από την εγκατάσταση του μεταγλωττιστή και μόνο για την πρώτη φορά (καθώς η ρύθμιση, καθίσταται αυτόματα μόνιμη) που θα «τρέξει» ένα πρόγραμμα το οποίο τυπώνει ελληνικά, να εκτελεστεί και μια αναγκαία ρύθμιση. Την απαραίτητη αυτή ρύθμιση αποτυπώνουν στην επόμενη εικόνα, τα βέλη, **1**, **2**, **3** και **4**.


C:\Users\ ...


C:\Users\ ...
1
2
Μεγιστοποίηση
Κλείσιμο
Επεξεργασία
Προεπιλογές
Ιδιότητες


Ιδιότητες: " C:\Users\ ... "
Επιλογές
Γραμματοσειρά
Διάταξη
Χρώματα
Προεπισκόπηση παραθύρου
Μέγεθος
18
Γραμματοσειρά
Consolas
Lucida Console
3
4
C:\WINDOWS> dir
SYSTEM <DIR>
OK
Άκυρο


C:\Users\ ...
ΓΕΙΑ ΣΑΣ !!!

Σύνοψη κεφαλαίου

Στο πρώτο μέρος του πρώτου κεφαλαίου παρουσιάστηκε μια βασική ιστορική αναφορά, στην γλώσσα προγραμματισμού C, την ιστορία των δημιουργών της και τις γλώσσες «απογόνους» που ακολούθησαν και ακολουθούν. Στο δεύτερο μέρος του κεφαλαίου παρουσιάστηκαν αναλυτικά, με οδηγίες και εικόνες, όλες εκείνες οι απαραίτητες διαδικασίες για να εγκατασταθεί αυτόνομα σε έναν ηλεκτρονικό υπολογιστή, ένας μεταγλωττιστής της γλώσσας προγραμματισμού C++. Συμπερασματικά, οι εκπαιδευόμενοι απέκτησαν όλα τα κατάλληλα ψηφιακά εργαλεία για να συγγράψουν προγράμματα C++ και καθοδηγούμενοι από το εκπαιδευτικό υλικό της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ACTA, μπορούν να συνεχίσουν την επιμόρφωση τους στις γλώσσες προγραμματισμού.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Για την επιστήμη της *Πληροφορικής*, γνωρίζεται ποια είναι και πως ονομάζεται, η ύψιστη σε παγκόσμιο επίπεδο επιστημονική διάκριση;

Ερώτηση-2: Η γλώσσα προγραμματισμού C++, είναι κατά την γνώμη σας μια γλώσσα «υψηλού επιπέδου», ή μια γλώσσα «χαμηλού επιπέδου». Σχολιάστε την άποψη σας;

Ερώτηση-3: Τι ονομάζουμε C-like γλώσσες;

Ερώτηση-4: Δώστε 3 ονόματα από ομοιάζουσες γλώσσες, «απογόνους» της C;

Ερώτηση-5: Περιγράψτε, τι σημαίνει, ολοκληρωμένο περιβάλλον ανάπτυξης μιας γλώσσας (I.D.E. - Integrated Development Environment);

Ερώτηση-6: Ένα πηγαίο πρόγραμμα της γλώσσας C++, με ποιο όνομα επέκτασης (extension) δημιουργείται και αποθηκεύεται στον σκληρό δίσκο;

Ερώτηση-7: Ένα εκτελέσιμο πρόγραμμα της γλώσσας C++, με ποιο όνομα επέκτασης (extension) δημιουργείται και αποθηκεύεται στον σκληρό δίσκο;

Ερώτηση-8: Το πηγαίο αρχείο ενός προγράμματος πως δημιουργείται;

Ερώτηση-9: Περιγράψτε, τι σημαίνει ο όρος, compilation;

Ερώτηση-10: Είναι σωστό ότι, ένα εκτελέσιμο αρχείο της γλώσσας C++, μπορεί να τρέξει μόνο στο υπολογιστή που δημιουργήθηκε;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 1.1. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 1.2. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 1.2. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 1.4. του κεφαλαίου.

Απάντηση-6: Με την επέκταση **cpp**, συμβουλευτείτε και την ενότητα 1.5. του κεφαλαίου.

Απάντηση-7: Με την επέκταση **exe**, συμβουλευτείτε και την ενότητα 1.5. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 1.5. του κεφαλαίου.

Απάντηση-9: Συμβουλευτείτε την ενότητα 1.5. του κεφαλαίου.

Απάντηση-10: Όχι, δεν είναι σωστό, μπορεί να τρέξει και σε κάθε άλλο υπολογιστή αφού είναι εκτελέσιμο (αρκεί να έχουν το ίδιο λειτουργικό σύστημα).

2. ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΗΝ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++

Σκοπός και επιμέρους στόχοι

Ο σκοπός του δεύτερου κεφαλαίου είναι να παρουσιάσει στους εκπαιδευόμενους τα βασικά δομικά στοιχεία της γλώσσας C++, έτσι ώστε αυτά να χρησιμοποιηθούν στην συνέχεια, για την σύνταξη εντολών προγραμματισμού και την κατασκευή προγραμμάτων, μέσω του ολοκληρωμένου περιβάλλοντος εργασίας, Dev-C++, της BloodShed.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του δεύτερου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να αναγνωρίζουν τους βασικούς κανόνες λειτουργίας της γλώσσας C++,
- να αναγνωρίζουν τις λέξεις κλειδιά της γλώσσας C++,
- να αντιλαμβάνονται την σημασία της ενσωμάτωσης βιβλιοθηκών, στα προγράμματα της C++,
- να περιγράψουν ποιο μπορεί να είναι το περιεχόμενο μιας βιβλιοθήκης C++,
- να επιλέγουν κατάλληλες ονοματοδοσίες για το περιεχόμενο των προγραμμάτων τους,
- να τηρούν τους κανόνες του δομημένου προγραμματισμού και να συγγράφουν δομημένα προγράμματα,
- να αναφέρουν τουλάχιστον 3 χαρακτηριστικά που αφορούν την έννοια δομημένος προγραμματισμός,
- να καταλαβαίνουν γιατί θα πρέπει να ενσωματώνουν στα προγράμματα τους επεξηγηματικά σχόλια,
- να περιγράψουν τι είναι οι συναρτήσεις μέσα στα προγράμματα της C++,
- να αντιλαμβάνονται γιατί πρέπει να διασπούν τα προγράμματα τους σε μικρές αυτόνομες λειτουργικές μονάδες, στα πλαίσια της τεχνικής «διαίρει και βασίλευε»,
- να περιγράψουν τι είναι, τα ορίσματα εισόδου και εξόδου, σε μια συνάρτηση,
- να αντιλαμβάνονται, πως μια συνάρτηση μπορεί να καλέσει μια άλλη συνάρτηση.

Έννοιες – Λέξεις Κλειδιά

Αλφάβητο γλώσσας, Κατάλληλη ονοματοδοσία, Βιβλιοθήκες, Συναρτήσεις, Αρχεία επικεφαλίδων, Preprocessing directive, Προ-μεταφραστής, Μεταφραστής, Ορίσματα εισόδου, Ορίσματα εξόδου, Διαδικασιακός προγραμματισμός, Νήματα εκτέλεσης, Δομημένος προγραμματισμός.

Εισαγωγικές Παρατηρήσεις

Η γλώσσα προγραμματισμού C++, είναι μια ιδιαίτερα πλούσια σε δυνατότητες, πλατφόρμα προγραμματισμού. Το σύνολο των βασικών εντολών και των μηχανισμών που περιλαμβάνει μπορεί να δημιουργήσει την κατάλληλη «κουλτούρα προγραμματιστή» και για κάθε άλλη γλώσσα προγραμματισμού. Παρόλο που είναι δύσκολο να περιγραφούν σε ένα βιβλίο, όλες οι επιμέρους δυνατότητές της γλώσσας, το υλικό του συγκεκριμένου κεφαλαίου του συγγράμματος, παρουσιάζει με επιμέλεια όλες τις απαραίτητες βασικές έννοιες προγραμματισμού για την C++. Η μελέτη και η αφομοίωση αυτών των βασικών γνώσεων του κεφαλαίου, θα πρέπει να θεωρείται αναγκαία από τους εκπαιδευόμενους, καθώς διευκολύνει την δημιουργία της απαραίτητης στερεής θεωρητικής βάσης για την κατανόηση, στην συνέχεια, των περαιτέρω τεχνικών δυνατοτήτων της γλώσσας προγραμματισμού C++.

2.1. Το Αλφάβητο

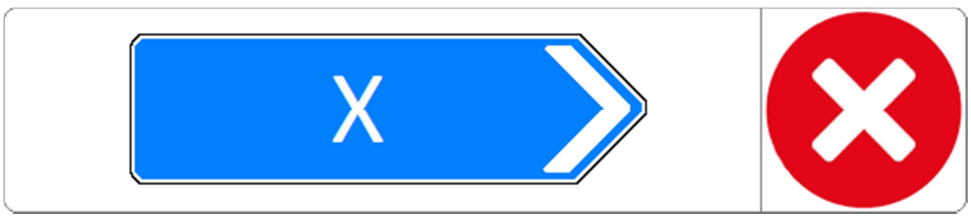
Το αλφάβητο της γλώσσας C++, περιλαμβάνει,

- κεφαλαία γράμματα του ελληνικού αλφαβήτου (Α-Ω),
- πεζά γράμματα του ελληνικού αλφαβήτου (α-ω),
- πεζά γράμματα του ελληνικού αλφαβήτου με τόνους (α-ω),
- κεφαλαία γράμματα του λατινικού αλφαβήτου (A-Z),
- πεζά γράμματα του λατινικού αλφαβήτου (a-z),
- αριθμητικά ψηφία (0-9),
- τα 29 ειδικά σύμβολα,

  _ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " ’
- τον κενό χαρακτήρα.

2.2. Η Ονοματοδοσία

Η κατάλληλη *ονοματοδοσία,* για τις γλώσσες προγραμματισμού είναι μια πολύ σημαντική έννοια και χρησιμοποιείται για να περιγράψει, την ανάγκη επιλογής και χρήσης κατάλληλων ονομάτων μέσα στα προγράμματα. Ένα κατάλληλο όνομα σε ένα πρόγραμμα, είναι απαραίτητο να επιλεγεί, όταν θα πρέπει να δηλωθεί μια νέα μεταβλητή, μια σταθερά, μια συνάρτηση, ένα σημείο ανακατεύθυνσης, ένας νέος τύπος δεδομένων, κ.λπ. Στην βάση των κανόνων του δομημένου προγραμματισμού, είναι απαραίτητο στα προγράμματα, να επιλέγονται περιγραφικά ονόματα.



Ο ορισμός μιας συνάρτησης, απλά με το γράμμα **X**, δεν υποδηλώνει απολύτως τίποτα και το μόνο που κάνει, είναι να προκαλεί σύγχυση. Είναι φτωχή εικόνα, να παρουσιάζονται προγράμματα (ακόμα και εκπαιδευτικά) με επιλογές ονομάτων, **X**, **Y**, **Z**, **A**, **B**, **I**, για συναρτήσεις

και μεταβλητές. Τα περιγραφικά ονόματα μέσα στα προγράμματα υποβοηθούν τους προγραμματιστές στο απαιτητικό έργο της ανάπτυξης και της συντήρησης του λογισμικού.

Τα επιλεγόμενα ονόματα μέσα σε ένα πρόγραμμα C++, μπορούν να αποτελούνται από γράμματα του λατινικού αλφαβήτου (πεζά ή κεφαλαία), ψηφία (0-9) και συνήθως τους χαρακτήρες, κάτω παύλα (underscore) (_) και κανονική παύλα (-). Η γλώσσα C++, γενικά δεν επιβάλει περιορισμούς στο μήκος των επιλεγόμενων ονομάτων, αλλά είναι καλό, να αναζητείται σχετική πληροφόρηση από το *επίσημο τεχνικό εγχειρίδιο* (reference guide manual) του κατασκευαστή της έκδοσης της γλώσσας. Τα ονόματα δεν επιτρέπεται να αρχίζουν από αριθμητικό ψηφίο και δεν επιτρέπεται επίσης η χρήση των δεσμευμένων λέξεων της C++. Ο χαρακτήρας της κάτω παύλας (underscore) (_) μπορεί να χρησιμοποιηθεί σαν πρώτος χαρακτήρας σε ένα όνομα, αλλά καθώς αυτό εφαρμόζεται μέσα στις «εργοστασιακές» συναρτήσεις, είναι καλό να αποφεύγεται.

Η γλώσσα C++, κάνει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων και όπως αυτό συνηθίζεται να περιγράφεται στις κοινότητες των προγραμματιστών, η γλώσσα από την κατασκευή της, είναι case sensitive. Αυτό σημαίνει ότι, εάν ο προγραμματιστής κάνει χρήση μέσα στο πρόγραμμα, μιας μεταβλητής με το όνομα **w_mia_metavliti** και σε μια επόμενη εντολή κάνει χρήση του ονόματος **w_Mia_metavliti**, ο μεταφραστής (compiler) θα θεωρήσει ότι λόγω του κεφαλαίου **M**, έχει να αντιμετωπίσει δυο διαφορετικές μεταβλητές. Αντίθετα, σε μια γλώσσα που δεν είναι case sensitive, μια μεταβλητή η οποία γράφεται με πεζά και λίγο παρακάτω στο ίδιο πρόγραμμα αναφέρεται, με έστω και έναν χαρακτήρα κεφαλαίο, η γλώσσα δεν κάνει διάκριση και θεωρεί ότι είναι η ίδια μεταβλητή.

2.3. Τα Σχόλια

Όπως όλες οι γλώσσες έτσι και η C++, δίνει την δυνατότητα στον προγραμματιστή να τοποθετήσει σχόλια οπουδήποτε μέσα στο πρόγραμμα. Τα σχόλια λειτουργούν σαν επεξηγηματικά κείμενα και υποβοηθούν το έργο της τεκμηρίωσης του προγράμματος. Τα σχόλια δεν λαμβάνονται υπόψη από τις γλώσσες προγραμματισμού, δεν μεταφράζονται και δεν εκτελούνται. Για να τοποθετήσουμε σχόλια μέσα σε ένα πρόγραμμα C++, πρέπει να «ανοίξουμε» με την ειδική δήλωση **/***, να γράψουμε το σχόλιο και να «κλείσουμε» με την ειδική δήλωση ***/**,

```
/*
αυτό είναι ένα σχόλιο πολλών γραμμών και αυτή η 1η γραμμή
αυτό είναι ένα σχόλιο πολλών γραμμών και αυτή η 2η γραμμή
αυτό είναι ένα σχόλιο πολλών γραμμών και αυτή η 3η γραμμή
*/
```

επίσης, εάν θέλουμε να εισάγουμε σχόλια σε μια μόνο γραμμή μπορούμε να το δηλώσουμε, με την ειδική δήλωση **//**,

```
// αυτό είναι ένα σχόλιο μιας γραμμής
```

Βασικός άτυπος κανόνας στον δομημένο προγραμματισμό, για κάθε γλώσσα, είναι η τοποθέτηση σχολίων, διάσπαρτων μέσα στο πραγματικό κώδικα, τα οποία λειτουργούν σαν υποβοηθητικά επεξηγηματικά κείμενα, για κάθε αναγνώστη, ακόμα και του ίδιου μας του εαυτού. Πρέπει πάντα να λαμβάνουμε υπόψη ότι, εάν προσπαθήσουμε να διαβάσουμε μετά από καιρό ένα σύνθετο πρόγραμμα, θα είναι δύσκολο να θυμηθούμε τι ακριβώς κάνει, ακόμα και εάν το έχουμε γράψει εμείς οι ίδιοι.

2.4. Το Λεξιλόγιο (οι Δεσμευμένες Λέξεις)

Η γλώσσα προγραμματισμού C++, μπορεί να αναγνωρίζει ένα συγκεκριμένο λεξιλόγιο το οποίο μπορεί να διαφοροποιείται ανάλογα με την έκδοση που χρησιμοποιεί ο προγραμματιστής. Για παράδειγμα, με βάση το πρότυπο που εκδόθηκε από τον παγκόσμιο οργανισμό ISO, για την

γλώσσα C++, το έτος 2020 και είναι γνωστό σαν πρότυπο ISO 2020, ή σαν πρότυπο C++20, το λεξιλόγιο, περιλαμβάνει τις λέξεις,

Δεσμευμένες Λέξεις κατά το πρότυπο C++20			
alignas	const_cast	int	static_assert
alignof	consteval	long	static_cast
and	constexpr	mutable	struct
and_eq	constinit	namespace	switch
asm	continue	new	template
auto	decltype	noexcept	this
bitand	default	not	thread_local
bitor	delete	not_eq	throw
bool	do	nullptr	true
break	double	operator	try
case	dynamic_cast	or	typedef
catch	else	or_eq	typeid
char	enum	private	typename
char16_t	explicit	protected	union
char32_t	export	public	unsigned
char8_t	extern	register	using
class	false	reinterpret_cast	virtual
co_await	float	requires	void
co_return	for	return	volatile
co_yield	friend	short	wchar_t
compl	goto	signed	while
concept	if	sizeof	xor
const	inline	static	xor_eq

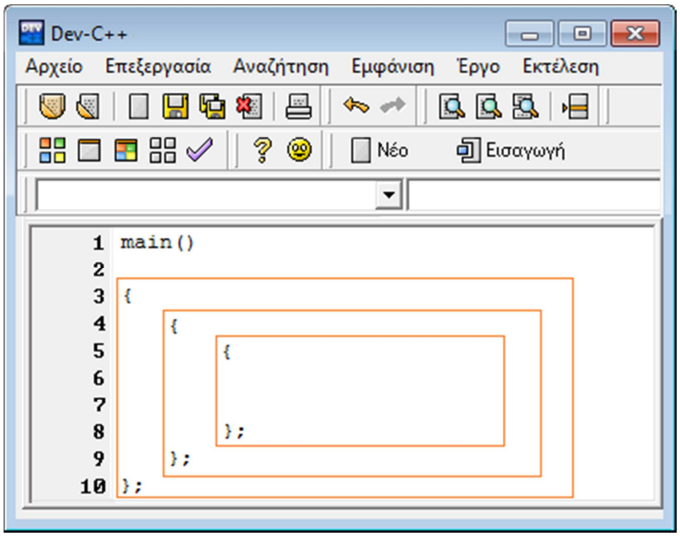
Με αυτές τις λέξεις και την γνώση της γραμματικής, του συντακτικού και της σημασιολογίας της γλώσσας ο προγραμματιστής, καλείται να κατασκευάσει κάθε πιθανό αλγόριθμο. Κάποιες από τις λέξεις κλειδιά, μπορούν εναλλακτικά να δηλωθούν με αντίστοιχους συμβολισμούς,

and	&&	or	||	compl	~
and_eq	&=	xor_eq	^=	not_eq	!=
bitor	|	xor	^	bitand	&
or_eq	|=	not	!		

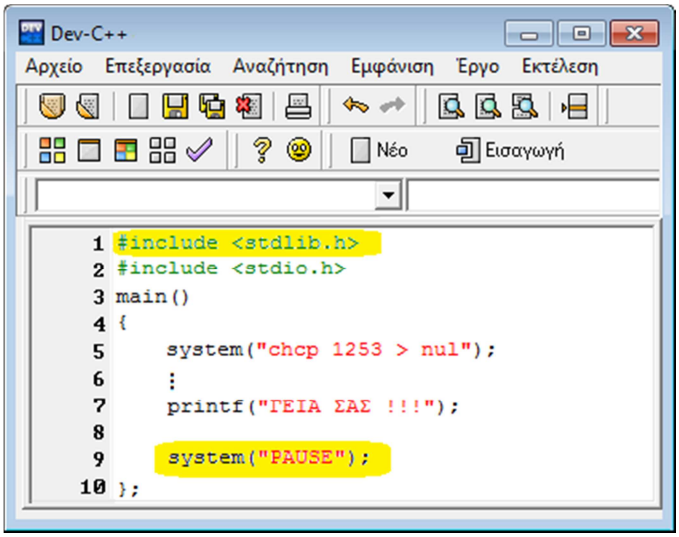
Καθώς οι λέξεις κλειδιά είναι δεσμευμένες, από την ίδια την γλώσσα, είναι φυσικό να μην μπορούν να χρησιμοποιηθούν από τον προγραμματιστή, για να αποδοθούν σαν ονόματα σε μεταβλητές ή σε άλλα σημεία του προγράμματος (ρουτίνες, συναρτήσεις, σημεία ανακατεύθυνσης, κ.λπ.). Οι λέξεις κλειδιά, ονομάζονται επίσης, *keywords*, «*εργοστασιακές λέξεις*», *δεσμευμένες λέξεις* ή *reserved words*. Το λεξιλόγιο της C++, κατά το ISO 2020, είναι ελαφρά εμπλουτισμένο, σε σχέση με το προηγούμενο πρότυπο C++17 του 2017.

2.5. Βασικοί Κανόνες Σύνταξης, Σύνολα Εντολών και παύση ροής

Συντάσσοντας με τον απαραίτητο τρόπο τις εντολές της C++, θα πρέπει να λαμβάνεται υπόψη ότι πρέπει απαραίτητα να υποδεικνύεται στον compiler και το σημείο στο οποίο τελειώνει η κάθε εντολή. Το σημείο αυτό, ορίζεται με την χρήση του συμβόλου του ελληνικού ερωτηματικού **;** (semicolon) το οποίο τοποθετείται στο τέλος της κάθε εντολής και οριοθετεί το τέλος των λέξεων που αφορούν την εντολή. Το σύμβολο **;** στην C++, χρησιμοποιείται όπως το σύμβολο της τελείας **.** στα γραπτά κείμενα, που υποδεικνύει το τέλος της κάθε πρότασης. Γενικότερα, κάθε εντολή της C++, πρέπει να τελειώνει με ένα ελληνικό ερωτηματικό **;**. Η υποχρέωση της χρήσης του ελληνικού ερωτηματικού **;** αφορά και την τελευταία λέξη για όλες τις εντολές με σύνθετη συντακτική δομή.



Επιπλέον, όταν κάποιες εντολές ανήκουν, σαν σώμα, μέσα σε μια άλλη εντολή θα πρέπει αυτό το σύνολο εντολών (block) να οριοθετείται με άγκιστρα **{ }**. Ένα block εντολών μπορεί να ενσωματώνεται ολόκληρο μέσα σε ένα άλλο block εντολών και πρέπει επίσης και αυτό να οριοθετείται με εσωτερικά άγκιστρα **{ }**.



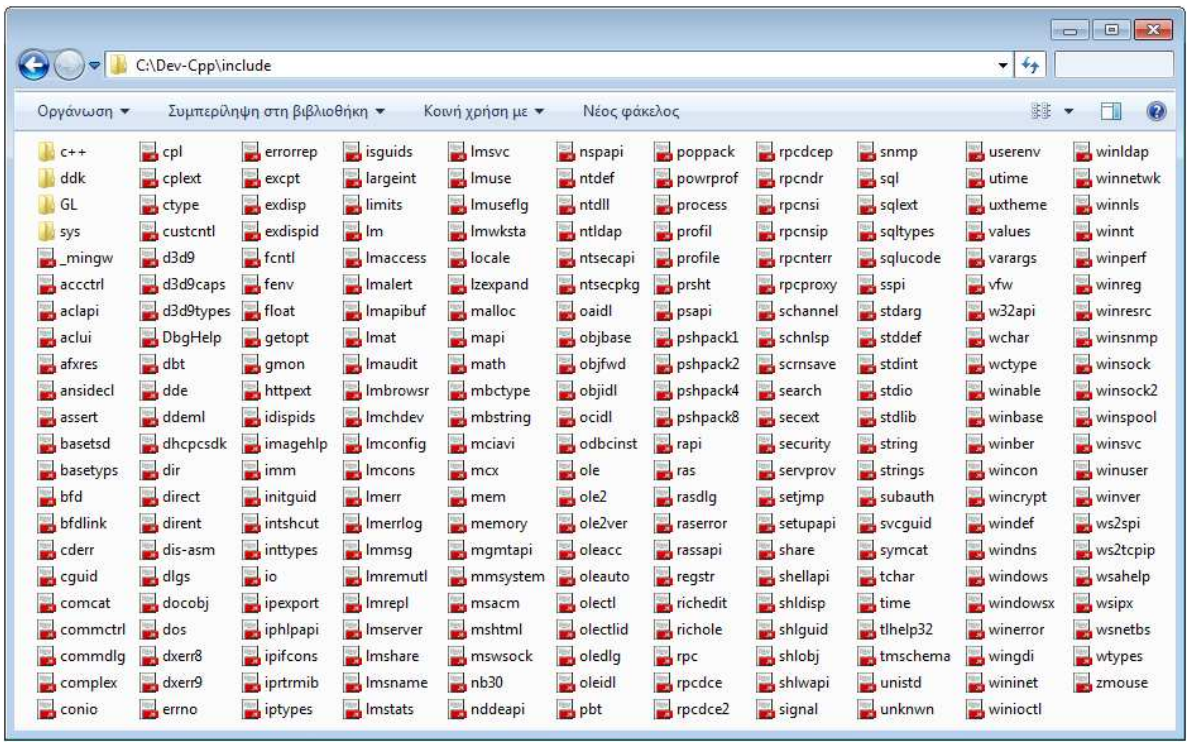
Στα προγράμματα που στην συνέχεια θα παρουσιάσουμε και θα μελετήσουμε θα χρειαστεί να συμπεριλάβουμε και μια τελευταία πρόσθετη ρύθμιση με την μορφή μιας τελευταίας εντολής στο τέλος κάθε προγράμματος. Η εντολή αυτή θα έχει σαν αποτέλεσμα την προσωρινή παύση της ροής του προγράμματος, σαν το πλήκτρο «παύση» (ή pause) μιας κάμερας. Ο λόγος τοποθέτησης της συγκεκριμένης δήλωσης είναι για να προλαβαίνουμε να βλέπουμε τις

τελευταίας ενέργειες του προγράμματος μας. Για να γίνει αυτό, τοποθετούμε στο σημείο που θέλουμε να γίνεται η προσωρινή διακοπή, την συνάρτηση, **system ("PAUSE")**. Η συνάρτηση **system**, προϋποθέτει την ενσωμάτωση της βιβλιοθήκης **stdlib.h**, με την εντολή **#include <stdlib.h>**. Οι έννοιες βιβλιοθήκη και συναρτήσεις, επεξηγούνται αναλυτικά στην συνέχεια του συγγράμματος.

2.6. Βιβλιοθήκες, Συναρτήσεις και Αρχεία Επικεφαλίδων

Όλες οι γλώσσες προγραμματισμού, εκτός από τις βασικές ενσωματωμένες εντολές και λειτουργίες, που διαθέτουν, συνοδεύονται και από κάποιες συλλογές με έτοιμες εξωτερικές βιβλιοθήκες. Οι βιβλιοθήκες αυτές, είναι εξωτερικά στοιχεία που παρέχονται από τους κατασκευαστές, σαν έτοιμες λειτουργικές μονάδες οι οποίες συνοδεύουν την γλώσσα και είναι προορισμένες, όταν καλούνται, να ενσωματώνονται για να μπορούν να επιτελούν λειτουργίες συναφούς τύπου, μέσω της κλήσης συγκεκριμένων συναρτήσεων. Για τις γλώσσες προγραμματισμού η έκφραση *συνάρτηση,* αποτελεί ένα σημαντικό στοιχείο της ορολογία τους, καθώς χρησιμοποιείται για να περιγράψει, με ένα συγκεκριμένο όνομα, την οριοθέτηση μιας ομάδας εντολών προγραμματισμού οι οποίες έχουν συγγραφεί με σκοπό να επιτελούν μια συγκεκριμένη εργασία. Η λέξη συνάρτηση, στην «αργκό» των προγραμματιστών (την «συνθηματική» τους γλώσσα), ορίζεται και με τα συνώνυμα, *ρουτίνα, υποπρόγραμμα*, *μέθοδος.* Συναφείς συναρτήσεις αποθηκεύονται μέσα σε συγκεκριμένες βιβλιοθήκες. Για παράδειγμα, μια βιβλιοθήκη μπορεί να περιλαμβάνει τις απαραίτητες πρότυπες συναρτήσεις για την διαχείριση του χρόνου, των μηνών, των ωρών, των ονομάτων των ημερών, κ.λπ. Μια άλλη βιβλιοθήκη, μπορεί να περιλαμβάνει τις απαραίτητες πρότυπες συναρτήσεις για την εκτέλεση μαθηματικών πράξεων, όπως maximum, minimum, average, κ.λπ. Μια άλλη βιβλιοθήκη, μπορεί να περιλαμβάνει τις απαραίτητες πρότυπες συναρτήσεις για την ενεργοποίηση των εκτυπωτών, print, skip_page, κ.λπ. Ο λόγος για τον οποίο οι βιβλιοθήκες παρέχονται σαν εξωτερικά στοιχεία και δεν ενσωματώνονται εξαρχής μέσα στην ίδια την γλώσσα είναι καθαρά λόγος προ-οικονομίας. Ο κατασκευαστής της γλώσσας δηλαδή, γνωρίζει ότι το έργο του (η γλώσσα) θα χρησιμοποιηθεί από πολλούς και διαφορετικούς τύπους προγραμματιστών. Συνεπώς, κάθε τύπος προγραμματιστή μπορεί να χρειάζεται μια λειτουργία ή οποία όμως θα είναι αδιάφορη σε κάποιους άλλους. Για ποιον λόγο λοιπόν να δημιουργήσει

μια γλώσσα μεγαλύτερη σε όγκο (και άρα μικρότερη σε ταχύτητα) για κάτι το οποίο θα χρησιμοποιηθεί, από έναν.



Αυτή ακριβώς είναι η ιδέα. Όταν χρειαζόμαστε κάτι, πρέπει να το ζητήσουμε από την γλώσσα να το ενσωματώσει (να το «φορτώσει», στην «αργκό» των προγραμματιστών) με την κλήση της κατάλληλης βιβλιοθήκης. Γενικότερα, οι κατασκευαστές αλλά και οι κοινότητες των προγραμματιστών, παρέχουν πλούσιες συλλογές από έτοιμες βιβλιοθήκες για κάθε ανάγκη και κάθε απαίτηση. Πολλές βιβλιοθήκες διατίθενται δωρεάν και κάποιες πωλούνται εμπορικά. Στην ορολογία της γλώσσας C++, οι βιβλιοθήκες, ονομάζονται *αρχεία επικεφαλίδων* (header files). Για την ενσωμάτωση ενός αρχείου επικεφαλίδων μέσα σε ένα πρόγραμμα C++, θα πρέπει να γίνει δήλωση-κλήση προς τον μεταφραστή (compiler) με την εντολή ενσωμάτωσης βιβλιοθηκών, **#include**. Η εντολή **include** (ενσωμάτωσε) συντάσσεται μαζί με το σύμβολο **#**, το οποίο προηγείται από την καθαυτή εντολή και αποτελεί μια ειδική εντολή που ονομάζεται «*οδηγία*» (directive), ή *preprocessing directive,* ή «*ντιρεκτίβα*» (στην ελληνική «αργκό» προγραμματισμού). Η «ντιρεκτίβα» απευθύνεται σε ένα ειδικό σύστημα της γλώσσας που ονομάζεται *προ-μεταφραστής*, ή *pre-processor,* ή *pre-compiler* και από το οποίο «ζητείται», στο σημείο αυτό, να ενσωματώσει στο πηγαίο πρόγραμμα κάποια βιβλιοθήκη, πριν να

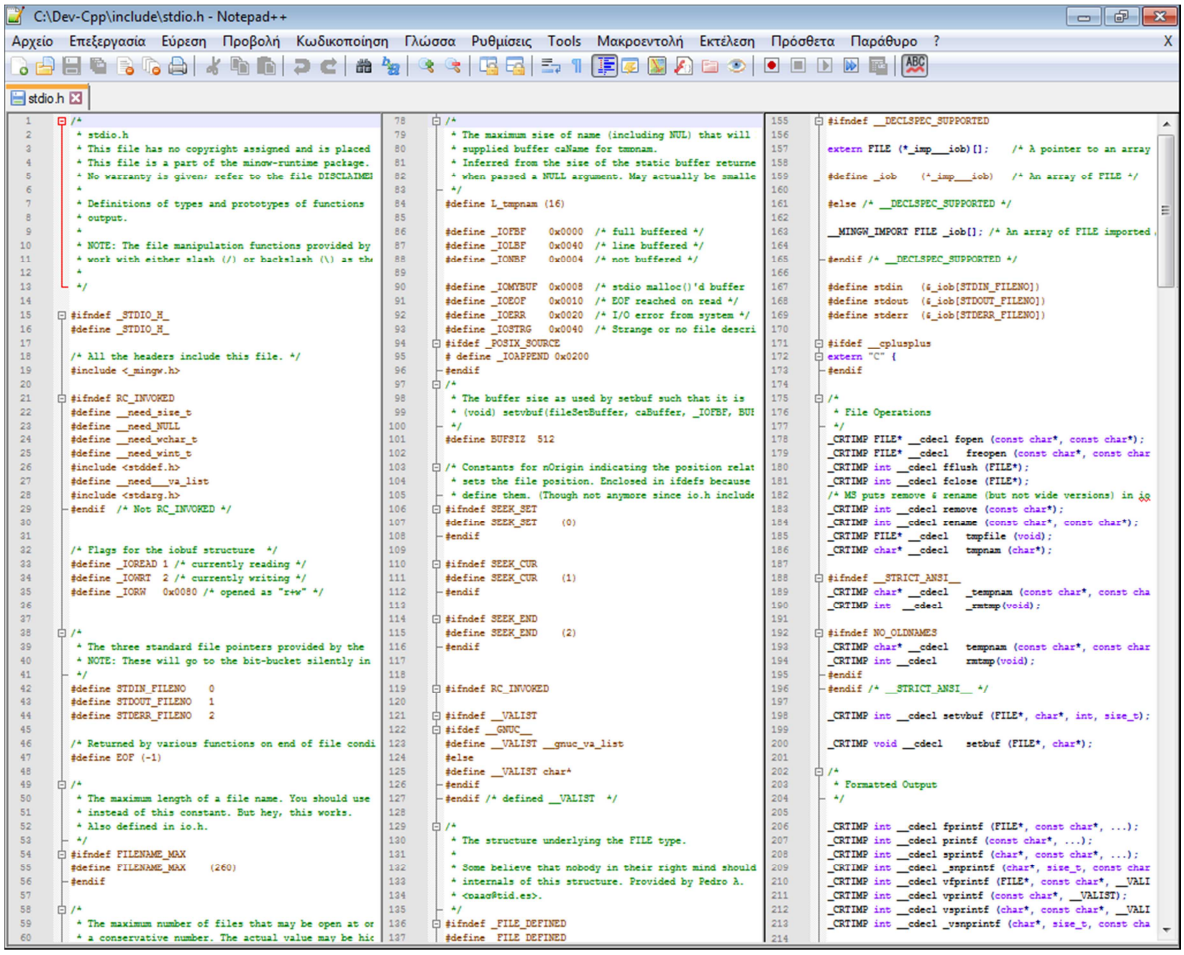
αναλάβει ο μεταφραστής (compiler) να μεταφράσει το πηγαίο πρόγραμμα (source), σε εκτελέσιμο αρχείο (executable). Προ-μεταφραστή (pre-compiler) διαθέτουν όλες οι γλώσσες προγραμματισμού και η δουλειά του συστήματος αυτού, είναι να εκτελεί μια μορφή προεπεξεργασίας του προγράμματος, (σε επίπεδο κειμένου), προτού προχωρήσει η κανονική μεταγλώττιση από τον compiler. Στην γλώσσα C++, οι βιβλιοθήκες περιέχουν σειρά πρότυπων συναρτήσεων και είναι αρχεία, με κατάληξη (extension) το γράμμα **h**, το οποίο προέρχεται από το όνομα header. Οι δηλώσεις ενσωμάτωσης αρχείων επικεφαλίδων, τοποθετούνται στην αρχή κάθε προγράμματος και η γενική τους σύνταξη είναι,

#include <όνομα_αρχείου_επικεφαλίδας.h>

Για παράδειγμα, η εντολή **#include <stdio.h>**, στην πρώτη γραμμή ενός προγράμματος C++, θα προκαλέσει την εισαγωγή του header αρχείου, **stdio.h** μέσα στο πρόγραμμα το οποίο στην συνέχεια, δίνει την δυνατότητα στον προγραμματιστή να χρησιμοποιήσει μέσω κλήσης συγκεκριμένων έτοιμων συναρτήσεων το πληκτρολόγιο και την οθόνη για είσοδο και έξοδο αντίστοιχα. Στην βιβλιογραφία, αναφέρεται και σαν, βιβλιοθήκη για προκαθορισμένες συσκευές εισόδου και εξόδου (**st**an**d**ard **i**nput **o**utput). Η σύνταξη συνοδεύεται από τα σύμβολα **< >**, ενώ μπορούν να χρησιμοποιηθούν και τα σύμβολα **“ ”**. Στην πρώτη περίπτωση, η βιβλιοθήκη αναζητείται από τον compiler, στον χώρο (στον υποφάκελο, folder, directory) που προβλέπει για αποθήκευση βιβλιοθηκών, ο κατασκευαστής, ενώ στην δεύτερη περίπτωση η βιβλιοθήκη αναζητείται στον υποφάκελο που βρίσκεται το ίδιο το πρόγραμμα που την καλεί.

Συχνά χρησιμοποιούμενα αρχεία επικεφαλίδων	
#include <stdio.h>	βιβλιοθήκη με χρήσιμες συναρτήσεις για τις συσκευές εισόδου και εξόδου,
#include <math.h>	Βιβλιοθήκη με μαθηματικές συναρτήσεις,
#include <stdlib.h>	βιβλιοθήκη με χρήσιμες συναρτήσεις γενικής φύσεως,
#include <time.h>	περιέχει συναρτήσεις σχετικές με την διαχείριση του χρόνου,
#include <io.h>	περιέχει συναρτήσεις για την διαχείριση των μέσων εισόδου και εξόδου, σε χαμηλό επίπεδο.

Εάν προσπαθήσουμε να διαβάσουμε το περιεχόμενο ενός αρχείου επικεφαλίδων, για παράδειγμα το, **stdio.h**, όπως και κάθε άλλο αρχείο επικεφαλίδας θα περιέχει κώδικα με έτοιμες εντολές C++.



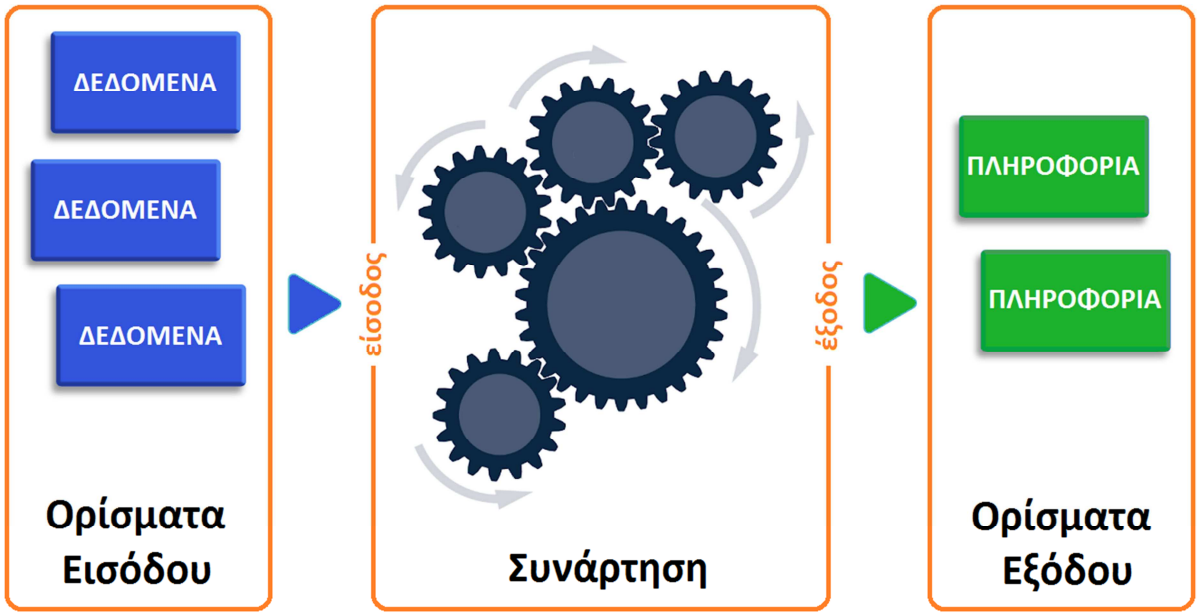
Εκτός από τα «εργοστασιακά» αρχεία επικεφαλίδων (ή βιβλιοθήκες) που παρέχει ο κατασκευαστής, οι προγραμματιστές έχουν την δυνατότητα να δημιουργούν δικά τους αρχεία επικεφαλίδων, με τυποποιημένους έτοιμους κώδικες, τους οποίους μπορούν στην συνέχεια να τους καλούν κατά περίπτωση και να τους επαναχρησιμοποιούν. Στον προγραμματισμό γενικότερα, η λογική της χρήσης έτοιμων βιβλιοθηκών, που δημιουργήθηκαν από εμάς ή ακόμα και από άλλους προγραμματιστές, σε διαφορετικό χρόνο και τόπο, είναι μια ιδιαίτερα συνηθισμένη τακτική. Η κλήση των προσωπικών αρχείων επικεφαλίδων γίνεται ακριβώς με τον ίδιο τρόπο, όπως και με τα «εργοστασιακά» αρχεία επικεφαλίδων. Εκτός από την «ντιρεκτίβα» εντολή ενσωμάτωσης βιβλιοθηκών **include**, υπάρχει και μια πληθώρα άλλων ειδικών εντολών οι οποίες μπορούν να συνταχθούν με το σύμβολο **#** και να δηλωθούν σαν

«ντιρεκτίβες» προς τον μεταφραστή (compiler). Στην περίπτωση αυτή ο compiler ενεργοποιεί τον pre-processor του, ο οποίος θα εκτελέσει ειδικές λειτουργίες, για όλες τις εντολές που θα δει σαν «ντιρεκτίβες», όσες δηλαδή έχουν μπροστά το σύμβολο **#**, και θα επέμβει (ο pre-processor) επάνω στο πηγαίο πρόγραμμα, πριν από την πραγματική μεταγλώττιση. Οι «ντιρεκτίβες», συντάσσονται χωρίς την υποχρέωση του ελληνικού ερωτηματικού **;** στο τέλος της εντολής. Οι δηλώσεις μπορούν να τοποθετηθούν και σε άλλα σημεία, εκτός της αρχής κάθε προγράμματος.

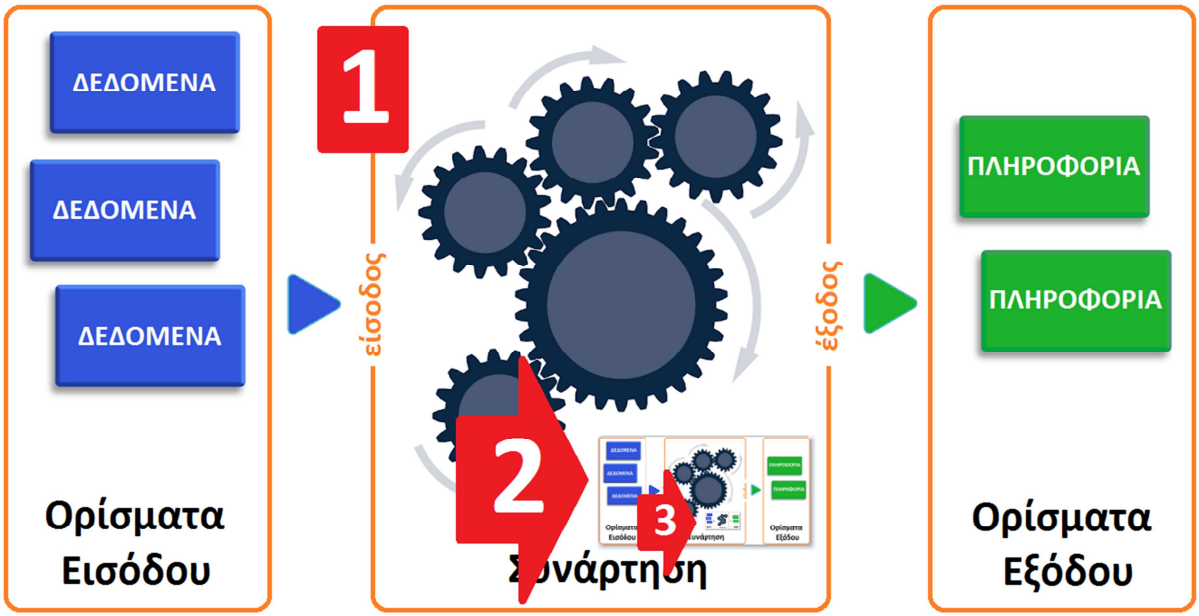
Οδηγίες Προ-μεταφραστή - Preprocessing Directives			
# define	**# endif**	**# ifde**	**# pragma**
# elif	**# error**	**# ifndef**	**# undef**
# else	**# if**	**# line**	

2.7. Ο κορμός ενός Προγράμματος C++

Η κατασκευή ενός προγράμματος σε μια γλώσσα προγραμματισμού μπορεί πολλές φορές να συμπεριλαμβάνει έναν πολύ μεγάλο αριθμό από εντολές προγραμματισμού. Υπάρχουν προγράμματα τα οποία αποτελούνται από πολλές χιλιάδες γραμμές κώδικα. Ένα μεγάλο πρόγραμμα χιλιάδων γραμμών είναι πολύ δύσκολο ή σχεδόν αδύνατο, να συντηρηθεί σωστά και γρήγορα καθώς και η παραμικρή διόρθωση ή επέκταση, αποτελεί προγραμματιστικό πονοκέφαλο αφού εμπεριέχει πολλούς κινδύνους για λάθη και αστοχίες. Για τον λόγο αυτό τα μεγάλα προγράμματα διασπώνται σε μικρότερες μονάδες, στην λογική του «*διαίρει και βασίλευε*», έτσι ώστε να διαχωρίζεται ο κώδικας σε ανεξάρτητα και αυτόνομα κομμάτια τα οποία στην συνέχεια είναι ευκολότερο να συντηρηθούν και να επεκταθούν. Οι μικρότερες αυτές αυτόνομες μονάδες κώδικα μπορούν να υλοποιούν συναρτήσεις, με συγκεκριμένο όνομα και να έχουν σκοπό να επιτελούν συγκεκριμένες εργασίες.

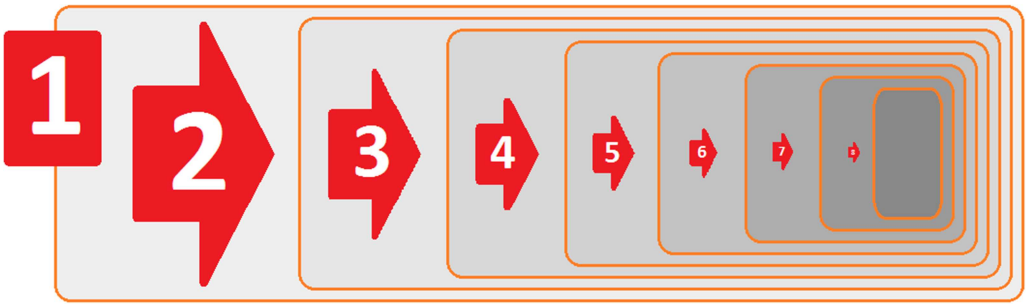


Καθώς, οι συναρτήσεις, είναι ανεξάρτητοι μηχανισμοί κώδικα μπορούν να διαθέτουν είσοδο και έξοδο. Μπορούν συνεπώς, με βάση το τι θα περάσει από μια μεταβλητή στην είσοδο τους («καπελάκι»), να παραμετροποιείται και η εσωτερική συμπεριφορά τους. Εάν δηλαδή, τους δοθεί κάτι σαν δεδομένο εισόδου, να εκτελούν μια συγκεκριμένη λειτουργία ενώ εάν τους δοθεί κάτι άλλο, να εκτελούν κάτι εντελώς διαφορετικό. Τα δεδομένα εισόδου των συναρτήσεων ονομάζονται *ορίσματα εισόδου* της συνάρτησης. Επιπλέον, μια συνάρτηση με την ολοκλήρωση της μπορεί στην έξοδο της να επιστρέφει και κάποιο αποτέλεσμα σε μια άλλη μεταβλητή («καπελάκι»). Οι επιστροφές στην έξοδο ονομάζονται *ορίσματα εξόδου* της συνάρτησης. Μια συνάρτηση μπορεί να έχει, μία, καμία, ή πολλές μεταβλητές εισόδου και επίσης, μία, καμία ή πολλές μεταβλητές εξόδου.

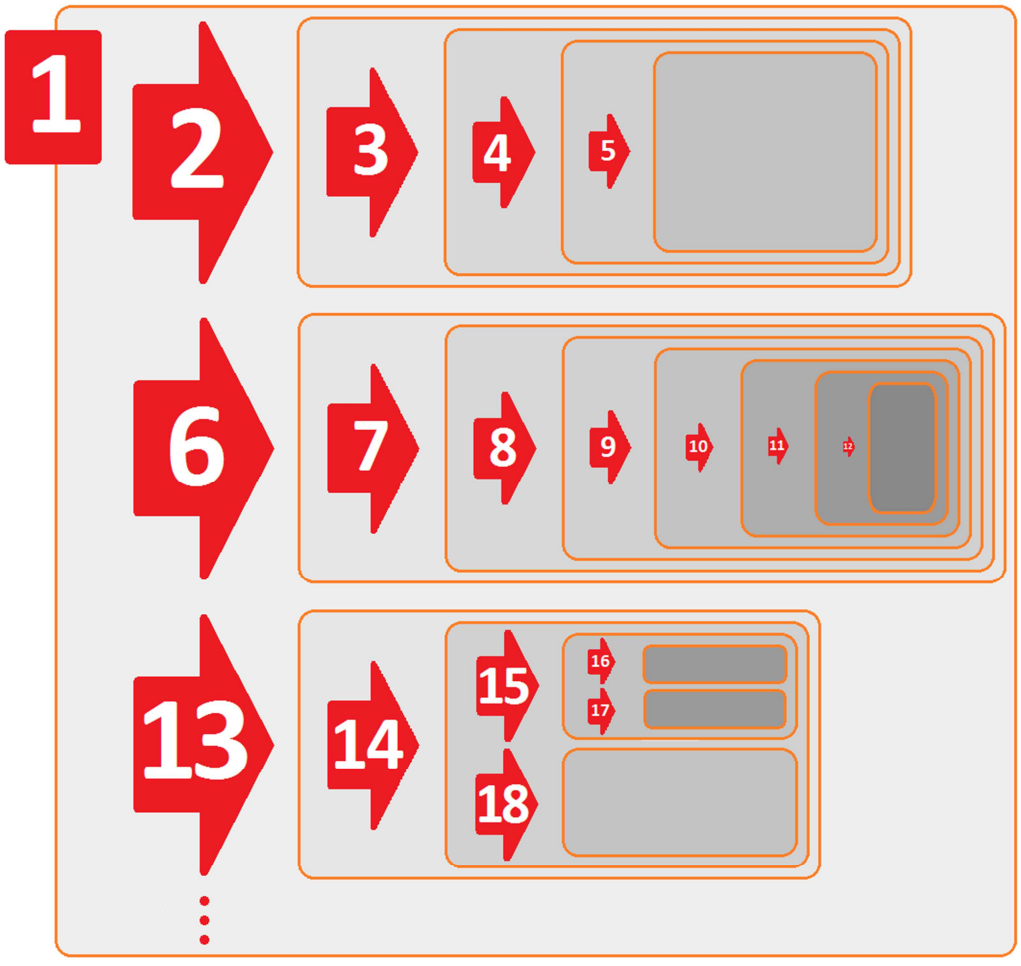


Μια συνάρτηση, εκτός από το σώμα των δικών της εντολών (βλ. ετικέτα, _1_), μπορεί με την σειρά της να περιλαμβάνει την κλήση μιας άλλης συνάρτησης (βλ. βέλος, _2_) στην οποία, εσωτερική συνάρτηση, να περιλαμβάνεται επίσης η κλήση μιας άλλης συνάρτησης (βλ. βέλος, _3_), που και αυτή με την σειρά της να καλεί μια επόμενη συνάρτηση, κ.λπ. Οι γλώσσες προγραμματισμού διαθέτουν την ικανότητα, της διαδοχικής κλήσης πολλών συναρτήσεων, με αποτέλεσμα να καθιστούν την διαχείριση των μεγάλων προγραμμάτων μια σχετικά εύκολη διαδικασία. Αρκεί, ο προγραμματιστής να έχει προσχεδιάσει κατάλληλα τον αλγόριθμο, διαχωρίζοντας τον σε αυτόνομα τμήματα με τα κατάλληλα ορίσματα εισόδου και εξόδου και οριοθετώντας την διαδικασία της μετάβασης από την μια συνάρτηση, στην επόμενη και στην μεθεπόμενη. Η μεθοδολογία της διαδοχικής κλήσης των συναρτήσεων μέσα σε ένα πρόγραμμα, δεν σημαίνει απαραίτητα ότι όλες οι κλήσεις θα πρέπει να γίνονται πάντα από ένα «υψηλότερο» επίπεδο προς ένα «βαθύτερο» επίπεδο που και αυτό με την σειρά του θα μεταφέρει τον έλεγχο (την «μπίλια» ροής) σε κάποιο «βαθύτερο» επίπεδο.

Οι γλώσσες προγραμματισμού διαθέτουν έναν ικανότερο τρόπο διαχείρισης των διαδοχικών κλήσεων αυτόνομων τμημάτων κώδικα. Ειδικότερα, οι συναρτήσεις δεν χρειάζεται απαραίτητα όλες να «βυθίζονται», η μια μέσα στην άλλη,

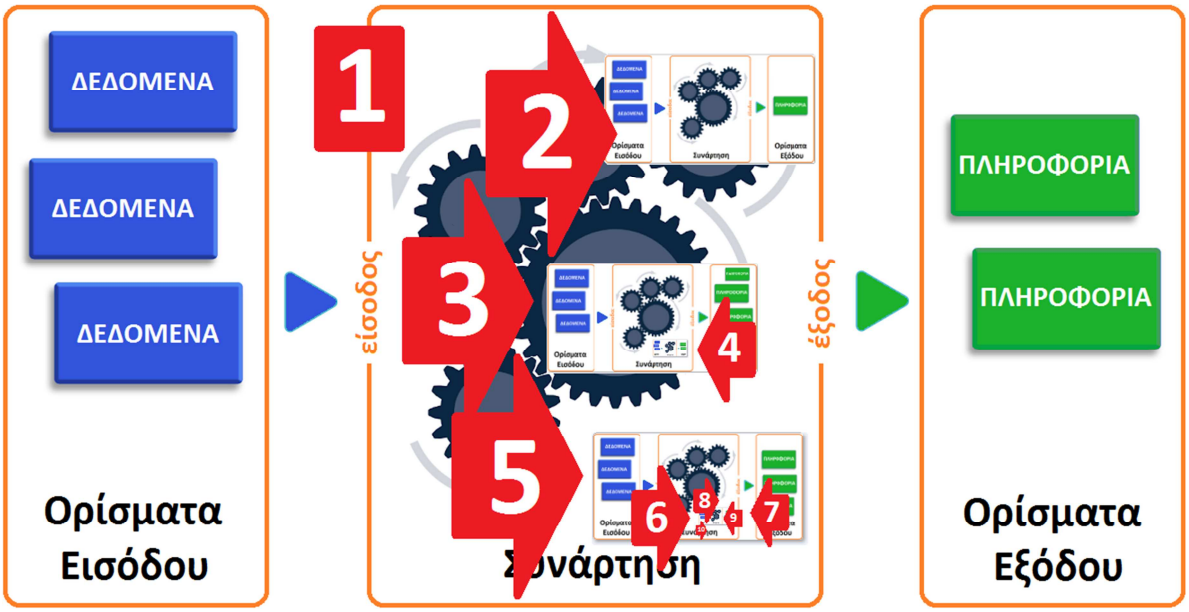


αλλά μπορούν να λειτουργούν σε μια συνδυαστική κατάσταση όπου: α) ή να βρίσκονται, η μια μετά από την άλλη, β) ή να «βυθίζονται», η μια μέσα στην άλλη, γ) ή να συνδυάζονται και τα δυο προηγούμενα.

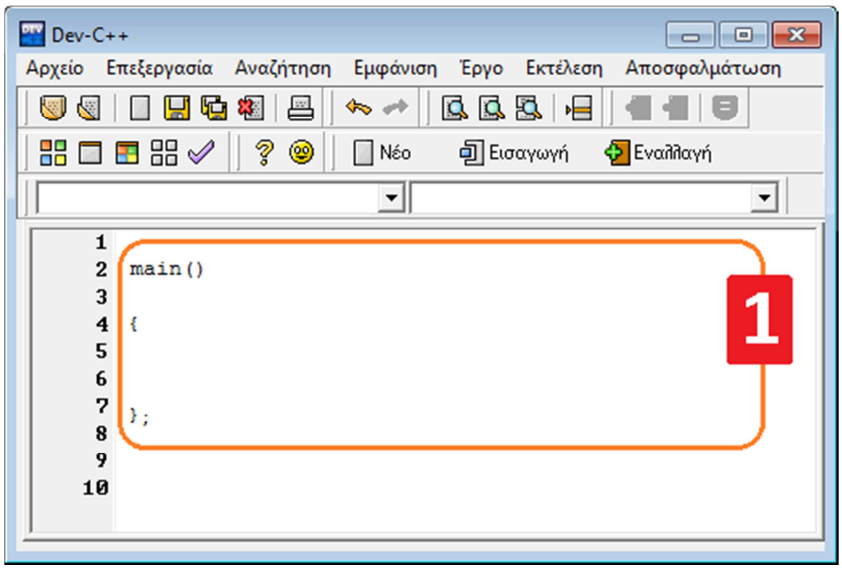


Η «μπίλια» ροής δηλαδή (ο έλεγχος του προγράμματος), μπορεί να ξεκινά από το σημείο έναρξης (βλ. ετικέτα, _1_), να «βυθίζεται» σταδιακά μέσα σε κάποια επίπεδα (βλ. βέλη, _2_, _3_, _4_, _5_), να επιστρέφει σε ένα υψηλότερο επίπεδο (βλ. βέλος, _6_), να «βυθίζεται» πάλι μέσα σε πολλαπλά κατώτερα επίπεδα, (βλ. βέλη, _7_, _8_, _9_, _10_, _11_, _12_), να επιστρέφει σε ένα υψηλότερο επίπεδο (βλ. βέλος, _13_), να «βυθίζεται» πάλι σε κάποια επίπεδα, (βλ. βέλη, _14_, _15_, _16_, _17_), να επιστρέφει σε ένα υψηλότερο επίπεδο (βλ. βέλος, _18_) και τελικά να τερματίζει. Αυτή η ακολουθιακή διαδικασία όπου, μια συνάρτηση καλείται **μέσα** από κάποια άλλη, ή μια συνάρτηση καλείται διαδοχικά **μετά** από κάποια άλλη, είναι στην ουσία αυτό που ονομάζεται *ακολουθιακός* ή *διαδικασιακός προγραμματισμός*. Μια φανταστική «μπίλια» η οποία διατρέχει το σημείο από το οποίο περνά η «αντίληψη» του υπολογιστή εκείνη την στιγμή. Την γραμμή του κώδικα που είναι ενεργή και εκτελείται εκείνη την στιγμή. Μέσα στα προγράμματα που μελετάμε στην σειρά αυτών των συγγραμμάτων, μπορεί να υπάρχει κάθε φορά ενεργή μονό μια «μπίλια», όμως η «μπίλια» αυτή μπορεί να ακολουθεί πολλούς διαφορετικούς δρόμους σε κάθε εκτέλεση του ίδιου προγράμματος, ανάλογα με το τι συμβαίνει μέσα στις μεταβλητές και την πολυπλοκότητα του κώδικα. Οι διαφορετικές αυτές πιθανές διαδρομές που ενδεχομένως θα ακολουθήσει η «μπίλια», σε κάθε εκτέλεση και σε σχέση με αυτά που συναντά στον δρόμο της, ονομάζονται «*νήματα*» εκτέλεσης. Στα μεγάλα προγράμματα ο αριθμός αυτών των διαφορετικών «νημάτων» που μπορεί να διαγράψει σαν πορείες η «μπίλια», είναι τόσο μεγάλος, που οι προγραμματιστές δυσκολεύονται να εκτελέσουν στο μυαλό τους όλες τις πιθανές διαδρομές. Αυτό είναι ένα πολύ μεγάλο πρόβλημα στον προγραμματισμό, καθώς σημαίνει ότι πολλά προγράμματα μπορούν δυνητικά να περιέχουν λάθη που δεν έχουν ακόμα ενεργοποιηθεί διότι η «μπίλια», δεν κατάφερε ποτέ να περάσει από το «νήμα» που τα φιλοξενεί. Για την αντιμετώπιση αυτών των καταστάσεων τα μεγάλα προγράμματα διασπώνται στην λογική του «*διαίρει και βασίλευε*».

Αυτή η κουλτούρα, της διάσπασης του κώδικα σε αυτόνομες συναρτήσεις, δομημένες, με τάξη, επίπεδα, κανόνες και κατάλληλα ορίσματα (εισόδου και εξόδου), ονομάζεται, *δομημένος προγραμματισμός*.



Το σημείο έναρξης (βλ. ετικέτα, 1), για τα προγράμματα της C++, ορίζεται από μια κεντρική συνάρτηση η οποία ονομάζεται, **main()**. Η συνάρτηση αυτή είναι η κεντρική συνάρτηση (βλ. ετικέτα, 1) και από εκεί ξεκινούν για την C++, τα πάντα, δηλαδή οι συναρτήσεις (βλ. βέλη, 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10), που καλούνται η μια κάτω από την άλλη ή οι συναρτήσεις που καλούνται η μια μέσα από την άλλη ή ο συνδυασμός και των δυο.



Σύνοψη κεφαλαίου

Στο δεύτερο κεφάλαιο του συγγράμματος αρχικά έγινε μια παρουσίαση των βασικών δομικών στοιχείων της γλώσσας C++, ώστε να διαμορφωθεί για τους εκπαιδευόμενους το κατάλληλο υπόβαθρο γνώσης για την συνέχεια. Ακολούθησε, η εκτεταμένη αναφορά στις έννοιες των βιβλιοθηκών και των συναρτήσεων της γλώσσας καθώς και στους τρόπους με τους οποίους ενσωματώνονται βιβλιοθήκες και χρησιμοποιούνται οι συναρτήσεις τους. Το κεφάλαιο ολοκληρώθηκε με μια ιδιαίτερα σημαντική αναφορά, από περιγραφικά σχήματα και κατάλληλες επεξηγήσεις για τους μηχανισμούς που εφαρμόζονται στο διαδικασιακό προγραμματισμό και την διαδοχική κλήση των συναρτήσεων.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Για ποιόν λόγο πρέπει, στο προγραμματισμό να χρησιμοποιούμε περιγραφική ονοματοδοσία;

Ερώτηση-2: Στην C++, τι σημαίνει ότι, η γλώσσα από την κατασκευή της, είναι case sensitive;

Ερώτηση-3: Για ποιόν λόγο πρέπει, μέσα στα προγράμματα να τοποθετούμε επεξηγηματικά σχόλια;

Ερώτηση-4: Στην γλώσσα C++, με ποιο σύμβολο πρέπει να δηλώνουμε το τέλος της κάθε εντολής;

Ερώτηση-5: Τι γνωρίζετε για τις βιβλιοθήκες και τα αρχεία επικεφαλίδων, στην γλώσσα προγραμματισμού C++;

Ερώτηση-6: Περιγράψτε, τι γνωρίζετε για τον όρο *συναρτήσεις*, στην γλώσσα C++;

Ερώτηση-7: Περιγράψτε, τι γνωρίζετε για τον όρο «ντιρεκτίβα», στην γλώσσα C++;

Ερώτηση-8: Περιγράψτε, σε μια συνάρτηση στην γλώσσα C++, τι ονομάζουμε, *ορίσματα εισόδου* και τι *ορίσματα εξόδου*;

Ερώτηση-9: Είναι δυνατό σε μια συνάρτηση να μην υπάρχουν καθόλου ορίσματα εισόδου;

Ερώτηση-10: Είναι δυνατό σε μια συνάρτηση να μην υπάρχουν καθόλου ορίσματα εξόδου;

Ερώτηση-11: Περιγράψτε για ποιον λόγο τα μεγάλα προγράμματα πρέπει να διασπώνται σε μικρότερες μονάδες;

Ερώτηση-12: Περιγράψτε, τι σημαίνει ο όρος, *διαδικασιακός προγραμματισμός*;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε την ενότητα 2.2. του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 2.2. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 2.3. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 2.5. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 2.6. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 2.6. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 2.6. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 2.7. του κεφαλαίου.

Απάντηση-9: Ναι, μια συνάρτηση μπορεί να μην έχει κανένα όρισμα εισόδου.

Απάντηση-10: Ναι, μια συνάρτηση μπορεί να μην έχει κανένα όρισμα εξόδου.

Απάντηση-11: Συμβουλευτείτε την ενότητα 2.7. του κεφαλαίου.

Απάντηση-12: Συμβουλευτείτε την ενότητα 2.7. του κεφαλαίου.

3. ΜΕΤΑΒΛΗΤΕΣ, ΣΤΑΘΕΡΕΣ ΚΑΙ ΤΕΛΕΣΤΕΣ

Σκοπός και επιμέρους στόχοι

Σκοπός του τρίτου κεφαλαίου είναι η εξοικείωση των εκπαιδευομένων στη χρήση των μεταβλητών, των σταθερών και των τελεστών, μέσα από την γλώσσα προγραμματισμού, C++. Στο κεφάλαιο παρουσιάζονται πολλά εκτεταμένα παραδείγματα και επεξηγούνται με λεπτομέρεια τα σημαντικά σημεία για την προτεραιότητα και την προσεταιριστικότητα των τελεστών στην σύνταξη σχετικών εντολών.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του τρίτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να αναφέρουν τουλάχιστον 3 διαφορετικούς τύπους δεδομένων που μπορεί να υποστηρίξει η γλώσσα C++,
- να αναφέρουν τουλάχιστον 3 διαφορετικούς τροποποιητές, τύπων δεδομένων που μπορεί να υποστηρίξει η γλώσσα C++,
- να επεκτείνουν τις ιδιότητες μιας μεταβλητής, ορίζοντας επάνω σε αυτήν τροποποιητές,
- να αντιλαμβάνονται τον λόγο που θα πρέπει να αρχικοποιούν τις μεταβλητές τους,
- να περιγράφουν την διαφορά ανάμεσα στις μεταβλητές και τις σταθερές στην γλώσσα C++,
- να αντιλαμβάνονται τι σημαίνει η προτεραιότητα των τελεστών στην γλώσσα προγραμματισμού C++,
- να περιγράψουν με απλά λόγια την έννοια της προσεταιριστικότητας για την γλώσσα προγραμματισμού C++,
- να χρησιμοποιούν τους τελεστές για να δημιουργούν κατάλληλες εντολές C++,
- να δηλώσουν σε μια εντολή C++, ειδικούς χαρακτήρες διαφυγής,
- να καταστρώνουν εντολές C++, που εκτελούν αριθμητικούς υπολογισμούς και λογικές πράξεις,

- να αντιλαμβάνονται την σημασία των λογικών εκφράσεων στην γλώσσα προγραμματισμού C++.

Έννοιες – Λέξεις Κλειδιά

Θεμελιώδεις τύποι μεταβλητών, Εμβέλεια μεταβλητής, Προσδιοριστές, Αρχικοποίηση μεταβλητής, Καθοριστές τύπου, Δήλωση τροποποιητή, Τροποποιητές τύπου, Εύρος τιμών, bit και Byte, Λογικός διακόπτης, Αλφαριθμητικά, Πίνακας κωδικοποίησης ASCII, Μεταβλητές, Σταθερές, Ακολουθίες διαφυγής, Τελεστές, Τελεστέοι, Προτεραιότητες τελεστών, Προσεταιριστικότητα, Επίπεδα προτεραιότητας.

Εισαγωγικές Παρατηρήσεις

Σε κάθε διαδικασιακή γλώσσα προγραμματισμού υψηλού επιπέδου, το θέμα της διαχείρισης των διαθέσιμων δεδομένων αποτελεί ιδιαίτερα σημαντικό στοιχείο καθώς μέσα από τα δεδομένα, ένα πρόγραμμα υπολογιστή αποκτά δυναμική συμπεριφορά. Οι γλώσσες προγραμματισμού χρησιμοποιούν τις έννοιες, των μεταβλητών και των τελεστών, για να χειρίζονται τα δεδομένα και με βάση το αρχικό τους περιεχόμενο ή το αποτέλεσμα από μια πράξη ή μια σύγκριση του περιεχομένου τους, να διαμορφώνουν ανάλογα την ροή εκτέλεσης του προγράμματος. Η γλώσσα προγραμματισμού C++, διαθέτει ένα μεγάλο σύνολο παραμέτρων για την διαμόρφωση των δεδομένων. Στο υλικό του συγκεκριμένου κεφαλαίου παρουσιάζεται ένα μέρος αυτών των δυνατοτήτων. Η γνώση των διαθέσιμων επιλογών που περιγράφονται, θα πρέπει να θεωρείται αναγκαία γνώση για τους εκπαιδευόμενους και θα διευκολύνει στην συνέχεια την κατασκευή λειτουργικών και αποδοτικών προγραμμάτων.

3.1. Οι Μεταβλητές

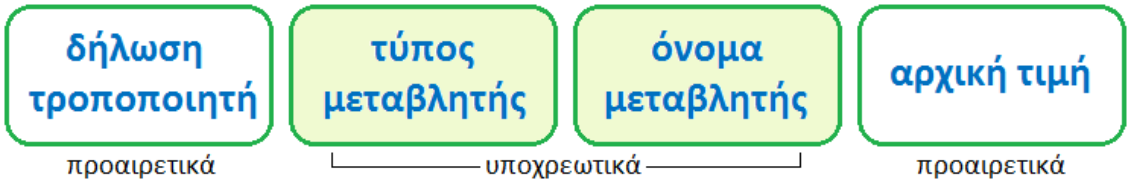
Οι τύποι των δεδομένων και κατά συνέπεια, οι τύποι των μεταβλητών που μπορεί να υποστηρίξει η γλώσσα προγραμματισμού C++, είναι πολλοί και διαφορετικοί. Επιπρόσθετα, παρέχονται ειδικές δηλώσεις με τις οποίες διαφοροποιείται η τυπική συμπεριφορά κάθε μεταβλητής, ενώ είναι δυνατό να κατασκευαστούν, ακόμα και νέοι τύποι μεταβλητών. Όπως

και σε κάθε άλλη γλώσσα, στην C++, υποστηρίζονται κυρίως, οι ακέραιοι αριθμοί, οι πραγματικοί αριθμοί, οι χαρακτήρες και ο λογικός τύπος.

Θεμελιώδεις Τύποι Μεταβλητών	
char	χαρακτήρας
int	ακέραιος χωρίς δεκαδικά
float	πραγματικός, κινητής υποδιαστολής, με δεκαδικά
double	πραγματικός, κινητής υποδιαστολής, διπλής ακριβείας, με δεκαδικά
bool	λογικός διακόπτης true/false

Με την επιλογή του τύπου μιας μεταβλητής ο προγραμματιστής δηλώνει, την ποσότητα μνήμης που θα δεσμεύσει από τον υπολογιστή, το όριο των δεδομένων που θέλει να δέχεται η μεταβλητή και τις ενέργειες στις οποίες θα μπορεί αυτή, να συμμετέχει. Ένα πρόσθετο σημαντικό στοιχείο που μπορεί επίσης να καθοριστεί, είναι η εμβέλεια του νέου στοιχείου. Ο όρος, *εμβέλεια μεταβλητής*, στον προγραμματισμό σημαίνει ότι είναι δυνατό να καθοριστεί εάν μια μεταβλητή θα είναι αποδεκτή σε ολόκληρο το πρόγραμμα ή μόνο μέσα σε μια συγκεκριμένη περιοχή στα όρια μιας συνάρτησης. Στην περίπτωση που η μεταβλητή ορίζεται εντός των ορίων μιας συγκεκριμένης συνάρτησης ονομάζεται *τοπική μεταβλητή* (local) και είναι αναγνωρίσιμη μόνο μέσα σε αυτή την συνάρτηση, ενώ όταν ορίζεται στην αρχή του προγράμματος και έξω από την κεντρική συνάρτηση **main()**, μπορεί να είναι αναγνωρίσιμη σε ολόκληρο το πρόγραμμα και τότε θα ονομάζεται *καθολική μεταβλητή* ή *εξωτερική μεταβλητή* (external). Για να ορίσουμε μια νέα μεταβλητή θα πρέπει υποχρεωτικά να δηλώσουμε αρχικά τον τύπο και το όνομα της. Ακολούθως και προαιρετικά, μπορούμε να δηλώσουμε τροποποίηση της βασικής της συμπεριφοράς (με τους τροποποιητές ή προσδιοριστές) και να της δώσουμε και αρχική τιμή. Η δήλωση μιας μεταβλητής μπορεί να γίνει σε οποιοδήποτε σημείο αυτό κριθεί αναγκαίο μέσα στο πρόγραμμα. Είναι όμως συνηθισμένη τακτική οι δηλώσεις των μεταβλητών να τοποθετούνται όλες μαζί σε ένα συγκεκριμένο σημείο ώστε να είναι στη συνέχεια εύκολη η αναζήτηση και ο έλεγχος τους.

Η γενική σύνταξη των δηλώσεων μεταβλητών, είναι,



Για παράδειγμα, η εντολή,

```
int  w_mia_metavliti  =  0 ;
```

δημιουργεί μια νέα μεταβλητή με το όνομα **w_mia_metavliti**, η οποία θα είναι κατάλληλη για τον χειρισμό ακεραίων αριθμών (εξαιτίας της δήλωσης **int**) και θα αναλάβει για αρχικό περιεχόμενο την τιμή **0** (εξαιτίας της δήλωσης **= 0**). Η δήλωση του τύπου **int** ανήκει στους θεμελιώδεις τύπους μεταβλητών της γλώσσας και αφορά τους ακέραιους αριθμούς. Οι μεταβλητές αυτού του τύπου μπορούν να γεμίσουν με ακέραιες ποσότητες χωρίς κλασματικό μέρος. Ο καθορισμός ολοκληρώνεται και με το απαραίτητο ελληνικό ερωτηματικό **;** στο τέλος της εντολής. Κατά την δήλωση κάθε νέας μεταβλητής εκτός των περιγραφικών ονομάτων, είναι καλή πρακτική, να δίνεται πάντα και αρχική τιμή εκκαθάρισης, ώστε να είναι σίγουρο το περιεχόμενο της. Στα πλαίσια των κανόνων του δομημένου προγραμματισμού αυτό ονομάζεται, *αρχικοποίηση μεταβλητής*. Η αρχικοποίηση στο παράδειγμα, γίνεται με την δήλωση **= 0**, η οποία ζητά από τον compiler να αναθέσει στην μεταβλητή, την τιμή **0.** Ονομάζεται και *δήλωση εκχώρησης*. Η χρήση του χαρακτήρα **=** δεν υποδηλώνει την ισότητα στα μαθηματικά, αλλά την έννοια της ανάθεσης (την τοποθέτηση ή την εκχώρηση) της αξίας **0**, μέσα στην μεταβλητή που βρίσκεται πριν από το **=**. Αυτή η ειδοποιός διαφορά, συνηθίζεται σε όλες τις ομοιάζουσες γλώσσες με την C. Η εντολή αρχικοποίησης θα μπορούσε ισοδύναμα να γίνει με την δήλωση **int w_mia_metavliti = { 0 }** ή με την δήλωση **int w_mia_metavliti {0}**. Δηλώνοντας έναν τύπο μεταβλητής, έχουμε την δυνατότητα να ορίσουμε περισσότερες από μια μεταβλητές, με την διαμόρφωση **int w_ianouarios=31, w_fevrouarios=28, w_martios=31 ;**, προσέχοντας να διαχωρίζουμε τις μεταβλητές με ένα κόμμα. Εκτός από την γενική σύνταξη των δηλώσεων μεταβλητών, η γλώσσα διαθέτει και πολλούς άλλους πρόσθετους δηλωτικούς κανόνες, οι οποίοι θα περιγραφούν στην συνέχεια.

3.2. Τύποι Μεταβλητών - Τύποι Δεδομένων

Όπως έχει ήδη αναφερθεί η C++, υποστηρίζει για θεμελιώδεις τύπους μεταβλητών τους ακέραιους αριθμούς, τους πραγματικούς αριθμούς, τους χαρακτήρες και τον λογικό τύπο. Η επιλογή του κάθε *τύπου μεταβλητής*, γίνεται με την κατάλληλη δεσμευμένη λέξη. Οι τύποι των μεταβλητών ονομάζονται και *καθοριστές τύπου* (type specifiers). Επιπλέον, η γλώσσα διαθέτει και δυνατότητες ανα-προσδιορισμού των ιδιοτήτων για κάθε τύπο μεταβλητής με την *δήλωση τροποποιητή*. Αυτό σημαίνει ότι υπάρχουν κάποιες πρόσθετες λέξεις κλειδιά, οι οποίες, εάν δηλωθούν δίπλα (μπροστά) από τον τύπο της μεταβλητής, μπορούν να διαφοροποιήσουν (να εξειδικεύσουν), την «κλασσική» συμπεριφορά που παρέχει ο τύπος των δεδομένων, της μεταβλητής στην οποία αναφέρονται. Αυτές οι δηλώσεις ανα-προσδιορισμού, ονομάζονται *προσδιοριστές τύπου* ή *τροποποιητές τύπου* (type modifiers) και είναι προαιρετικές δηλώσεις. Οι προσδιοριστές μπορούν να αφορούν μονή, διπλή ή και τριπλή δήλωση (π.χ. **unsigned long long**). Κάποιοι από τους προσδιοριστές μπορούν να συνδυαστούν μόνο με συγκεκριμένους τύπους μεταβλητών. Ειδικά για τις αριθμητικές μεταβλητές ο τροποποιητής **signed** θεωρείται μη αναγκαίος γιατί ακόμα και όταν δεν εμφανίζεται, υπονοείται η ύπαρξη του, καθώς η δήλωση ενός αριθμού, υπονοεί ότι θα έχει πρόσημο. Στις νεότερες εκδόσεις της C++, προστέθηκε ο ειδικός τύπος προσδιοριστή **auto**, ο οποίος με βάση το περιεχόμενο που αναθέτουμε (τοποθετούμε) στην μεταβλητή, μπορεί η γλώσσα αυτόματα να διαμορφώνει τον τύπο της.

Τροποποιητές Τύπου
long – μεγάλος
unsigned – χωρίς πρόσημο
signed – με πρόσημο
short – μικρός
register – αυτόματος προσδιορισμός
static – αυτόματος προσδιορισμός

Τύποι Μεταβλητών
int – ακέραιος, χωρίς δεκαδικά
float – πραγματικός, κινητής υποδιαστολής, με δεκαδικά
double – πραγματικός, κινητής υποδιαστολής, διπλής ακριβείας, με δεκαδικά

extern – αυτόμ.εξωτ.προσδιορισμός

auto – αυτόματος προσδιορισμός

void – χωρίς επιστροφή

char – χαρακτήρας

bool – λογικός διακόπτης, true/false

char8_t

char16_t

char32_t

enum – με απαρίθμηση των επιτρεπτών τιμών

Στον πίνακα που ακολουθεί παρουσιάζονται όλοι οι πιθανοί συνδυασμοί, των τύπων με τους τροποποιητές καθώς και το εύρος των τιμών που μπορεί καθένας από αυτούς να αποδώσει. Είναι εύκολα κατανοητό ότι ο βασικός ρόλος των τροποποιητών είναι η δέσμευση μεγαλύτερου χώρου μνήμης. Στον πίνακα, εμπεριέχονται δυο πρόσθετες στήλες με την ποσότητα μνήμης που δεσμεύει η κάθε σύνθεση. Οι στήλες αναφέρονται στις δυο βασικές έννοιες (το bit και το Byte) με τις οποίες αναπαριστάται στους ηλεκτρονικούς υπολογιστές η ποσότητα της μνήμης. Το bit είναι η μικρότερη (η στοιχειώδης) μονάδα με την οποία μετράμε την μνήμη που δεσμεύουμε. Μπορούμε να φανταστούμε το bit σαν μια κατασκευή, έναν «μικρό ηλεκτρικό διακόπτη» ο οποίος μπορεί να πάρει δυο τιμές, το ένα και το μηδέν (ή καλύτερα, «το έχει ρεύμα» ή «το δεν έχει ρεύμα»). Μια κλίμακα μεγαλύτερη αναπαράστασης της μνήμης είναι το Byte, το οποίο αποτελείται από 8 bit. Όλες οι τεχνολογίες και όλες οι τεχνικές στην επιστήμη της *Πληροφορικής*, βασίζονται σε αυτές τις δυο θεμελιώδεις μονάδες μέτρησης της μνήμης, το bit και το Byte.

Όλες οι γλώσσες προγραμματισμού υποστηρίζουν μια σχετική προ-οικονομία, καθώς αυτό βελτιώνει την χωρική πολυπλοκότητα των αλγορίθμων που κατασκευάζονται. Για ποιον λόγο

να χρησιμοποιηθεί μια μεταβλητή με τύπο, **unsigned long long int w_imeres_ergasias** ο οποίος τύπος έχει την ικανότητα να χειριστεί περιεχόμενο μέχρι τον αριθμό 18.446.744.073.709.551.615 (18 πεντάκις εκατομμύρια…!), για να χρησιμοποιηθεί μέσα σε ένα πρόγραμμα, από την μεταβλητή **w_imeres_ergasias** η οποία θα καταμετρά τις εργάσιμες ημέρες ενός εργαζόμενου ανά έτος. Προφανώς, η απλή δήλωση, **int w_imeres_ergasias** είναι ικανή από μόνης της να καλύψει τις ανάγκες της συγκεκριμένης μεταβλητής, που σε καμία περίπτωση δεν θα ξεπεράσει τον αριθμό 365. Δεν χρειάζεται δηλαδή, να δεσμευτούν 64 bit (ή 8 Byte) αφού αρκεί να δεσμευτούν μόνο 16 bit (ή 2 Byte).

Τροποποιητές	Τύπος	Byte	bit	Εύρος τιμών που μπορεί να λάβει
-	char	1	8	-128 έως 127
unsigned	char	1	8	0 έως 255
signed	char	1	8	-128 έως 127
-	int	2	16	-32768 έως 32767
unsigned	int	2	16	0 έως 65535
signed	int	2	16	-32768 έως 32767
short	int	2	16	-32768 έως 32767
unsigned short	int	2	16	0 έως 65535
signed short	int	2	16	-32768 έως 32767
long	int	4	32	-2147483648 έως 2147483647
signed long	int	4	32	-2147483648 έως 2147483647
unsigned long	int	4	32	0 έως 4294967295
unsigned long long	int	8	64	0 έως 18446744073709551615
-	float	4	32	$3.4*10^{-38}$ έως $3.4*10^{38}$
-	double	8	64	$1.7*10^{-308}$ έως $1.7*10^{308}$
long	double	10	80	$3.4*10^{-4932}$ έως $1.1*10^{4932}$
long long	int	8	64	-9223372036854775808 έως 9223372036854775807
-	bool		1	0 έως 1 ή false και true

Στα πλαίσια των κανόνων του δομημένου προγραμματισμού, αυτό ονομάζεται, *προ-οικονομία* και καθορίζει την *χωρική πολυπλοκότητα* ενός προγράμματος αλλά και σαν συνέπεια και την *χρονική του πολυπλοκότητα*. Ένα μικρότερο πρόγραμμα χρειάζεται λιγότερο χρόνο για να εκτελεστεί.

Ο τύπος **bool** συμπεριφέρεται σαν ακέραιος αριθμός λαμβάνοντας τις τιμές, 0 και 1. Το 0 σημαίνει «όχι» και το 1 σημαίνει «ναι». Χρησιμοποιείται μέσα στα προγράμματα σαν ένας

«λογικός διακόπτης», ο οποίος μπορεί να ελεγχθεί για την κατάσταση στην οποία βρίσκεται με τις λέξεις **false** και **true**. Εάν δηλαδή μια μεταβλητή τύπου **bool** είναι **true**, ο compiler μπορεί να οδηγηθεί να εκτελέσει αυτή την συνάρτηση, διαφορετικά να εκτελέσει μιαν άλλη συνάρτηση. Οι μεταβλητές τύπου **bool** λειτουργούν σαν «σηματοδότες» («διακόπτες») επιτρέποντας την ροή της «μπίλιας» από ένα σημείο κώδικα ή όχι, ανάλογα με το εάν ο «σηματοδότης» είναι αναμμένος ή σβηστός. Τα «νήματα» μέσα στα προγράμματα δημιουργούνται από τους «σηματοδότες». Οι μεταβλητές «σηματοδότες», είναι ιδιαίτερα συνηθισμένη τακτική των προγραμματιστών για όλες τις γλώσσες προγραμματισμού.

Ο τύπος **char** ανήκει στους θεμελιώδεις τύπους της γλώσσας και ο χώρος μνήμης που καταλαμβάνει είναι 8 bit ή 1 Byte. Τα δεδομένα αυτού του τύπου ονομάζονται και *αλφαριθμητικά*. Το περιεχόμενο που θα υποδέχεται μια τύπου **char** μεταβλητή είναι ένα και μόνο ένα από τα στοιχεία που αποτελούν το αλφάβητο της γλώσσας. Παρόλο που ο τύπος **char** χρησιμοποιείται για να αποτυπώνει αλφαβητικούς χαρακτήρες (οποιονδήποτε χαρακτήρα από το αλφάβητο) στην πραγματικότητα ο τύπος **char** συμπεριφέρεται επίσης σαν ακέραιος αριθμός λαμβάνοντας τιμές, από το -128 έως το 127. Εάν δηλαδή, σε μια μεταβλητή τύπου **char**, θελήσουμε να τοποθετήσουμε σαν περιεχόμενο το γράμμα **W**, αυτό που θα τοποθετηθεί από την γλώσσα μέσα στην μεταβλητή θα είναι ο αριθμός 87 και όχι το ίδιο το γράμμα **W**. Αντίστοιχα, εάν θελήσουμε να τοποθετήσουμε σαν περιεχόμενο το ελληνικό κεφαλαίο γράμμα **Δ**, αυτό που θα τοποθετηθεί από την γλώσσα μέσα στην μεταβλητή θα είναι ο αριθμός 196. Ακόμα και στην περίπτωση που θελήσουμε να τοποθετήσουμε σαν περιεχόμενο τον αριθμό **3**, το πραγματικό περιεχόμενο που θα τοποθετηθεί από την γλώσσα μέσα στην μεταβλητή θα είναι ο αριθμός 51 και όχι ο αριθμός 3. Η γλώσσα στην πραγματικότητα εκτελεί πάντα έναν μετασχηματισμό χρησιμοποιώντας έναν τυποποιημένο πίνακα κωδικοποίησης και αποκωδικοποίησης που ονομάζεται ASCII. Ο πίνακας αυτός διαθέτει 255 αριθμούς και κάθε ένας από αυτούς τους αριθμούς έχει αντιστοιχηθεί από τον οργανισμό προτύπων ASCII, με ένα από τα γράμματα του αλφαβήτου. Παρόλα αυτά οι προγραμματιστές δεν χρειάζεται να ασχολούνται με την μετατροπή καθώς κατά την ανάγνωση του περιεχομένου της μεταβλητής, η γλώσσα έχει τους κατάλληλους μηχανισμούς για να εκτελεί αυτόματα την μετατροπή και ο προγραμματιστής να βλέπει τελικά τα **W**, **Δ** και **3**. Πρέπει επίσης να σημειωθεί, χωρίς όμως να υπεισέλθουμε σε λεπτομέρειες, ότι οι

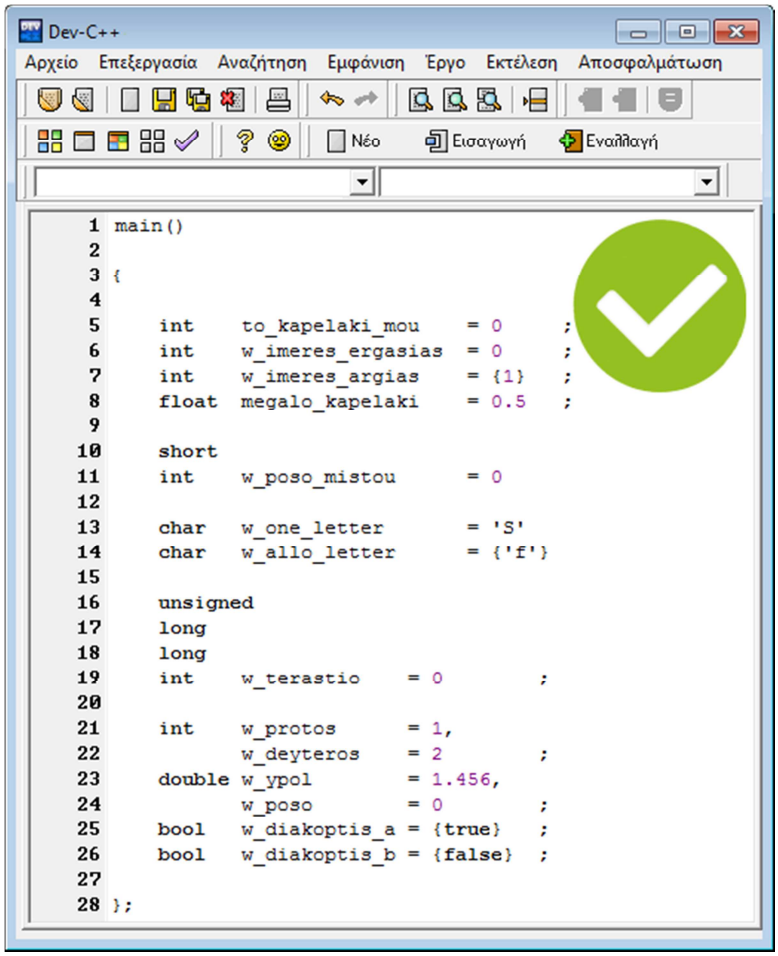
θεμελιώδεις μεταβλητές τύπου **char** μπορούν να αποθηκεύσουν ένα και μόνο ένα στοιχείο του αλφαβήτου. Εάν δηλαδή θελήσουμε να αποθηκεύσουμε το όνομα **ΑΘΗΝΑ** δεν αρκεί μια και μόνο, μεταβλητή τύπου **char**, αφού η μεταβλητή θα κρατήσει μόνο έναν χαρακτήρα από το **ΑΘΗΝΑ**. Θα πρέπει συνεπώς να δημιουργηθούν 5 διαφορετικές μεταβλητές τύπου **char**. Ευτυχώς όμως για τους προγραμματιστές η γλώσσα διαθέτει ισχυρότερες ικανότητες για την υποστήριξη αυτών των περιπτώσεων και την διαχείριση σύνθετων δομών δεδομένων, οι οποίες θα περιγραφούν στην συνέχεια. Ο χειρισμός του τύπου **char** των μεταβλητών υποχρεώνει τους προγραμματιστές να τοποθετούν τα περιεχόμενα ανάμεσα σε διπλά εισαγωγικά, το **ΑΘΗΝΑ** δηλαδή πρέπει να γραφεί σαν **“ΑΘΗΝΑ”**, το **W** πρέπει να γραφεί σαν **“W”** και το **3** πρέπει να γραφεί σαν **“3”**.

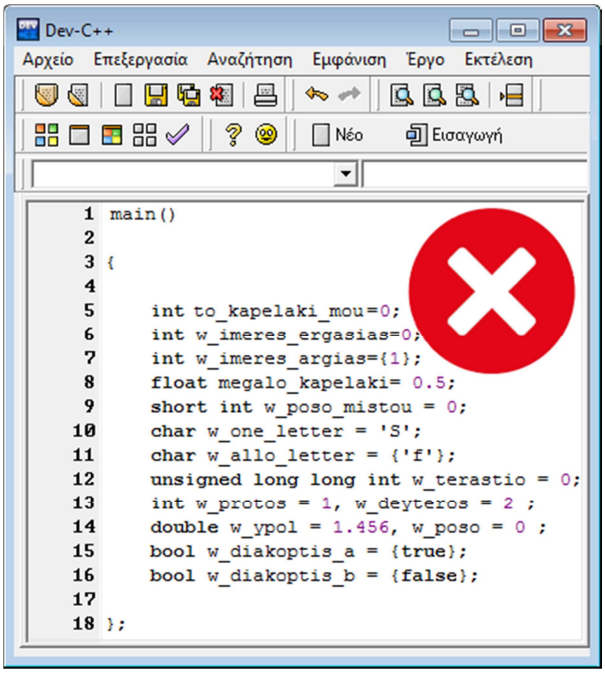
Για την αποτύπωση αριθμών με κλασματικό μέρος, (με δεκαδικά ψηφία), η γλώσσα διαθέτει τους τύπους μεταβλητών **float** και **double**. Οι μεταβλητές τύπου **double** χρησιμοποιούνται όταν απαιτείται αποτύπωση πολύ μεγάλων αριθμών. Οι μεταβλητές τύπου **float** και **double**, θεωρούνται πάντα προσημασμένοι και κατά συνέπεια δεν μπορούν να πάρουν του τροποποιητές **unsigned** και **signed**.

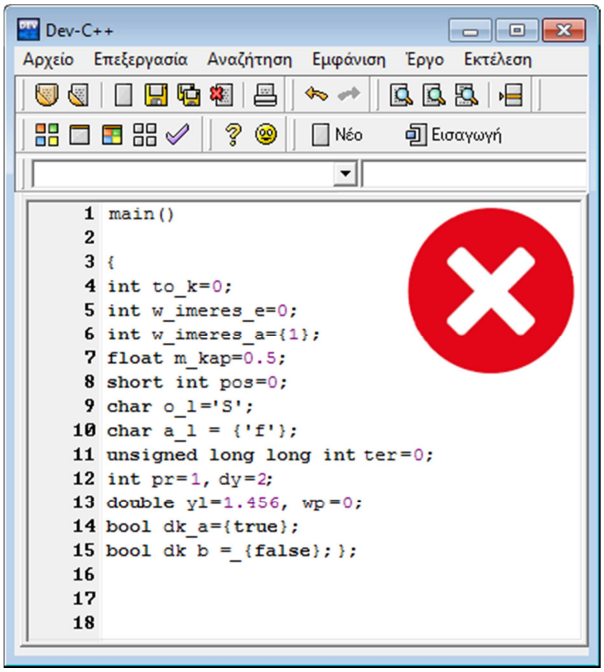
Στην βάση του κατά το ISO 2020 πρότυπου της C++, υπάρχουν συντομογραφίες για πολλές από τις δηλώσεις των τύπων και των τροποποιητών της γλώσσας, όπως οι, **u**, **U**, **l**, **L**, **ll** και **LL**, συστήνεται όμως η χρήση των κανονικών περιγραφικών δηλώσεων του λεξιλογίου.

Στον κώδικα που ακολουθεί παρουσιάζονται συνδυασμοί, δηλώσεων τύπων μεταβλητών με κανόνες δομημένου προγραμματισμού. Ειδικότερα, οι μεταβλητές δηλώνονται με περιγραφικά ονόματα και επίσης εφαρμόζονται κανόνες «κομψότητας», με τις εντολές να στηλοθετούνται με τρόπο ώστε να διευκολύνουν την ανάγνωση και την κατανόηση. Στα οργανωμένα software house (τις εταιρίες ανάπτυξης λογισμικού), οι κώδικες των προγραμματιστών, που δεν πληρούν αυτούς τους βασικούς κανόνες, δεν γίνονται αποδεκτοί. Για να γίνει αυτό κατανοητό, ο κώδικας παρουσιάζεται ακολούθως, χωρίς την στηλοθέτηση των εντολών (αριστερός κώδικας) και επιπλέον χωρίς περιγραφικά ονόματα (δεξιός κώδικας).

Καλούνται οι εκπαιδευόμενοι να παρατηρήσουν αυτές τις τρεις εικόνες και στην βάση της έκφρασης, «μια εικόνα είναι χίλιες λέξεις» να υιοθετήσουν συμπεριφορές συγγραφής προγραμμάτων με «κομψότητα». Αυτή η κουλτούρα, «κομψότητας» στα προγράμματα, ονομάζεται, *δομημένος προγραμματισμός.*







3.3. Οι Σταθερές (με ονομασία)

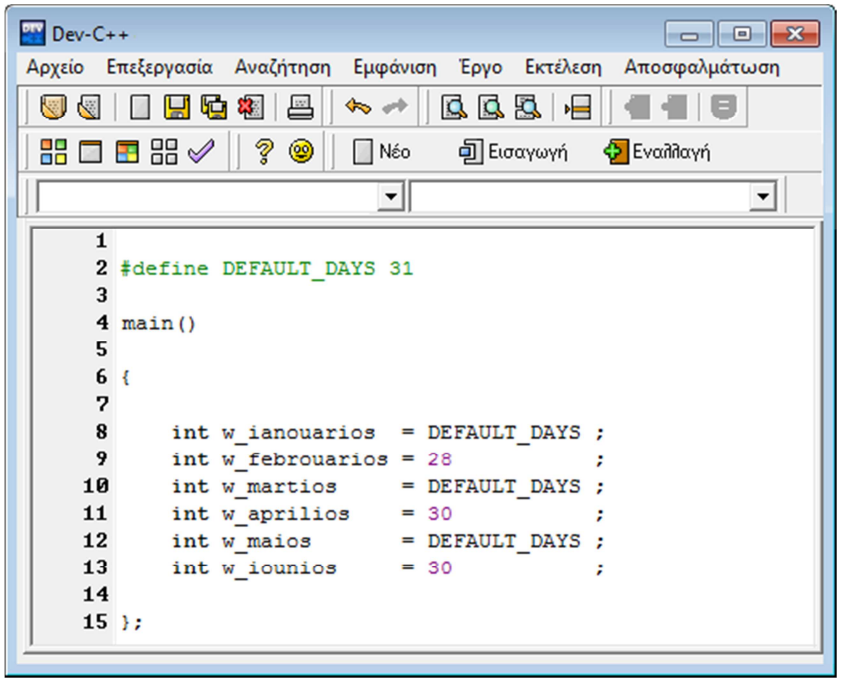
Ολοκληρώνοντας τη μελέτη καθορισμού των μεταβλητών πρέπει να περιγραφεί και ένας ειδικός τύπος μεταβλητών που ονομάζονται *σταθερές* (constants). Ο σκοπός ύπαρξης των σταθερών, είναι να λάβουν μια αρχική τιμή (π.χ. το 3.14, την λέξη «ΝΑΙ», κ.λπ.) και να την διατηρήσουν σε όλη την διάρκεια της εκτέλεσης ενός προγράμματος. Η δήλωση μιας σταθεράς γίνεται με την «εργοστασιακή» λέξη **define**. Η εντολή **define** (όρισε) συντάσσεται μαζί με το σύμβολο **#**, το οποίο προηγείται από την καθαυτή εντολή και έτσι δημιουργεί μια οδηγία «ντιρεκτίβα» προς τον μεταφραστή (compiler) για να ενεργοποιήσει τον pre-processor, να εκτελέσει μια ενέργεια αντικατάστασης. Οι δηλώσεις **define**, τοποθετούνται στην αρχή κάθε προγράμματος και η γενική τους σύνταξη είναι,

#define **όνομα_σταθεράς** **τιμή_σταθεράς**

Για παράδειγμα, η εντολή **#define ISOTIMIA-EURO-DRAXMH 340.74**, στην πρώτη γραμμή ενός προγράμματος C++, θα προκαλέσει τον pre-processor να αντικαταστήσει μέσα στο πρόγραμμα οποιαδήποτε εμφάνιση της έκφρασης **ISOTIMIA-EURO-DRAXMH**, με την τιμή **340.74**. Εάν αναρωτηθούμε γιατί το κάνουμε αυτό; και δεν γράφουμε κατ' ευθείαν το **340.74** εκεί που το χρειαζόμαστε, η απάντηση είναι απλή, γιατί το **ISOTIMIA-EURO-DRAXMH** μπορεί να

παριστάνει την δήλωση **340.74** που ενδεχομένως να χρησιμοποιείται σε πάρα πολλά σημεία μέσα στο πρόγραμμα και άρα θα είναι δυσκολότερη η εύρεση και η αντικατάσταση του με μια νεότερη ισοτιμία. Ενώ, εάν το κάνουμε constant, αρκεί μια μοναδική αλλαγή στην αρχή του προγράμματος και ο pre-processor θα αναλάβει τα υπόλοιπα.

Η γλώσσα προσφέρει την δυνατότητα αναίρεσης μιας σταθεράς μέσα σε ένα πρόγραμμα με την εντολή «ντιρεκτίβα», **#undef ISOTIMIA-EURO**. Οι σταθερές όταν είναι γνωστό ότι δεν θα αλλάξουν τιμή είναι καλό να δηλώνονται σαν constant, καθώς ο compiler βελτιώνει την ταχύτητα του παραγόμενου κώδικα.



3.4. Οι Απλές Σταθερές και οι Ειδικοί Χαρακτήρες Διαφυγής

Οι βασικοί χαρακτήρες του λεξιλογίου της γλώσσας (τα γράμματα και οι αριθμοί), μπορούν να θεωρηθούν ότι είναι απλές *στοιχειώδεις σταθερές.* Το νούμερο **3**, μπορεί να είναι μια απλή *ακέραια σταθερά* για ένα πρόγραμμα, ενώ το ίδιο μπορεί να συμβεί και για το γράμμα **E**, το οποίο εάν περικλείεται από διπλά εισαγωγικά, μπορεί να είναι μια απλή *αλφαριθμητική σταθερά.* Αυτές οι στοιχειώσεις σταθερές, όταν είναι αριθμοί μπορούν να χρησιμοποιηθούν σε υπολογισμούς και συγκρίσεις, ενώ όταν είναι γράμματα μπορούν να χρησιμοποιηθούν για επεξεργασίες αλφαριθμητικού περιεχομένου. Απλή *ακέραια σταθερά* είναι και ο ακέραιος αριθμός **1024**, ενώ ο πραγματικός αριθμός **3.14**, είναι μια απλή *πραγματική σταθερά*. Το ίδιο

και η λέξη **ΕΛΛΑΣ**, η οποία εάν περικλείεται από διπλά εισαγωγικά, είναι μια απλή *αλφαριθμητική σταθερά.* Εκτός αυτών των σταθερών οι γλώσσες C, διαθέτουν και μια πρόσθετη κατηγορία ειδικών σταθερών που ονομάζονται, *χαρακτήρες διαφυγής* ή *ακολουθίες διαφυγής* ή *σταθερές αναστροφής* ή *σταθερές ανάποδης καθέτου* (escape sequence). Η χρησιμότητα τους είναι σημαντική και πρέπει να περιγραφεί, καθώς χρησιμοποιούνται για να προκαλέσουν ειδικές ενέργειες, όπως για παράδειγμα, να προκαλέσουν έναν ήχο κουδουνίσματος από το πληκτρολόγιο, να προκαλέσουν αλλαγή γραμμής στον εκτυπωτή, αλλαγή σελίδα, κ.λπ. Η χρήση του χαρακτήρα της ανάποδου καθέτου ****, προειδοποιεί τον compiler ότι αυτό που ακολουθεί είναι κάτι πολύ ειδικότερο από ένας απλός χαρακτήρας, που έχει συμφωνηθεί να προκαλεί έναν ιδιαίτερο μηχανισμό. Αυτή είναι και η σημασία της έκφρασης, χαρακτήρας διαφυγής, ότι περιγράφει έναν χαρακτήρα που διαφεύγει από τα συνηθισμένα και αλλάξει την ερμηνεία του. Η γλώσσα C++, διαθέτει πολλούς *χαρακτήρες διαφυγής.*

Βασικές Ακολουθίες Διαφυγής - Escape sequences	
\'	να τοποθετηθεί στο κείμενο ένα μονό εισαγωγικό
\"	να τοποθετηθεί στο κείμενο ένα διπλό εισαγωγικό
\?	να τοποθετηθεί στο κείμενο ένα ερωτηματικό
****	να τοποθετηθεί στο κείμενο μια ανάποδη κάθετος
\a	να προκληθεί ένας ήχος κουδουνίσματος
\b	να διαγραφεί ο προηγούμενος χαρακτήρας
\f	να γίνει αλλαγή σελίδας
\n	να γίνει αλλαγή γραμμής
\r	να μετακινηθεί ο δείκτης θέσης στην αρχή της γραμμής
\t	να εκτελεστεί ένα οριζόντιο TAB (στηλοθέτηση)
\v	να εκτελεστεί ένα κάθετο TAB (στηλοθέτηση)
\0	το μηδέν, είναι ένας χρήσιμος χαρακτήρας που χρησιμοποιείται στον χειρισμό αλφαριθμητικού περιεχομένου, (σηματοδοτεί το τέλος του περιεχομένου ενός αλφαριθμητικού πεδίου)

3.5. Οι Τελεστές

Αυτό το στάδιο μελέτης, του συγγράμματος αφορά την εφαρμογή κατάλληλων μεθόδων για την εκμετάλλευση των μεταβλητών. Οι μεταβλητές στις γλώσσες προγραμματισμού αποκτούν ουσία ύπαρξης όταν συμμετέχουν σε πράξεις. Οι πράξεις στον προγραμματισμό μπορεί, να είναι αριθμητικές, λογικές ή ενέργειες διαμόρφωσης περιεχομένου. Εφόσον μια πράξη

χρειάζεται μεταβλητές για να εκτελεστεί θα πρέπει επίσης και με έναν κατάλληλο τρόπο να καθορίζεται το είδος της πράξης που θα εκτελεστεί. Την υπόδειξη της πράξης, όπως και στα μαθηματικά, την υποδεικνύουν τα σύμβολα που ονομάζονται *τελεστές* (operators). Οι ίδιες οι μεταβλητές στις πράξεις ονομάζονται *τελεστέοι* (operands). Στην C++, η γνώση της σωστής τοποθέτησης των τελεστών έχει πολύ μεγάλη σημασία καθώς καθορίζουν τους τρόπους με τους οποίους εκτελούνται οι πράξεις. Η δήλωση ενός τελεστή σε κάποιο σημείο μιας πράξης, δεν σημαίνει ότι αυτό θα γίνεται αποκλειστικά και μόνο με την παρουσία ενός ειδικού χαρακτήρα. Πολλές ενέργειες τελεστών υποδηλώνονται στον compiler με την χρήση δυο ή και τριών ειδικών χαρακτήρων. Οι τελεστές στην C++, διαχωρίζονται αρχικά με βάση το πώς τοποθετούνται σε σχέση με τους τελεστέους. Υπάρχουν τρεις διαφορετικές περιπτώσεις τοποθέτησης: α) ανάμεσα σε δυο τελεστέους, β) πριν από έναν τελεστέο και γ) μετά από έναν τελεστέο. Ειδικότερα, η C++, έχει επτά γενικές κατηγορίες τελεστών. Πολλοί τελεστές από μια κατηγορία, μπορούν να συνδυαστούν με τελεστές από μια άλλη κατηγορία. Στα όρια των εκπαιδευτικών στόχων του συγγράμματος αυτού, παρουσιάζονται οι τέσσερεις βασικότερες κατηγορίες.

Οι αριθμητικοί τελεστές χρησιμοποιούνται κυρίως για να εκτελούν τις βασικές αριθμητικές πράξεις και συνδυάζονται συνήθως με κατάλληλους τελεστές ανάθεσης.

Αριθμητικοί Τελεστές - Arithmetic Operators	
-	αφαίρεση
-	αρνητικό πρόσημο (αν τοποθετηθεί μπροστά από αριθμό, θα κάνει αλλαγή στο πρόσημο του)
+	πρόσθεση
*****	πολλαπλασιασμός
/	διαίρεση (αποκόπτει το υπόλοιπο της διαίρεσης)
%	διαίρεση - ακέραιο υπόλοιπο διαίρεσης (ονομάζεται και *πράξη modulo*, επιστρέφει το υπόλοιπο της διαίρεσης) (αποκόπτει το πηλίκο της διαίρεσης)
--	μείωση κατά ένα (μπορεί να τοποθετηθεί πριν ή μετά την μεταβλητή)
++	αύξηση κατά ένα (μπορεί να τοποθετηθεί πριν ή μετά την μεταβλητή)

Επιπλέον η C++, διαθέτει δυο ιδιαίτερα χρήσιμους ειδικούς αριθμητικούς τελεστές. Τον τελεστή μοναδιαίας αύξησης **++** και τον τελεστή μοναδιαίας μείωσης **--**. Ο τελεστής της μοναδιαίας αύξησης προσθέτει **1** στον τελεστέο, ενώ ο τελεστής μοναδιαίας μείωσης αφαιρεί **1** από τον τελεστέο. Οι τελεστές **++** και **--** μπορούν να μπουν και πριν και μετά από μια μεταβλητή, με την διαφορά ότι όταν ο τελεστής, βρίσκεται πριν από την μεταβλητή, η γλώσσα εκτελεί την πράξη της αύξησης ή της μείωσης πριν χρησιμοποιήσει την τιμή της μεταβλητής. Ενώ, όταν ο τελεστής βρίσκεται μετά από την μεταβλητή, η γλώσσα χρησιμοποιεί την τιμή της μεταβλητής πριν την αυξήσει ή την μειώσει. Για να γίνει αυτό κατανοητό και να αναδειχθεί η δυναμική της σύνθεσης με διαφορετικό τρόπο των τελεστών, ακολουθεί ένα σχετικό παράδειγμα.

Στον κώδικα,

```
my_sum = 0 ;
w_metritis = 12 ;
my_sum = ++w_metritis ;
```

η γλώσσα,

- αναθέτει στην μεταβλητή **w_sum** το **0**,
- αναθέτει στην μεταβλητή **w_metritis** το **12**,
- πρώτα, εκτελεί την πράξη της αύξησης κατά **1**, στην μεταβλητή **w_metritis** που γίνεται **13**, πριν προχωρήσει στην χρησιμοποίηση του περιεχομένου της μεταβλητής,
- αναθέτει (δίνει) το περιεχόμενο της μεταβλητής **w_metritis** στην **my_sum** που γίνεται **13**.

Ενώ στο παράδειγμα,

```
my_sum = 0 ;
w_metritis = 12 ;
my_sum = w_metritis ++ ;
```

η γλώσσα,

- αναθέτει στην μεταβλητή **w_sum** το **0**,
- αναθέτει στην μεταβλητή **w_metritis** το **12**,
- πρώτα, αναθέτει (δίνει) το περιεχόμενο της μεταβλητής **w_metritis** στην **my_sum** που γίνεται **12**,
- εκτελεί την πράξη της αύξησης κατά **1**, στην μεταβλητή **w_metritis** που γίνεται **13**.

Οι συγκριτικοί τελεστές επιστρέφουν μια απάντηση απόφασης, true ή false (αλήθεια ή ψέμα) εκτελώντας μια σύγκριση, ανάμεσα σε δυο εκφράσεις, μια στην αριστερή πλευρά του τελεστή και μια στην δεξιά πλευρά του τελεστή. Η έκφραση μπορεί να είναι μια απλή μεταβλητή ή μια σταθερά. Η απόφαση από τους συγκριτικούς τελεστές, true ή false (αλήθεια ή ψέμα) μπορεί στην συνέχεια να χρησιμοποιηθεί από κατάλληλες αλγοριθμικές δομές επιλογής ή επανάληψης.

Συγκριτικοί Τελεστές - Relational Operators	
<	μικρότερο (θέτει την ερώτηση: είναι η αριστερή έκφραση μικρότερη από την δεξιά)
>	μεγαλύτερο (θέτει την ερώτηση: είναι η αριστερή έκφραση μεγαλύτερη από την δεξιά)
<=	μικρότερο ή ίσο (θέτει την ερώτηση: είναι η αριστερή έκφραση μικρότερη ή ίση από την δεξιά)
>=	μεγαλύτερο ή ίσο (θέτει την ερώτηση: είναι η αριστερή έκφραση μεγαλύτερη ή ίση από την δεξιά)
!=	διάφορο (θέτει την ερώτηση: είναι η αριστερή έκφραση διαφορετική από την δεξιά)
==	ίσο με (θέτει την ερώτηση: είναι η αριστερή έκφραση ίση με την δεξιά)

Οι λογικοί τελεστές λειτουργούν όπως και οι συγκριτικοί, επιστρέφονταν μια απάντηση απόφασης, true ή false (αλήθεια ή ψέμα) εκτελώντας μια λογική πράξη, ανάμεσα σε δυο εκφράσεις, μια στην αριστερή πλευρά του τελεστή και μια στην δεξιά πλευρά του τελεστή. Η

απόφαση από τους λογικούς τελεστές, true ή false (αλήθεια ή ψέμα) μπορεί στην συνέχεια να χρησιμοποιηθεί από κατάλληλες αλγοριθμικές δομές επιλογής ή επανάληψης.

Λογικοί Τελεστές - Logical Operators	
!	άρνηση (όχι), στην πληροφορική ονομάζεται και λογικό NOT (χρησιμοποιείται σε λογικές εκφράσεις, για την δημιουργία αντιστροφών), ονομάζεται και *αντιστροφέας*
&&	σύζευξη (και), στην πληροφορική ονομάζεται και λογικό AND
||	διάζευξη (ή), στην πληροφορική ονομάζεται και λογικό OR

Οι τελεστές ανάθεσης είναι μια ομάδα τελεστών που βασίζονται όμως σε έναν και μόνο τελεστή. Τον τελεστή **=**, ο οποίος όπως έχει ήδη αναφερθεί, δεν υποδηλώνει την ισότητα των μαθηματικών, αλλά την έννοια της ανάθεσης (ή τοποθέτησης ή εκχώρησης ή μεταφοράς) στην μεταβλητή η οποία προηγείται του συμβόλου **=**, που βρίσκεται στην μεταβλητή μετά από το **=**. Οι τελεστές της ομάδας αυτής, εκτελούν πρώτα κάποια από τις βασικές αριθμητικές πράξεις και στην συνέχεια αναθέτουν το αποτέλεσμα της πράξης στην μεταβλητή που βρίσκεται αριστερά από το **=**. Είναι δηλαδή, ένας συνδυασμός του τελεστή **=** με έναν από τους βασικούς αριθμητικούς τελεστές **+ - * /** και **%**.

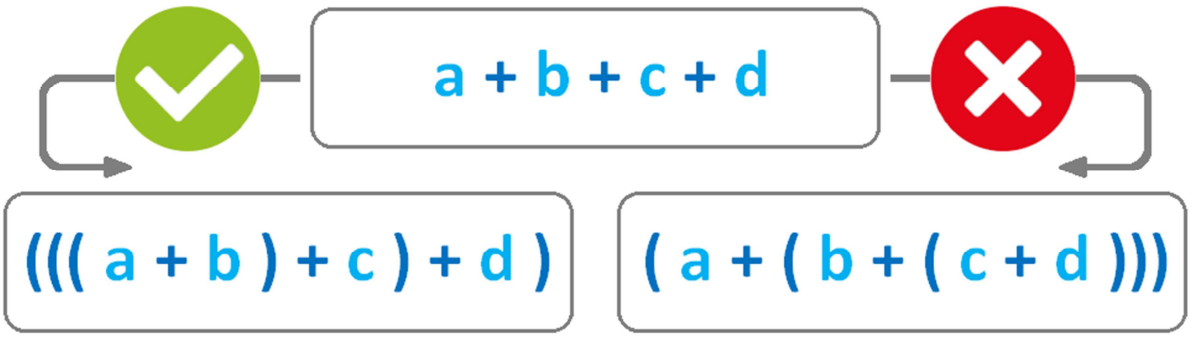
Τελεστές Ανάθεσης/Εκχώρησης - Assignments Operators	
=	στην εντολή **a=b**, αναθέτει το περιεχόμενο του **b** και στο **a**
+=	στην εντολή **a+=b**, εκτελεί την πράξη, **a=a+b** (πράξη και μετά ανάθεση)
-=	στην εντολή **a-=b**, εκτελεί την πράξη, **a=a-b** (πράξη και μετά ανάθεση)
*=	στην εντολή **a*=b**, εκτελεί την πράξη, **a=a*b** (πράξη και μετά ανάθεση)
/=	στην εντολή **a/=b**, εκτελεί την πράξη, **a=a/b** (πράξη και μετά ανάθεση)
%=	στην εντολή **a%=b**, εκτελεί την πράξη, **a=%b** (πράξη και μετά ανάθεση)

Στον επόμενο πίνακα παρουσιάζονται συγκεντρωμένοι οι τελεστές από τις υπόλοιπες 3 κατηγορίες τελεστών, χωρίς όμως να επεξηγούνται οι λειτουργίες τους.

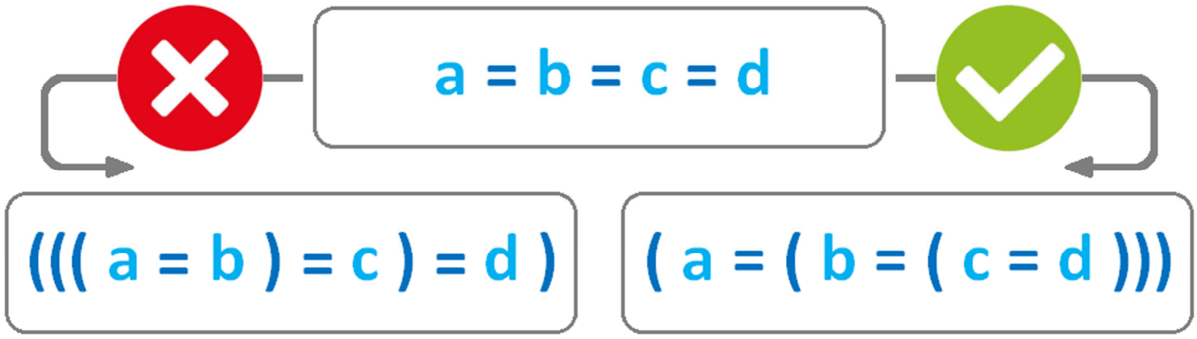
Ειδικοί Τελεστές & Τελεστές Δεικτοδότησης - bitwise & Indexing Operators				
&	&=	~	<<=	*
|	^=	>>	>>=	&
^	|=	<<	&=	

3.6. Προτεραιότητες Τελεστών

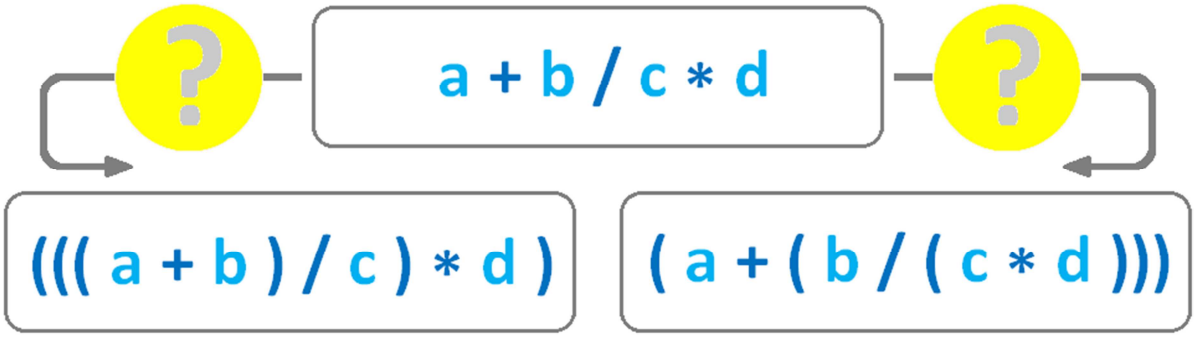
Ο μεγάλος αριθμός των διαθέσιμων τελεστών και των συνδυασμών τους, υποδηλώνει την ικανότητα μιας γλώσσας προγραμματισμού να υποστηρίζει την σύνταξη σύνθετων μαθηματικών και λογικών εκφράσεων. Καθόσον μια σύνθετη έκφραση μπορεί να περιλαμβάνει πολλούς τελεστές, τίθεται θέμα προτεραιότητας με την οποία οι τελεστές θα συμμετάσχουν στην πράξη. Γενικότερα, σε μια έκφραση υπολογισμού (αριθμητικού ή λογικού), η φορά υπολογισμού ονομάζεται *προσεταιριστικότητα.*



Εάν για παράδειγμα, στην έκφραση **a + b + c + d**, εφαρμόζαμε παντού αριστερή προσεταιριστικότητα τότε η συγκεκριμένη έκφραση θα εκτελείτο σαν **(((a + b) + c) + d)**, ενώ αντίθετα εάν είχαμε παντού δεξιά προσεταιριστικότητα τότε η έκφραση θα εκτελείτο σαν **(a + (b + (c + d)))**, λαμβάνοντας υπόψη ότι και στις δυο περιπτώσεις, πρώτα εκτελούνται αυτά που βρίσκονται εμφωλιασμένα, μέσα σε παρενθέσεις. Στην γλώσσα C++, η έκφραση **a + b + c + d**, έχει αριστερή προσεταιριστικότητα και άρα εκτελείται σαν **(((a + b) + c) + d)**, ενώ η έκφραση **a = b = c = d**, έχει δεξιά προσεταιριστικότητα και άρα εκτελείται σαν **(a = (b = (c = d)))**.



Οι περισσότεροι τελεστές στην γλώσσα C++, έχουν αριστερή προσεταιριστικότητα, αλλά όχι όλοι. Επιπλέον, σε μια έκφραση της μορφής **a + b / c * d**, η εκτέλεση **(((a + b) / c) * d)** και η εκτέλεση **(a + (b / (c * d)))**, δεν θα δώσουν τα ίδια αποτελέσματα.



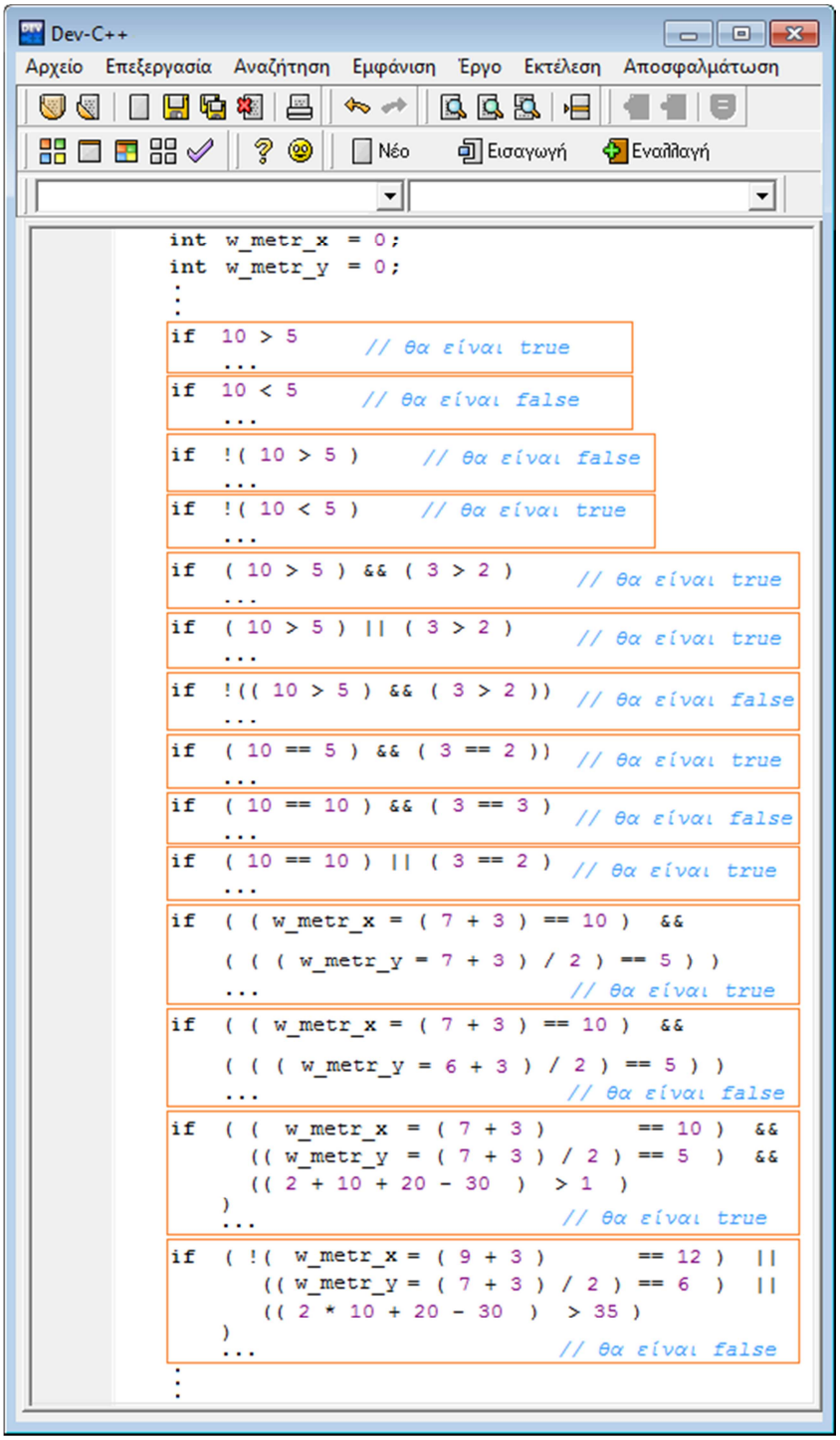
Την προσεταιριστικότητα δεν μπορούμε να την αλλάξουμε αλλά εάν θέλουμε να αλλάξουμε την προτεραιότητα, μπορούμε να το κάνουμε χρησιμοποιώντας παρενθέσεις. Συνεπώς η προτεραιότητα με την οποία οι τελεστές συμμετέχουν στις πράξεις αλλά και η προσεταιριστικότητα καθορίζει και το αποτέλεσμα της πράξης. Η γλώσσα προγραμματισμού C++, διαθέτει συγκεκριμένους κανόνες για τον καθορισμό της προτεραιότητας και της προσεταιριστικότητας μεταξύ των τελεστών. Οι κανόνες προτεραιότητας ονομάζονται *επίπεδα προτεραιότητας*, με τους τελεστές υψηλότερης προτεραιότητας να επιδρούν πριν από τους τελεστές χαμηλότερης προτεραιότητας. Οι κανόνες προτεραιότητας συνδυάζονται με τους κανόνες προσεταιριστικότητας.

Βασικοί Τελεστές	Προσεταιριστικότητα
υψηλότερη προτεραιότητα	
( ) [ ]	→
! **++** **--** **+** (πρόσημο) **-** (πρόσημο)	←
***** **/** **%**	→
+ **-**	→
< **<** **<=** **>=**	→
== **!=**	→
&&	→
||	→
?: (ειδικός τριαδικός τελεστής)	←
= **+=** **-=** ***=** **%=**	←
,	→
χαμηλότερη προτεραιότητα	

Ένας προγραμματιστής ο οποίος γνωρίζει καλά τους κανόνες προτεραιότητας μπορεί να συντάξει μια σύνθετη έκφραση χωρίς την ανάγκη τοποθέτησης παρενθέσεων. Αποτελεί όμως ισχυρή σύσταση να χρησιμοποιούνται στα προγράμματα, οι παρενθέσεις για να εξασφαλίζουν την ορθότητα σύνταξης των σύνθετων λογικών και αριθμητικών εκφράσεων υπολογισμού.

3.7. Παραδείγματα συνδυασμού Τελεστών

Η γλώσσα C++, επιτρέπει τη σύνταξη σύνθετων εκφράσεων με συνδυασμούς τελεστών από πολλές και διαφορετικές κατηγορίες. Επιπλέον, μέσα στις συνθέσεις αυτές μπορούν να συμπεριλαμβάνονται πράξεις, αριθμητικές, λογικές ή ενέργειες διαμόρφωσης περιεχομένου.



Σύνοψη κεφαλαίου

Η γλώσσα C++, διαθέτει μια πολύ μεγάλη ομάδα επιλογών για την διαμόρφωση της συμπεριφοράς των μεταβλητών της. Αυτές οι διαθέσιμες επιλογές αφορούν, τους τύπους των δεδομένων, τους τροποποιητές, τα μεγέθη, το περιεχόμενο, τα αριθμητικά συστήματα, την κωδικοποίηση και πολλά άλλα πρόσθετα και εξειδικευμένα στοιχεία. Ειδικότερα, στο κατά το ISO 2020 πρότυπο, η γλώσσα C++, διαθέτει ακόμα περισσότερες επιλογές. Στην βάση των εκπαιδευτικών στόχων του συγκεκριμένου συγγράμματος, περιγράφεται στο τρίτο κεφάλαιο, ένα μέρος αυτών των δυνατοτήτων, ικανό όμως να εφοδιάσει τους εκπαιδευόμενους με όλη την κατάλληλη και απαραίτητη τεχνογνωσία. Ολοκληρώνοντας ο εκπαιδευόμενος αναγνώστης, την μελέτη αυτών των τελευταίων παραγράφων του τρίτου κεφαλαίου, έχει ήδη εισέλθει μέσα στον κόσμο του «χαμηλού επιπέδου» της γλώσσας. Έχει δηλαδή διαμορφώσει μια αρχική προσωπική αντίληψη για, α) το τι σημαίνει «χαμηλό επίπεδο» γλώσσας, β) το τι είναι «συμπυκνωμένο» λεξιλόγιο της γλώσσας και γ) το τι είναι οι πολλοί «διακόπτες». Αυτή η ανάγκη της επικέντρωσης στα εσωτερικά, ενδότερα τεχνικά στοιχεία της γλώσσας C++, είναι μια γενικότερη κατάσταση που επικρατεί για όλες τις γλώσσες προγραμματισμού σε όλα τα software house (τις εταιρίες ανάπτυξης λογισμικού), απαιτώντας από τους προγραμματιστές, να μάθουν να μαθαίνουν τις ιδιαιτερότητες της γλώσσας για να μπορούν στη συνέχεια να την χρησιμοποιούν αποδοτικά.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Ποιους θεμελιώδεις τύπους δεδομένων της γλώσσας C++, γνωρίζετε;

Ερώτηση-2: Τι σημαίνει ο όρος, *εμβέλεια μεταβλητής*, στον προγραμματισμό;

Ερώτηση-3: Τι σημαίνει ο όρος, *αρχικοποίηση μεταβλητής*, στον προγραμματισμό;

Ερώτηση-4: Περιγράψτε τι σημαίνει, ο όρος, *δήλωση εκχώρησης*;

Ερώτηση-5: Όταν δηλώνουμε μια μεταβλητή, γιατί χρειαζόμαστε τους τροποποιητές;

Ερώτηση-6: Περιγράψτε, τι γνωρίζετε για τις σταθερές στην γλώσσα C++;

Ερώτηση-7: Περιγράψτε, τι γνωρίζετε για τους ειδικούς χαρακτήρες διαφυγής, δώστε ένα παράδειγμα;

Ερώτηση-8: Στην πράξη, **w_synolo = (w_poso + 12) / w_ypoloipo**, ποιο είναι οι τελεστές και ποιο είναι οι τελεστέοι;

Ερώτηση-9: Ο τελεστής **/**, τι σημασία έχει, σε αριθμητικές πράξεις της C++;

Ερώτηση-10: Ο τελεστής **%**, τι σημασία έχει, σε αριθμητικές πράξεις της C++;

Ερώτηση-11: Περιγράψτε, η πράξη **++w_metritis**, σε σχέση με την πράξη **w_metritis ++**, τι διαφορά έχει;

Ερώτηση-12: Ποιους λογικούς τελεστές γνωρίζετε;

Ερώτηση-13: Περιγράψτε, τι σημαίνει ο όρος *προτεραιότητα*, στις πράξεις υπολογισμού και στις λογικές εκφράσεις της γλώσσας C++;

Ερώτηση-14: Περιγράψτε, τι σημαίνει ο όρος *προσεταιριστικότητα*, στις πράξεις υπολογισμού και στις λογικές εκφράσεις της γλώσσας C++;

Ερώτηση-15: Είναι σωστό ότι μπορούμε να αλλάξουμε την προτεραιότητα των τελεστών;

Ερώτηση-16: Είναι σωστό ότι μπορούμε να αλλάξουμε την προσεταιριστικότητα των τελεστών;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-3: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 3.1. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 3.2. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 3.3. του κεφαλαίου.

Απάντηση-7: Συμβουλευτείτε την ενότητα 3.4. του κεφαλαίου.

Απάντηση-8: Τελεστές είναι οι, **= (+) /** και τελεστέοι είναι οι, **w_synolo**, **w_poso**, το **12** και το **w_ypoloipo**.

Απάντηση-9: Εκτελεί την πράξη της διαίρεσης και αποκόπτει το υπόλοιπο.

Απάντηση-10: Εκτελεί την πράξη της διαίρεσης και αποκόπτει το πηλίκο.

Απάντηση-11: Συμβουλευτείτε την ενότητα 3.5. του κεφαλαίου.

Απάντηση-12: Συμβουλευτείτε την ενότητα 3.5. του κεφαλαίου.

Απάντηση-13: Συμβουλευτείτε την ενότητα 3.6. του κεφαλαίου.

Απάντηση-14: Συμβουλευτείτε την ενότητα 3.6. του κεφαλαίου.

Απάντηση-15: Ναι, είναι σωστό. Μπορούμε να το κάνουμε με τις παρενθέσεις, αυτές που είναι εσωτερικά, εκτελούνται πρώτες.

Απάντηση-16: Όχι, δεν μπορούμε να αλλάξουμε την προσεταιριστικότητα.

4. Η ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΗΝ ΜΗΧΑΝΗ

Σκοπός και επιμέρους στόχοι

Ο σκοπός του τέταρτου κεφαλαίου είναι η παρουσίαση των κατάλληλων μηχανισμών προγραμματισμού με τους οποίους ένα λογισμικό αποκτά την δυνατότητα επικοινωνίας και αλληλεπίδρασης με το εξωτερικό περιβάλλον (τις συσκευές εισόδου και εξόδου). Οι εκπαιδευόμενοι αποκτούν την γνώση απλών εντολών της γλώσσας C++, για την δημιουργία κατάλληλων και λειτουργικών διεπαφών.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του τέταρτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να αντιλαμβάνονται την σημασία της διεπαφής για τα προγράμματα των υπολογιστών,
- να δημιουργούν απλά μηνύματα στην οθόνη και να σχεδιάζουν απλές μορφές διεπαφών,
- να καταλαβαίνουν πως ένα πρόγραμμα υπολογιστή, επικοινωνεί μέσω μιας διεπαφής, με το εξωτερικό περιβάλλον,
- να συντάσσουν μια εντολή εξόδου στην οθόνη, με την εντολή **printf()**, της C++,
- να συντάσσουν μια εντολή εισόδου από την οθόνη, με την εντολή **scanf()**, της C++,
- να τοποθετούν προσδιοριστές και μορφοποιητές μέσα στην εντολή **printf()**, της C++,
- να τοποθετούν προσδιοριστές μέσα στην εντολή **scanf()**, της C++.

Έννοιες – Λέξεις Κλειδιά

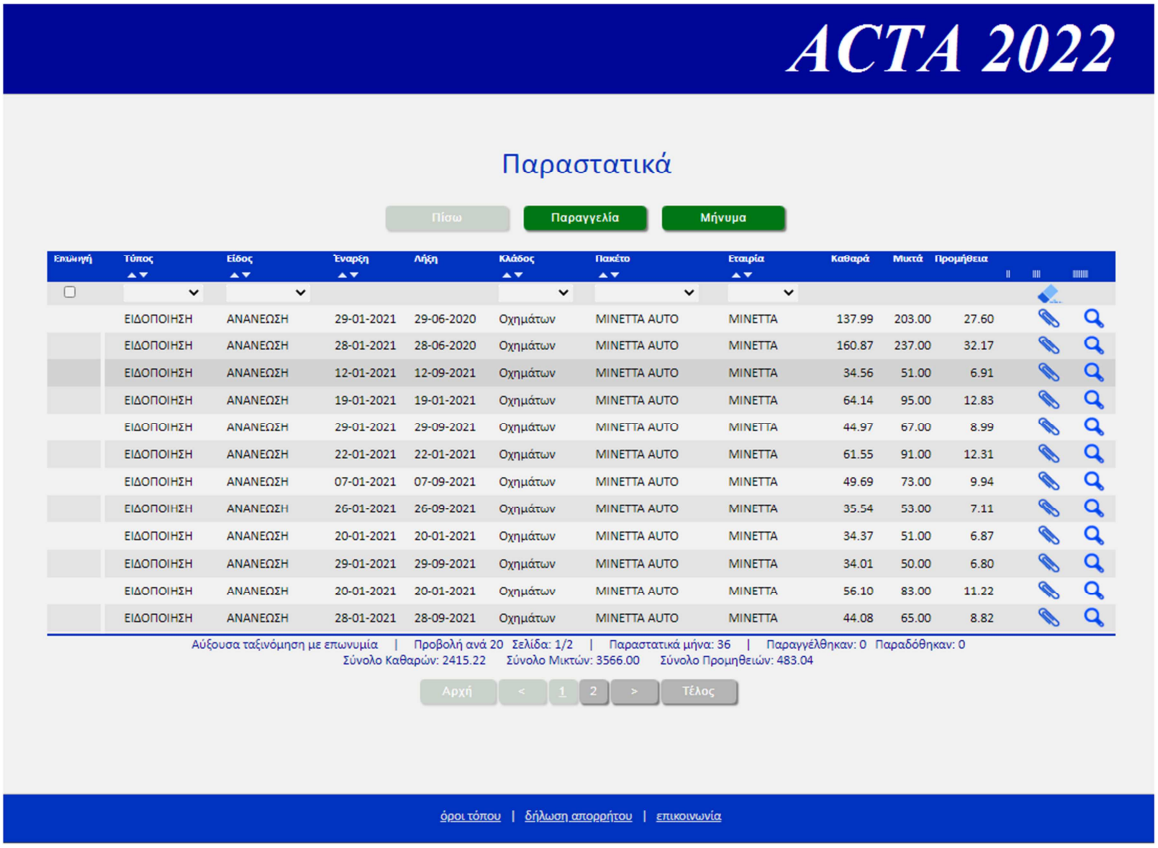
Διεπαφή, Αλληλεπίδραση, Μορφοποιημένη προβολή δεδομένων, Μορφοποιημένη υποδοχή δεδομένων, Προσδιοριστές, Μορφοποιητές, Λίστα ορισμάτων, Τελεστής διεύθυνσης.

Εισαγωγικές Παρατηρήσεις

Η ανάπτυξη ενός προγράμματος σε μια γλώσσα προγραμματισμού δεν περιλαμβάνει αποκλειστικά και μόνο μαθηματικές εκφράσεις, τελεστές, υπολογισμούς και μετασχηματισμούς δεδομένων. Ένα πρόγραμμα είναι κυρίως μια διαρκής αλληλεπίδραση ανάμεσα στο λογισμικό και τον χρήστη. Μέσα από αυτή την διαδικασία τα λογισμικά αποκτούν ικανότητες και αξία. Την επικοινωνία αυτή, τα προγράμματα την υλοποιούν με διάφορους τρόπους, όπως την αποστολή ενός sms μηνύματος, την εκτύπωση μιας σελίδας αναφοράς, την αυτόματη ενημέρωση ενός άλλου λογισμικού, αλλά κυρίως με κατάλληλες οθόνες με τις οποίες εγκαθιστούν μια αμφίδρομη μορφή επικοινωνίας ανάμεσα στον υπολογιστή και τον άνθρωπο χρήστη.

Ο υπολογιστής και κατά επέκταση το πρόγραμμα, που βρίσκεται πίσω από αυτόν, ζητά από τον χρήστη, την πληκτρολόγηση, μιας εντολής ή μιας πληροφορίας. Την εντολή αυτή η μηχανή την επεξεργάζεται και προχωρά ακολούθως στην υποβολή προς τον χρήστη, μιας νέας πληροφορίας, αναμένοντας μια κατάλληλη απάντηση ή μια νέα εντολή, συνεχίζοντας

επαναλαμβανόμενα αυτή την διαδικασία, μέχρι να λάβει μια τελική εντολή αποσύνδεσης. Αυτή η συνεχόμενη διαδικασία αλληλεπίδρασης μεταξύ των δυο μερών, στην πληροφορική, ονομάζεται *διεπαφή*. Οι διεπαφές στην πληροφορική μπορούν να αφορούν επικοινωνία ανάμεσα σε μια μηχανή και έναν άνθρωπο ή επικοινωνία ανάμεσα σε δυο μηχανές. Με τον όρο μηχανή μπορούμε να αναφερόμαστε και σε έναν υπολογιστή, ένα περιφερειακό ή και ένα λογισμικό. Μια αυτόματη δηλαδή επικοινωνία ανάμεσα σε δυο λογισμικά, είναι μια διεπαφή. Γενικότερα, η έννοια της διεπαφής υπάρχει παντού γύρω μας, καθώς η εγκατάσταση κάθε μορφής επικοινωνίας ανάμεσα σε δυο μέρη, είναι μια μορφή διεπαφής.



Ο σταδιακός εμπλουτισμός των γνώσεων των εκπαιδευομένων μέσω των συγγραμμάτων της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ACTA, στοχεύει στην απόκτηση των κατάλληλων ικανοτήτων για την υλοποίηση ικανών διεπαφών. Η γλώσσα C++, διαθέτει ένα πολύ μεγάλο πλούτο βιβλιοθηκών, συναρτήσεων και εντολών για την δημιουργία σύγχρονων και δυναμικών διεπαφών. Καθώς όμως μια σύνθετη διεπαφή απαιτεί εξειδικευμένες γνώσεις και προγραμματιστική εμπειρία, επιλέγουμε αρχικά να

χρησιμοποιήσουμε δυο απλές εντολές της C++, για την υλοποίηση απλουστευμένων και ευκολονόητων διεπαφών. Οι δυο αυτές εντολές, είναι η εντολή **printf()** και η εντολή, **scanf()**. Η εντολή **printf()** μπορεί να χρησιμοποιηθεί για την αποστολή πληροφοριών προς την οθόνη και κατά επέκταση προς τον χρήστη, ενώ η εντολή **scanf()**, μπορεί να χρησιμοποιηθεί για την υποδοχή πληροφοριών από το πληκτρολόγιο και το χρήστη. Οι εντολές αυτές περιγράφονται στην βιβλιογραφία και σαν συναρτήσεις εξόδου και εισόδου αντίστοιχα.

4.1. Η εντολή εξόδου, printf()

Η εντολή εξόδου **printf()**, αποτελεί μια βασική εντολή της γλώσσας C++, για την μορφοποιημένη προβολή των δεδομένων (**print f**ormatted data), που βρίσκονται μέσα στην παρένθεση **()**, προς μια συσκευή εξόδου (οθόνη, εκτυπωτής, κ.λπ.). Η εντολή περιλαμβάνεται μέσα στην βασική βιβλιοθήκη **<stdio.h>** και σαν συνέπεια, εάν η βιβλιοθήκη δεν κληθεί για ενσωμάτωση, ο compiler δεν θα αναγνωρίσει την εντολή. Η γενική σύνταξη της εντολής είναι,

```
printf("μορφοποίηση…", λίστα ορισμάτων) ;
```

Η **"μορφοποίηση…"**, που αναπτύσσεται μέσα στα διπλά εισαγωγικά **" "**, μπορεί να περιλαμβάνει,

1) απλούς χαρακτήρες σε μορφή ελεύθερου κειμένου,
2) δηλώσεις προσδιορισμού (ονομάζονται και *προσδιοριστές*), που καθορίζουν τον τρόπο με τον οποίο θα εμφανιστεί το περιεχόμενο από τα ορίσματα που ακολουθούν,
3) δηλώσεις μορφοποίησης (ονομάζονται και *μορφοποιητές*), που καθορίζουν πρόσθετους κανόνες μορφοποίησης του κειμένου, όπως η αλλαγή γραμμής, η αλλαγή σελίδας, κ.λπ.,
4) συνδυασμό κειμένου, προσδιοριστών και μορφοποιητών.

Η **λίστα ορισμάτων…**, μπορεί να περιλαμβάνει,

1) απλούς χαρακτήρες σε μορφή ελεύθερου κειμένου, που πρέπει να αναπτύσσεται μέσα σε διπλά εισαγωγικά **" "**,
2) μεταβλητές.

Η εντολή **printf()**, διαθέτει έναν μεγάλο αριθμό συνδυασμών, από προσδιοριστές, μορφοποιητές και χαρακτήρες διαφυγής (escape sequence). Στα πλαίσια των στόχων του συγγράμματος αυτού, αναπτύσσεται ένα μέρος αυτών των δυνατοτήτων. Μια δήλωση προσδιοριστή, αρχίζει με το σύμβολο **%** και αμέσως μετά ακολουθεί ο χαρακτήρας προσδιορισμού. Σε μια **printf()**, μπορούν να οριστούν πολλοί προσδιοριστές, τόσοι, όσα και τα ορίσματα που ακολουθούν.

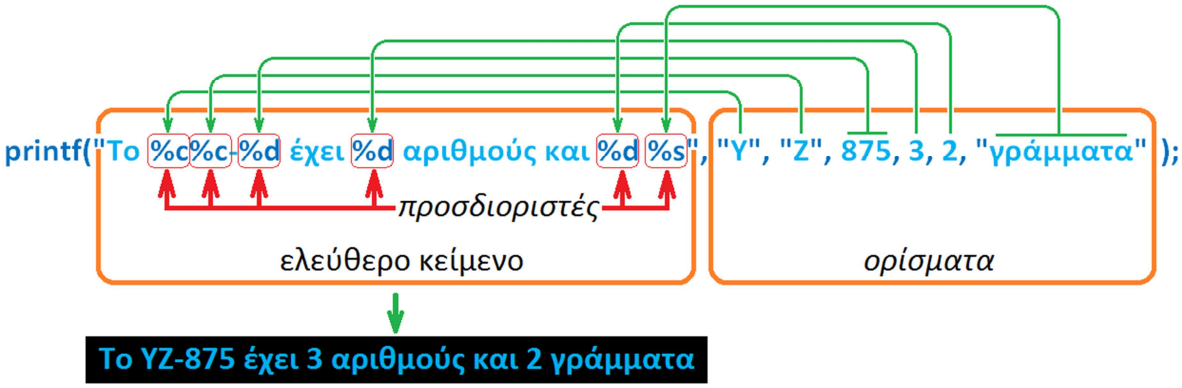
Βασικοί Προσδιοριστές	
%d	int
%i	int
%u	unsigned int
%f	float/double (με υποδιαστολή)
%e	float/double (σε εκθετική μορφή)
%g	float/double, (επιλέγεται αυτόματα %f ή %e ανάλογα με το περιεχόμενο)
%c	char χαρακτήρας
%s	αλφαριθμητικό (από συμβολοσειρά ή πίνακα)

Για παράδειγμα, η εντολή,

```
printf("Το %c%c-%d έχει %d αριθμούς και %d %s", "Y", "Z", 875, 3, 2, "γράμματα");
```

θα εμφανίσει στην οθόνη, το κείμενο, **Το YZ-875 έχει 3 αριθμούς και 2 γράμματα**

το οποίο, δημιουργείται όπως σχηματικά επεξηγείται στην επόμενη εικόνα, με το κάθε όρισμα να τοποθετείται εκεί που το «αναμένει» ο κατάλληλος προσδιοριστής.



Αντίστοιχα παραδείγματα είναι τα,

printf ("%10.2f", 873.274) ;	Θα τυπώσει το, _ _ _ _873.27
printf ("%10.2f", 873.278) ;	Θα τυπώσει το, _ _ _ _873.28
printf ("%-10.2f", 873.274) ;	Θα τυπώσει το, 873.27_ _ _ _
printf ("%10s", "ACTA");	Θα τυπώσει το, _ _ _ _ _ _ACTA
printf ("%-10s", "ACTA");	Θα τυπώσει το, ACTA_ _ _ _ _ _
printf ("%d", 106) ;	Θα τυπώσει το, 106
printf ("%c", 106) ;	Θα τυπώσει το, j
printf ("%10d", 106) ;	Θα τυπώσει το, _ _ _ _ _ _ _106
printf ("Περιέχει %d", w_poso) ;	Θα τυπώσει το, Περιέχει 234 (ότι είχε η w_poso)
printf("www.acta.edu.gr");	Θα τυπώσει το, www.acta.edu.gr
printf("ACTA\nwww.acta.edu.gr");	Θα τυπώσει το, ACTA www.acta.edu.gr
printf ("%10.2f", 5.12345);	Θα τυπώσει το, _ _ _ _ _ _5.12
printf("%.3f", 3.45678);	Θα τυπώσει το, 3.457
printf("%.3f", 3.45123);	Θα τυπώσει το, 3.451

4.2. Η εντολή εισόδου, scanf()

Η εντολή εισόδου **scanf()**, αποτελεί επίσης μια βασική εντολή της γλώσσας C++, για την μορφοποιημένη υποδοχή δεδομένων (**scan f**ormatted data), προς μεταβλητές που βρίσκονται μέσα στην παρένθεση **()**, από μια συσκευή εισόδου (πληκτρολόγιο). Η εντολή περιλαμβάνεται μέσα στην βασική βιβλιοθήκη **<stdio.h>** και σαν συνέπεια, εάν η βιβλιοθήκη δεν κληθεί για ενσωμάτωση, ο compiler δεν θα αναγνωρίσει την εντολή. Η γενική σύνταξη της εντολής είναι,

scanf("μορφοποίηση…", λίστα ορισμάτων) ;

Η **"μορφοποίηση…"**, που αναπτύσσεται μέσα στα διπλά εισαγωγικά **" "**, μπορεί να περιλαμβάνει,

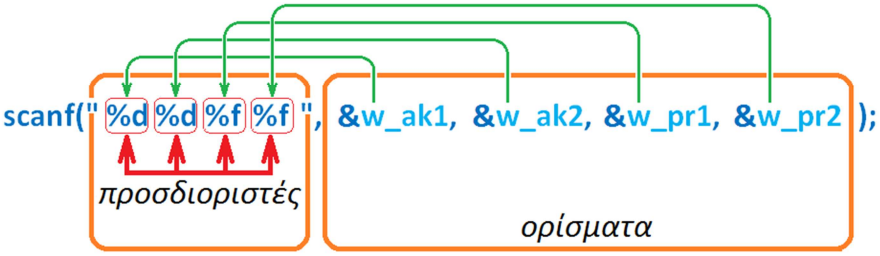
1) δηλώσεις προσδιορισμού (ονομάζονται και *προσδιοριστές*), που καθορίζουν τον τρόπο με τον οποίο τα δεδομένα εισαγωγής, θα προωθηθούν σαν περιεχόμενο, προς τα ορίσματα που ακολουθούν.

Η **λίστα ορισμάτων…**, μπορεί να περιλαμβάνει,

1) μεταβλητές.

Η εντολή **scanf()**, διαθέτει συνδυασμούς προσδιοριστών, όπως και η εντολή **printf()**. Στην συγκεκριμένη ενότητα περιγράφονται οι βασικότεροι από αυτούς. Μια δήλωση προσδιοριστή, αρχίζει με το σύμβολο **%** και αμέσως μετά ακολουθεί ο χαρακτήρας προσδιορισμού. Σε μια **scanf()**, μπορούν να οριστούν πολλοί προσδιοριστές, τόσοι, όσα και τα ορίσματα που ακολουθούν. Οι βασικοί προσδιοριστές της **scanf()**, είναι ίδιοι με τους προσδιοριστές της εντολής **printf()**. Τα ορίσματα εισόδου που αφορούν μεταβλητές τύπου, αριθμών ή χαρακτήρα πρέπει να αρχίζουν με το σύμβολο **&**. Ορίσματα εισόδου που αφορούν μεταβλητές αλφαριθμητικού τύπου, δεν πρέπει να αρχίζουν με το σύμβολο **&**. Αυτός ο τελεστής **&** (ονομάζεται και *τελεστής διεύθυνσης*) δεν εντάσσεται πραγματικά στην λειτουργικότητα της ίδιας της εντολής **scanf()**. Ο τελεστής **&** υποδεικνύει στον compiler ότι θα πρέπει να

ανακατευθύνει αυτό που πληκτρολογήθηκε, προς την διεύθυνση μνήμης στην οποία βρίσκεται τοποθετημένη η μεταβλητή των ορισμάτων της **scanf()**.



Παραδείγματα σύνταξης είναι τα,

scanf("%d", &w_ak1) ; διαβάζει από το πληκτρολόγιο, έναν ακέραιο και τον αποθηκεύει στην διεύθυνση της μεταβλητής, w_ak1

scanf("%d %d %f %f", &w_ak1, &w_ak2, &w_pr1, &w_pr2) ;

διαβάζει από το πληκτρολόγιο, σε σειρά, δυο ακεραίους και δυο πραγματικούς αριθμούς και τους αποθηκεύει αντίστοιχα στις διευθύνσεις των μεταβλητών, w_ak1, w_ak2, w_pr1 και w_pr2

Σύνοψη κεφαλαίου

Στον τέταρτο κεφάλαιο παρουσιάστηκαν αναλυτικά δυο απαραίτητες εντολές της γλώσσας προγραμματισμού C++, με τις οποίες είναι δυνατό να δημιουργηθούν σε ένα πρόγραμμα κατάλληλες διεπαφές. Το κεφάλαιο περιελάμβανε σχετικά παραδείγματα και επεξηγήσεις για τους τρόπους με τους οποίους συντάσσονται και παραμετροποιούνται οι εντολές αυτές.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε, τι σημαίνει ο όρος, *διεπαφή*;

Ερώτηση-2: Περιγράψτε, τι σημαίνει αλληλεπίδραση ανάμεσα στην μηχανή και τον χρήστη;

Ερώτηση-3: Είναι σωστό ότι δεν είναι δυνατό να έχουμε μια διεπαφή μεταξύ μηχανών;

Ερώτηση-4: Περιγράψτε, στην εντολή **printf**, τι σημαίνει ο όρος, *προσδιοριστής*;

Ερώτηση-5: Περιγράψτε, στην εντολή **printf**, τι σημαίνει ο όρος, *μορφοποιητής*;

Ερώτηση-6: Περιγράψτε, στην εντολή **scanf**, τι σημαίνει ο όρος, *προσδιοριστής*;

Ερώτηση-7: Είναι σωστό, σε μια εντολή **printf**, να έχουμε 3 μεταβλητές και 4 προσδιοριστές;

Ερώτηση-8: Περιγράψτε, τι θα τυπώσει η εντολή, **printf("ELTA_1\nELTA_1\nELTA_1")**;

Ερώτηση-9: Εξηγήστε για ποιον λόγο η εντολή, **printf ("%c", 106)**,θα τυπώσει το γράμμα **j**;

Ερώτηση-10: Περιγράψτε, τι θα τυπώσει η εντολή, **printf("%.3f", 1.1234678)** και γιατί;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-3: Όχι δεν είναι σωστό. Μπορούμε να έχουμε διεπαφές μεταξύ μηχανών.

Απάντηση-4: Συμβουλευτείτε την ενότητα 4.1. του κεφαλαίου.

Απάντηση-5: Συμβουλευτείτε την ενότητα 4.1. του κεφαλαίου.

Απάντηση-6: Συμβουλευτείτε την ενότητα 4.2. του κεφαλαίου.

Απάντηση-7: Όχι, είναι λάθος. Πρέπει να έχουμε ίσο αριθμό προσδιοριστών και μεταβλητών.

Απάντηση-8: Θα τυπώσει σε 3 διαφορετικές γραμμές με το **ELTA_1**. Την αλλαγή γραμμής προκαλεί ο ειδικός χαρακτήρας **\n** .

Απάντηση-9: Διότι το 106 είναι ο αντίστοιχος κωδικός ASCII για το γράμμα j, που ζητείται με τον προσδιοριστή **%c**.

Απάντηση-10: Θα τυπώσει **1.124**, με το τελευταίο ψηφίο (το 4) να προκύπτει από την στρογγυλοποίηση του επόμενου ψηφίου της ακολουθίας (το 8).

5. ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ

Σκοπός και επιμέρους στόχοι

Ο σκοπός του πέμπτου κεφαλαίου του συγγράμματος είναι να εισάγει τον εκπαιδευόμενο στην διαδικασία της δημιουργίας ευέλικτων προγραμμάτων τα οποία, με βάση το περιεχόμενο μεταβλητών και των καταστάσεων συνθηκών, θα υποστηρίζουν με κατάλληλο τρόπο, την αλλαγή της ροής εκτέλεσης ενός προγράμματος C++.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του πέμπτου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

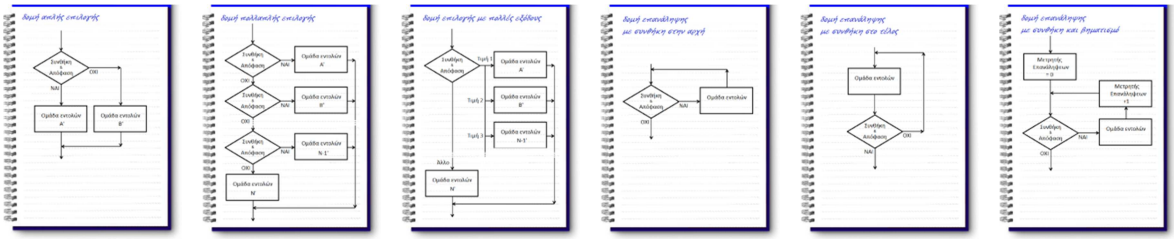
- να αντιλαμβάνονται και να περιγράφουν τις τρεις βασικές αλγοριθμικές δομές επιλογής στις γλώσσες προγραμματισμού,
- να μπορούν να συντάξουν κάθε μια από τις τρεις βασικές αλγοριθμικές δομές επιλογής, στην γλώσσα προγραμματισμού C++,
- να αντιλαμβάνονται την χρησιμότητα των συντακτικών κανόνων της γλώσσας προγραμματισμού C++,
- να αναφέρουν λόγους για τους οποίους θα αποφασίζουν ποια από τις τρεις βασικές αλγοριθμικές δομές επιλογής, θα χρησιμοποιούν,
- να δημιουργούν ευέλικτα προγράμματα τα οποία θα εξαρτώνται από το περιεχόμενο των μεταβλητών, το αποτέλεσμα των υπολογισμών και τις καταστάσεις των λογικών συνθηκών,
- να χρησιμοποιούν τελεστές σύγκρισης για να ελέγχουν την ροή ενός προγράμματος,
- να ελέγχουν και να εκτελούν μετατροπές σε έναν κώδικα, στην γλώσσα προγραμματισμού C++.

Έννοιες – Λέξεις Κλειδιά

Έλεγχος συνθήκης, Απόφαση, Διακόπτης, Εσωτερική δομή επιλογής, Εξωτερική δομή επιλογής, Εμφωλιασμένες δομές επιλογής, Δομές πολλαπλών επιλογών, Στηλοθέτηση κώδικα, Εντολή if, Εντολή if ... else ..., Εντολή switch.

Εισαγωγικές Παρατηρήσεις

Με την ολοκλήρωση της μελέτης των βασικών δομικών στοιχείων της γλώσσας το επόμενο απαραίτητο δομικό γνωσιακό πεδίο για κάθε γλώσσα προγραμματισμού είναι η γνώση της χρήσης των *αλγοριθμικών δομών*. Η γλώσσα C++, διαθέτει όλες τις επιλογές διαμόρφωσης των γνωστών αλγοριθμικών δομών προγραμματισμού. Οι αλγοριθμικές δομές στον προγραμματισμό κατατάσσονται σε δυο βασικές κατηγορίες: α) τις αλγοριθμικές δομές επιλογής και β) τις αλγοριθμικές δομές επανάληψης. Επιπλέον και για κάθε μια από αυτές τις δύο βασικές κατηγορίες, διατίθενται τρεις διαφορετικές μορφές διαμόρφωσης.



Κάθε μια από τις αλγοριθμικές δομές επιλογής, αναλύεται στην συνέχεια του συγγράμματος με ολοκληρωμένο παράδειγμα εντολών στην γλώσσα προγραμματισμού C++. Επικουρικά, οι δομές παρουσιάζονται και σε μορφή λογικού διαγράμματος και σε ψευδογλώσσα.

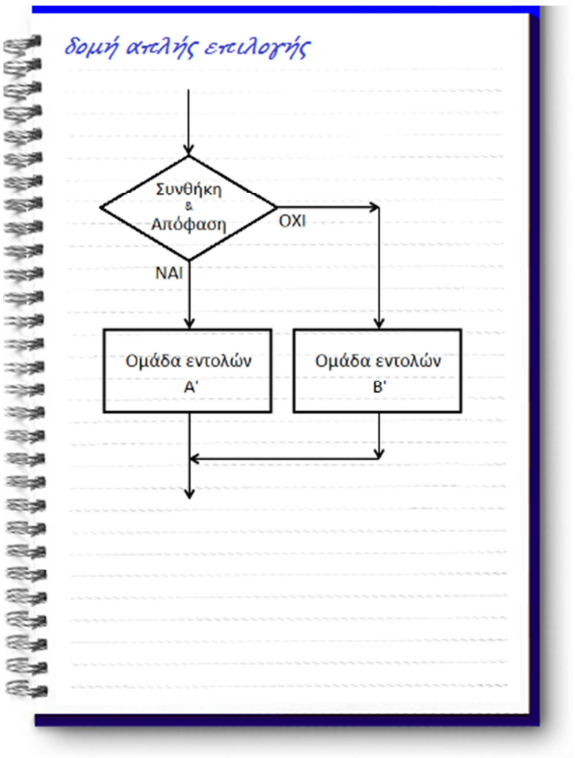
Αλγοριθμικές Δομές Επιλογής		
απλής επιλογής	πολλαπλών επιλογών	επιλογής με πολλές εξόδους

Οι αλγοριθμικές δομές επανάληψης παρουσιάζονται στο επόμενο σύγγραμμα της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών*, της ΑΣΤΑ. Οι δομές επανάληψης χρησιμοποιούνται στην περίπτωση που ένα συγκεκριμένο κομμάτι εντολών προγραμματισμού, πρέπει για κάποιον λόγο, να επαναληφθεί πολλές φορές.

Αλγοριθμικές Δομές Επανάληψης		
με την συνθήκη στην αρχή	με την συνθήκη στο τέλος	με συνθήκη και βηματισμό

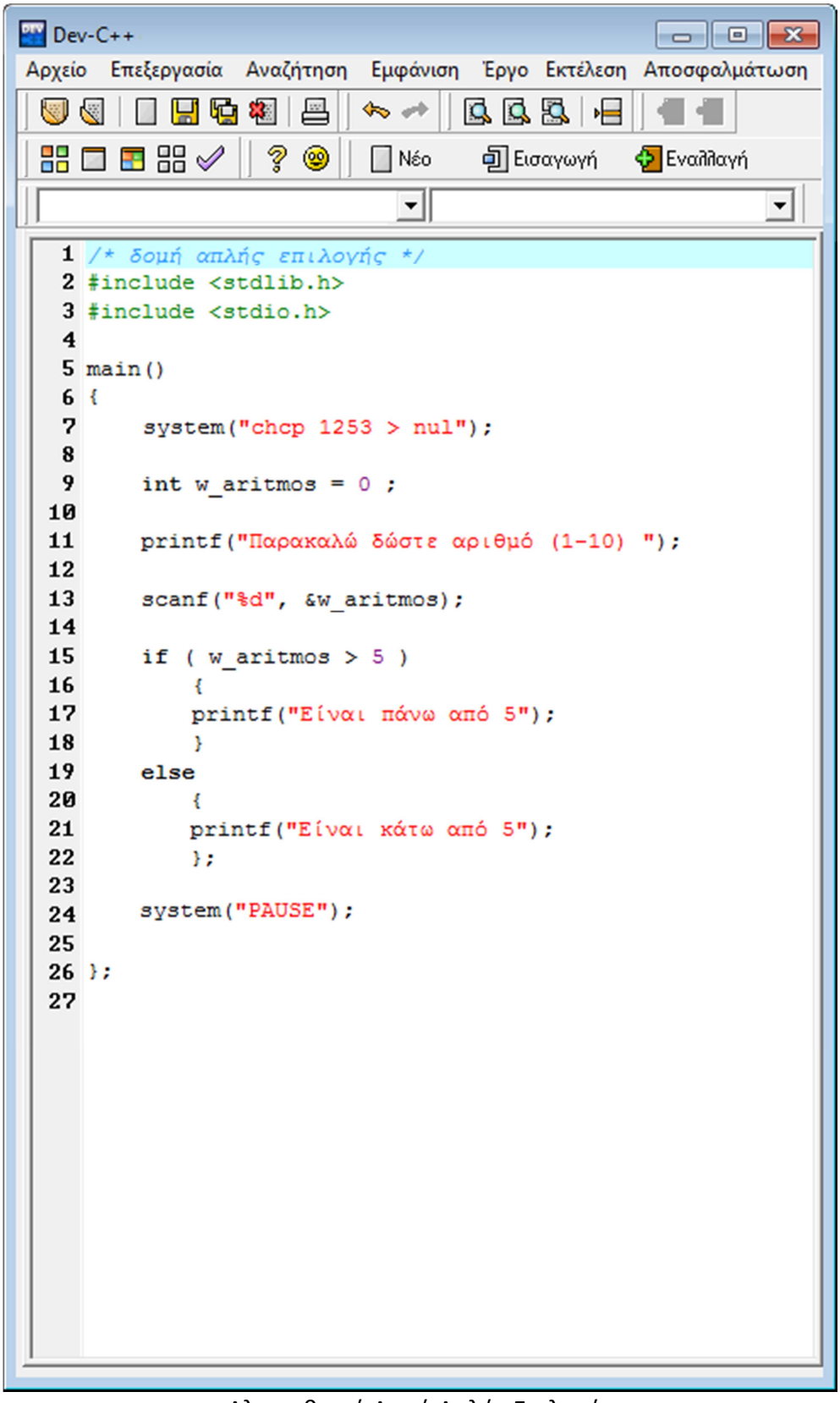
5.1. Αλγοριθμική Δομή Απλής Επιλογής – if … else …

Η αλγοριθμική δομή απλής επιλογής χρησιμοποιείται ευρέως στην επίλυση αλγορίθμων όπου είναι αναγκαίο κάποιες εντολές του προγράμματος να μην εκτελούνται με τη σειρά που είναι γραμμένες, αλλά να παρακάμπτονται και να εκτελούνται κάποιες άλλες που παρεμβάλλονται. Για να γίνει αυτό δυνατό ο μηχανισμός χρησιμοποιεί τον έλεγχο μιας συνθήκης, ο οποίος με βάση μια απόφαση, λειτούργει σαν διακόπτης επιτρέποντας ή αποτρέποντας κατάλληλα την ροή του προγράμματος. Η ελεγχόμενη συνθήκη μπορεί να αποτελείται από πολλαπλές λογικές και αριθμητικές εκφράσεις, με τελικό ζητούμενο, μια οριστική απόφαση, ΑΛΗΘΕΣ ή ΨΕΥΔΕΣ (true ή false).




```
ΕΑΝ ... ΤΟΤΕ
    ...
ΑΛΛΙΩΣ
    ...
ΕΑΝ-ΤΕΛΟΣ
```

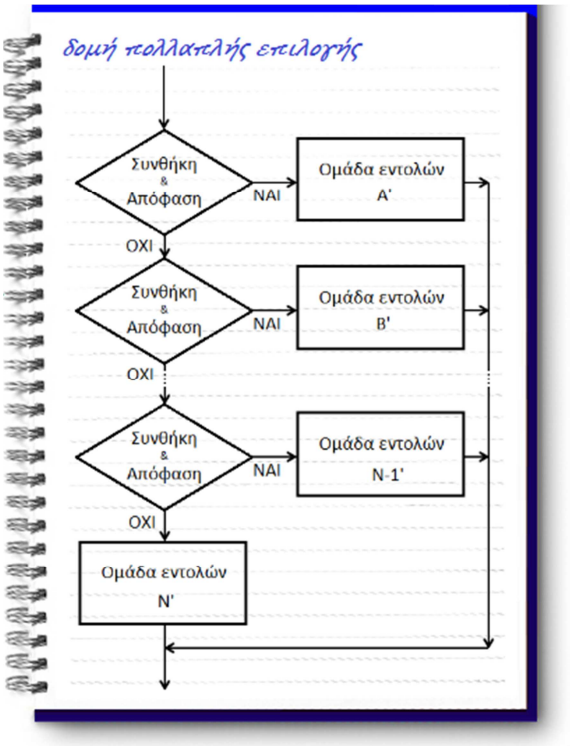
Οι εντολές που οριοθετηθούν την κάθε περιοχή ελέγχου (την ομάδα εντολών στο λογικό διάγραμμα και τις τελείες στον ψευδοκώδικα) πρέπει να βρίσκονται μέσα σε άγκιστρα **{ }**. Στην περίπτωση που σε κάποια οριοθέτηση, η εντολή είναι μόνο μια, τότε τα άγκιστρα μπορούν να παραλείπονται, όμως είναι ορθή προγραμματιστική πρακτική να τοποθετούνται, ακόμα και στην περίπτωση της μιας εντολής. Η δήλωση **else**, είναι προαιρετική.

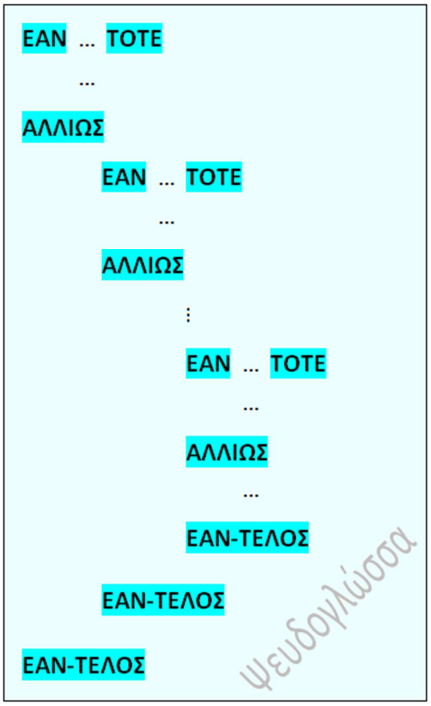


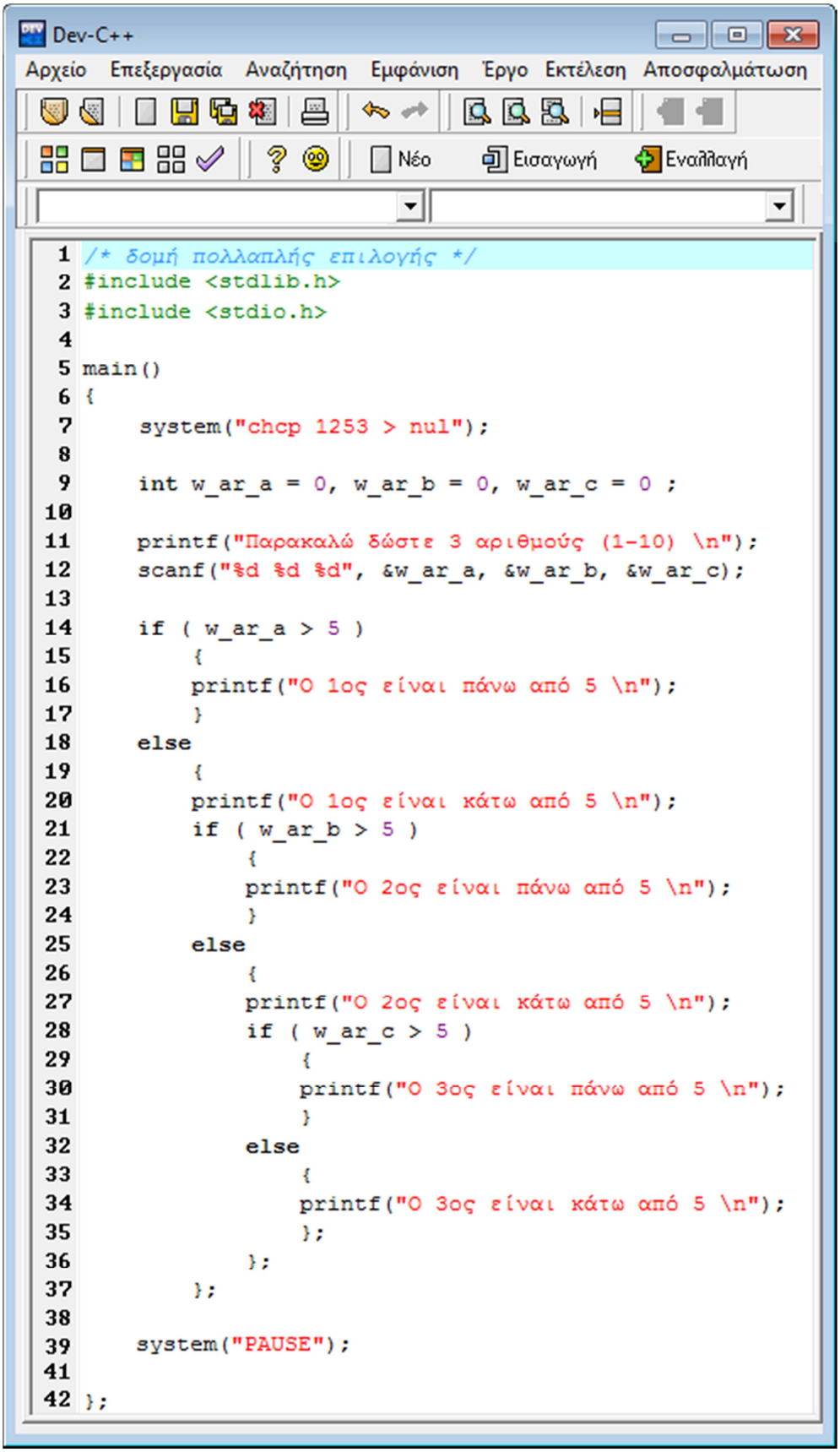
Αλγοριθμική Δομή Απλής Επιλογής

5.2. Αλγοριθμική Δομή Πολλαπλής Επιλογής – if … else … if … else …

Η αλγοριθμική δομή των πολλαπλών επιλογών είναι ένας μηχανισμός ο οποίος λειτουργεί, όπως και η απλή δομή επιλογής με την διαφορά ότι οι εκφράσεις που μπορούν να ελεγχθούν είναι πολλαπλές και η κάθε μια από αυτές λειτουργεί ανεξάρτητα από τις υπόλοιπες. Η κάθε έκφραση παράγει, μια δική της τελική απόφαση, ΑΛΗΘΕΣ ή ΨΕΥΔΕΣ (true ή false) δίνοντας την δυνατότητα της μετακίνησης της «μπίλιας» προς μια επόμενη εσωτερική δομή επιλογής που και αυτή με την σειρά της θα εκτελέσει μια παρόμοια λειτουργία απόφασης. Ολόκληρος αυτός ο μηχανισμός μπορεί να περιγραφεί σαν ένας εξωτερικός αρχικός διακόπτης ο οποίος θα επιτρέψει την ενεργοποίηση ή όχι, ενός επόμενου εσωτερικού διακόπτη, που και αυτός με την σειρά του θα ενεργοποιήσει κάποιον επόμενο διακόπτη, κ.λπ. Η σύνθετη αυτή αλγοριθμική δομή προγραμματισμού ονομάζεται, *ενσωματωμένη* ή *εμφωλιασμένη* (embedded) δομή επιλογής. Στις δομές πολλαπλών επιλογών είναι απολύτως απαραίτητο, σε όλα τα σημεία οριοθέτησης των τμημάτων κώδικα, να γίνεται χρήση των άγκιστρων **{ }**. Οι δηλώσεις **else**, είναι προαιρετικές. Επίσης, είναι απαραίτητο, όταν ένα αυτόνομο τμήμα εντολών, εμφωλιάζεται μέσα σε κάποιο άλλο τμήμα, να στηλοθετείτε προοδευτικά προς τα δεξιά, με 4 κενούς χαρακτήρες.





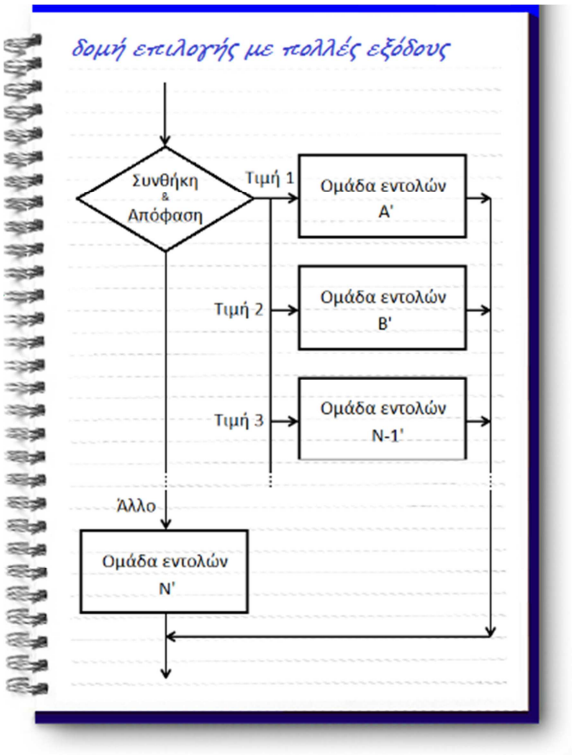


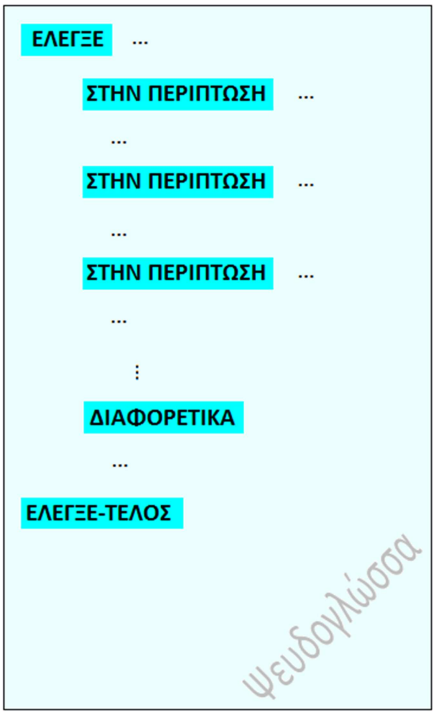

```
1  /* δομή πολλαπλής επιλογής */
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  main()
6  {
7       system("chcp 1253 > nul");
8
9       int w_ar_a = 0, w_ar_b = 0, w_ar_c = 0 ;
10
11      printf("Παρακαλώ δώστε 3 αριθμούς (1-10) \n");
12      scanf("%d %d %d", &w_ar_a, &w_ar_b, &w_ar_c);
13
14      if ( w_ar_a > 5 )
15           {
16           printf("Ο 1ος είναι πάνω από 5 \n");
17           }
18      else
19           {
20           printf("Ο 1ος είναι κάτω από 5 \n");
21           if ( w_ar_b > 5 )
22                {
23                printf("Ο 2ος είναι πάνω από 5 \n");
24                }
25           else
26                {
27                printf("Ο 2ος είναι κάτω από 5 \n");
28                if ( w_ar_c > 5 )
29                     {
30                     printf("Ο 3ος είναι πάνω από 5 \n");
31                     }
32                else
33                     {
34                     printf("Ο 3ος είναι κάτω από 5 \n");
35                     };
36                };
37           };
38
39      system("PAUSE");
41
42 };
```

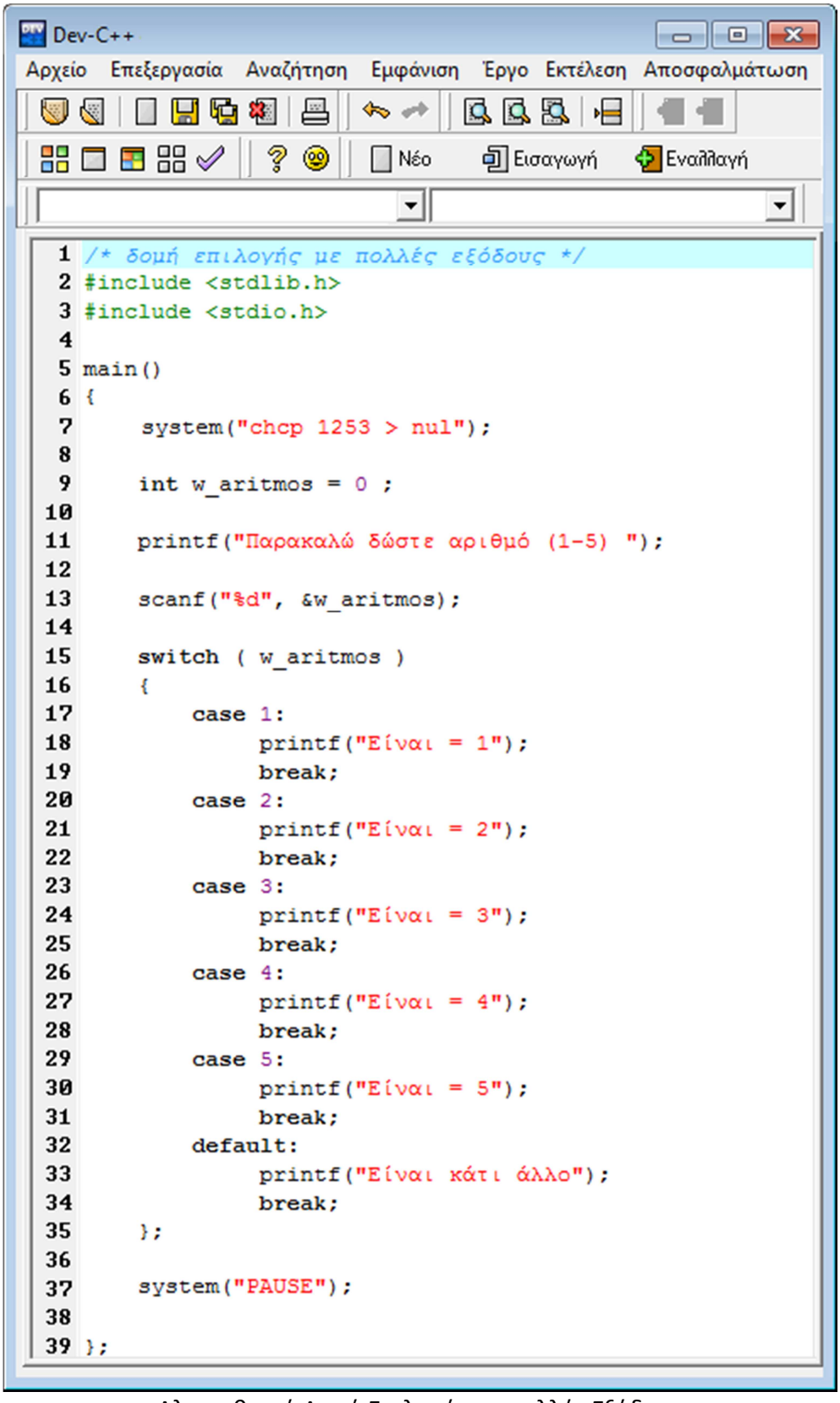
Αλγοριθμική Δομή Πολλαπλής Επιλογής, με προοδευτική στηλοθέτηση

5.3. Αλγοριθμική Δομή Επιλογής με πολλές Εξόδους – switch …

Η αλγοριθμική δομή επιλογής με πολλαπλές εξόδους, λειτουργεί σαν ένας σύνθετος διακόπτης ροής με δυνατότητες πολλαπλών διαφορετικών διαδρομών. Η γλώσσα υλοποιεί τον μηχανισμό με την εντολή **switch**, η οποία έχει την δυνατότητα να κατευθύνει την ροή εκτέλεσης, βάση του περιεχομένου (αριθμητικού ή αλφαριθμητικού) μιας μεταβλητής. Για κάθε διαφορετική περίπτωση περιεχομένου, συντάσσεται μια δήλωση **case** και κάτω από αυτή την δήλωση τοποθετούνται ομάδες εντολών. Κάθε δήλωση **case**, πρέπει υποχρεωτικά να τελειώνει με τον χαρακτήρα **:** (και όχι με το κλασικό ερωτηματικό **;** της C++). Επιπλέον, με την προαιρετική δήλωση **default**, η «μπίλια» ροής μπορεί να κατευθυνθεί σε ένα άλλο κομμάτι κώδικα, για να καλυφθεί και οποιαδήποτε άλλη περίπτωση, όταν δηλαδή δεν θα συμβεί τίποτα από όλα τα προηγούμενα **case**. Επίσης, στο τέλος των εντολών για κάθε δήλωση **case** πρέπει να τοποθετείται η δήλωση τέλους **break**. Εάν η δήλωση **break** απουσιάζει, ο compiler θα προχωρήσει και στην εκτέλεση των εντολών που είναι μέσα και στην επόμενη **case**. Συνεπώς κάθε περιεχόμενο πρέπει οπωσδήποτε να ολοκληρώνεται με ένα **break**. Η εντολή **switch** δεν μπορεί να ελέγξει σύνθετες συνθήκες αλλά μόνο την ισότητα με μια αξία (αριθμητική ή αλφαριθμητική), μέσω των δηλώσεων **case**.








```c
/* δομή επιλογής με πολλές εξόδους */
#include <stdlib.h>
#include <stdio.h>

main()
{
     system("chcp 1253 > nul");

     int w_aritmos = 0 ;

     printf("Παρακαλώ δώστε αριθμό (1-5) ");

     scanf("%d", &w_aritmos);

     switch ( w_aritmos )
     {
          case 1:
               printf("Είναι = 1");
               break;
          case 2:
               printf("Είναι = 2");
               break;
          case 3:
               printf("Είναι = 3");
               break;
          case 4:
               printf("Είναι = 4");
               break;
          case 5:
               printf("Είναι = 5");
               break;
          default:
               printf("Είναι κάτι άλλο");
               break;
     };

     system("PAUSE");

};
```

Αλγοριθμική Δομή Επιλογής με πολλές Εξόδους

Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκαν αναλυτικά οι 3 βασικές αλγοριθμικές δομές επιλογής της γλώσσας προγραμματισμού C++. Το κεφάλαιο περιελάμβανε αναλυτικά παραδείγματα για κάθε δομή και επεξηγήθηκε επίσης η σημασία των ενσωματωμένων δομών επιλογής. Η παρουσίαση της κάθε ομάδας εντολών στην C++, περιελάμβανε το αντίστοιχο λογικό διάγραμμα του αλγορίθμου και τον σχετικό ψευδοκώδικα.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Περιγράψτε, τι σημαίνει ο όρος, *αλγοριθμικές δομές επιλογής*;

Ερώτηση-2: Στην αλγοριθμική δομή επιλογής **if**, περιγράψτε για ποιόν λόγο τοποθετούμε εντολές μέσα σε άγκιστρα **{ }** ;

Ερώτηση-3: Στην αλγοριθμική δομή επιλογής **if**, είναι σωστό ότι, η δήλωση **else**, είναι προαιρετική;

Ερώτηση-4: Στην αλγοριθμική δομή επιλογής **if**, είναι σωστό ότι, μέσα στα όρια μιας δήλωσης **else**, μπορούμε να έχουμε και μια άλλη δήλωση **else**;

Ερώτηση-5: Είναι σωστό ότι, μπορούμε πάντα να αντικαταστήσουμε μια αλγοριθμική δομή επιλογής **switch**, με μια αλγοριθμική δομή επιλογής **if**;

Ερώτηση-6: Είναι σωστό ότι, μπορούμε πάντα να αντικαταστήσουμε μια αλγοριθμική δομή επιλογής **if**, με μια αλγοριθμική δομή επιλογής **switch**;

Ερώτηση-7: Στην αλγοριθμική δομή επιλογής **switch**, για ποιον λόγο χρησιμοποιούμε την δήλωση **break**;

Ερώτηση-8: Στην αλγοριθμική δομή επιλογής **switch**, για ποιον λόγο χρησιμοποιούμε την δήλωση **default**;

Ερώτηση-9: Είναι σωστό ότι, στην αλγοριθμική δομή επιλογής **switch**, μπορούμε να ελέγξουμε και σύνθετες συνθήκες, μέσω των δηλώσεων **case**;

Ερώτηση-10: Είναι λάθος ότι, στην αλγοριθμική δομή επιλογής **if**, μπορούμε να ελέγξουμε και σύνθετες συνθήκες;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Συμβουλευτείτε τις εισαγωγικές παρατηρήσεις του κεφαλαίου.

Απάντηση-2: Συμβουλευτείτε τις παραγράφους 5.1. και 5.2. του κεφαλαίου.

Απάντηση-3: Ναι, είναι σωστό.

Απάντηση-4: Όχι, είναι λάθος.

Απάντηση-5: Ναι, είναι σωστό.

Απάντηση-6: Όχι, είναι λάθος.

Απάντηση-7: Συμβουλευτείτε την ενότητα 5.3. του κεφαλαίου.

Απάντηση-8: Συμβουλευτείτε την ενότητα 5.3. του κεφαλαίου.

Απάντηση-9: Όχι, είναι λάθος.

Απάντηση-10: Όχι, είναι σωστό.

6. ΕΝΑΣ ΑΛΓΟΡΙΘΜΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σκοπός και επιμέρους στόχοι

Ο σκοπός του τελευταίου κεφαλαίου του συγγράμματος είναι να εισάγει τον εκπαιδευόμενο στην διαδικασία της δημιουργίας ενός ολοκληρωμένου προγράμματος στην γλώσσα προγραμματισμού C++. Ο εκπαιδευόμενος λαμβάνοντας συγκεκριμένες κατευθυντήριες οδηγίες και υποβοηθητικές παρατηρήσεις, οδηγείται στην διαδικασία να κατασκευάσει μόνος τους, ένα πλήρες πρόγραμμα το οποίο θα υποστηρίζει την λειτουργία μιας μηχανής.

Προσδοκώμενα αποτελέσματα

Με την μελέτη του τελευταίου κεφαλαίου οι εκπαιδευόμενοι θα μπορούν,

- να αναλύουν με μικρά βήματα το πώς θα πρέπει να καταστρώσουν μια πραγματική εφαρμογή,
- να αντιλαμβάνονται γιατί πρέπει πρώτα να σχεδιάζουν μια εφαρμογή πριν προχωρήσουν στην υλοποίηση της,
- να δημιουργούν καλαίσθητα πλαίσια διεπαφών και να σχεδιάζουν την επικοινωνία ανάμεσα στον υπολογιστή και τον χρήστη,
- να αναζητούν και να αποφασίζουν ποιες μεταβλητές θα πρέπει να χρησιμοποιήσουν στα προγράμματα τους,
- να χρησιμοποιούν αλγοριθμικές δομές επιλογής για την επίλυση προβλημάτων,
- να αναλύουν τα προγράμματα τους σαν μηχανισμούς οι οποίοι προσαρμόζονται κατάλληλα, βάση των εντολών (ή γεγονότων), που παίρνουν από το εξωτερικό περιβάλλον,
- να διαχωρίζουν γραφικά τον κώδικα της διεπαφής από τον υπόλοιπο λειτουργικό κώδικα,
- να χρησιμοποιούν στην πράξη ένα ολοκληρωμένο περιβάλλον ανάπτυξης (I.D.E.) μιας σύγχρονης και δυναμικής γλώσσας προγραμματισμού,
- να μετατρέπουν έναν αλγόριθμο σε εντολές σύνταξης στην γλώσσα προγραμματισμού C++.

Έννοιες – Λέξεις Κλειδιά

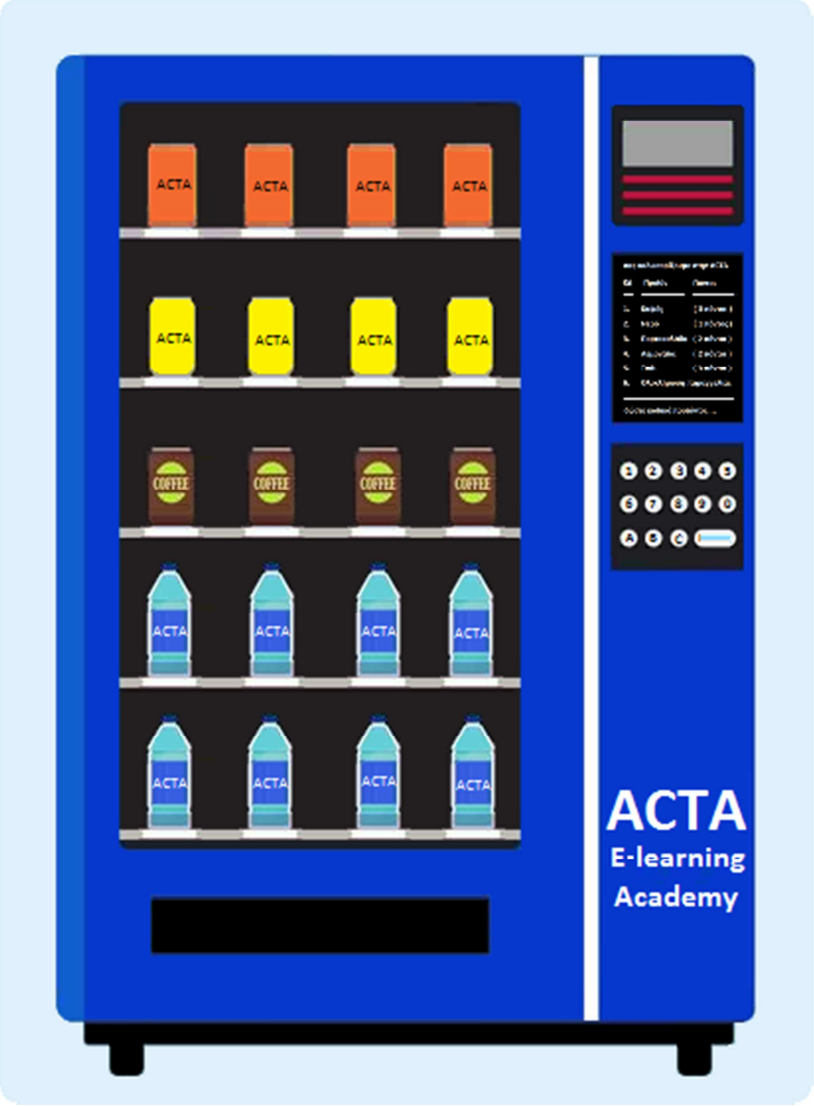
Προσομοίωση, Γραπτές προδιαγραφές, Μενού επιλογών, Όνομα μεταβλητής, Τύπος μεταβλητής, Ενημερωτικό μήνυμα, Διεπαφή, Αλληλεπίδραση, Σύνταξη προγράμματος.

Εισαγωγικές Παρατηρήσεις

Η βιωματική προσέγγιση στην εκπαιδευτική διαδικασία είναι δυναμική επιλογή και μπορεί να αποδώσει θετικά αποτελέσματα για τους εκπαιδευόμενους. Στην βάση αυτού του συλλογισμού, στο συγκεκριμένο κεφάλαιο οι εκπαιδευόμενοι καλούνται να δημιουργήσουν ένα ολοκληρωμένο πρόγραμμα C++, στο περιβάλλον Dev-C++, της Bloodshed.

6.1. Δημιουργία ενός Αλγορίθμου Προγραμματισμού

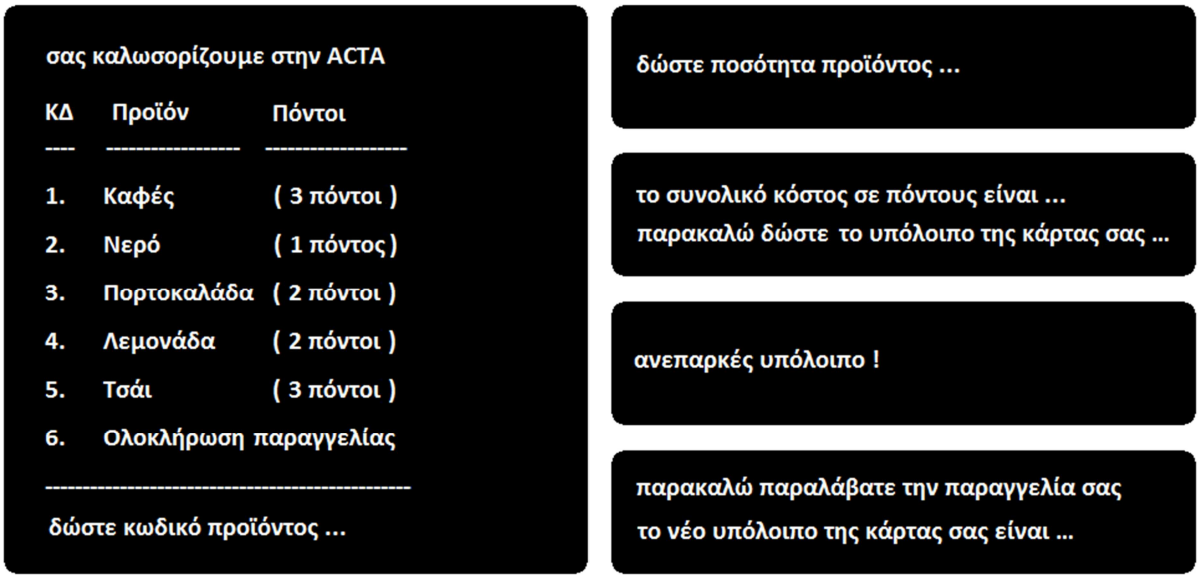
Το πρόγραμμα C++, που θα κατασκευαστεί θα πρέπει να προσομοιώνει την αυτόματη λειτουργία μιας αυτόματης μηχανής, όπως αυτή περιγράφεται, από τις επόμενες εικόνες και τις γραπτές προδιαγραφές.



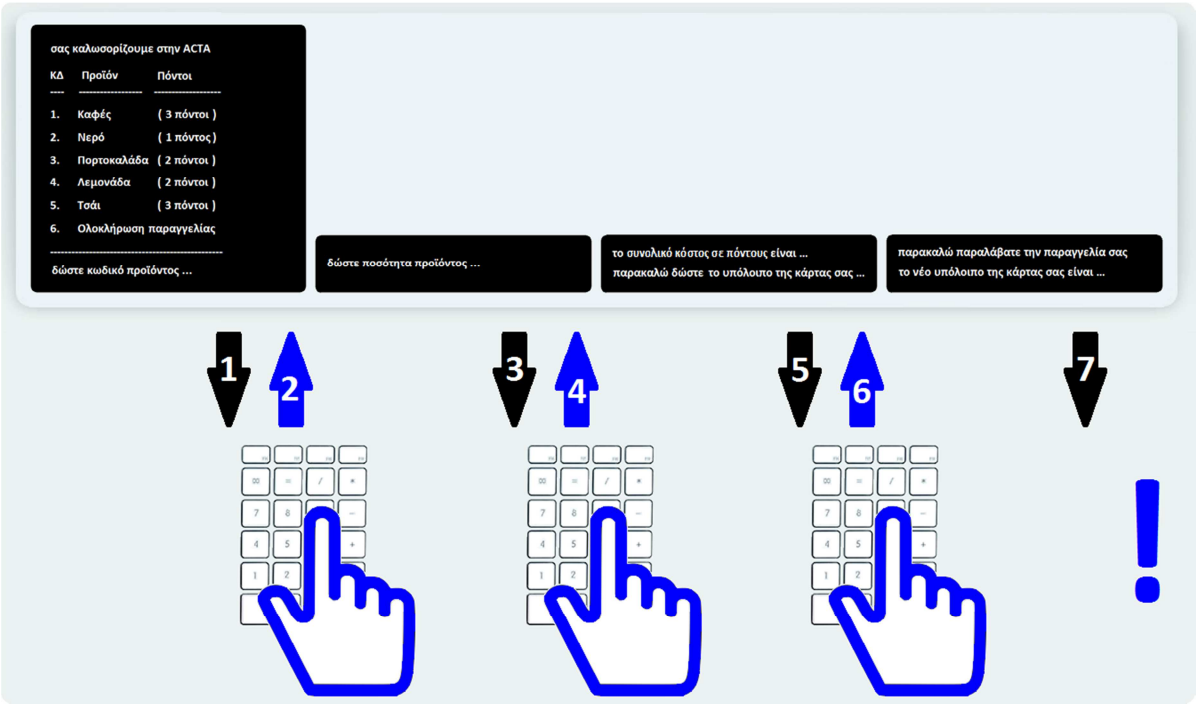
Ειδικότερα, θα πρέπει να κατασκευαστεί ένα πρόγραμμα το οποίο θα υποστηρίζει την λειτουργία ενός μηχανήματος αυτόματης πώλησης προϊόντων. Συγκεκριμένα, υποθέτουμε ότι σε ένα εκπαιδευτικό κέντρο της ACTA, οι εκπαιδευόμενοι έχουν παραλάβει μαζί με το εκπαιδευτικό τους υλικό και μια πλαστική ψηφιακή κάρτα με περιεχόμενο 60 πόντους, τους οποίους μπορούν να καταναλώσουν για να προμηθεύονται δωρεάν, νερό και αναψυκτικά από αυτόματους μηχανικούς πωλητές οι οποίοι βρίσκονται τοποθετημένοι στις εισόδους των χώρων εκπαίδευσης. Το πρόγραμμα θα πρέπει αρχικά, να προβάλει στην οθόνη του μηχανήματος, μενού επιλογών. Να δέχεται δήλωση κωδικού προϊόντος και στην συνέχεια, να ζητά την ποσότητα από το επιλεγόμενο προϊόν. Ακολούθως, να υπολογίζει και να προβάλει στην οθόνη, το συνολικό κόστος της παραγγελίας σε πόντους και να ζητά από τον χρήστη το υπόλοιπο των πόντων της ψηφιακής κάρτας. Εάν το υπόλοιπο των πόντων της κάρτας είναι επαρκές για να καλύψει την παραγγελία, να την εκτελεί, εμφανίζοντας και το νέο υπόλοιπο της κάρτας, διαφορετικά να τυπώνει ενημερωτικό μήνυμα ότι η κάρτα δεν διαθέτει επαρκές υπόλοιπο πόντων. Για λόγους απλούστευσης, η διεπαφή που θα έπρεπε να υλοποιηθεί για να αναγνωρίζεται ηλεκτρονικά η πλαστική κάρτα, μπορεί να αντικατασταθεί μέσω μιας απλής ερώτησης προς τον κάτοχο της κάρτας.

6.2. Δημιουργία της Κατάλληλης Διεπαφής

Βασικό στοιχείο για την υλοποίηση ενός τέτοιου συστήματος, είναι συνήθως η διεπαφή του, καθώς γύρω από αυτήν κινούνται όλες οι υπόλοιπες προγραμματιστικές διεργασίες. Η ζητούμενη διεπαφή περιγράφεται σαν μια συνεχόμενη αλληλεπίδραση ανάμεσα στην μηχανή και τον χρήστη, ο οποίος πληκτρολογεί απαντήσεις, σε συνεχόμενες ερωτήσεις που του υποβάλλει η μηχανή.



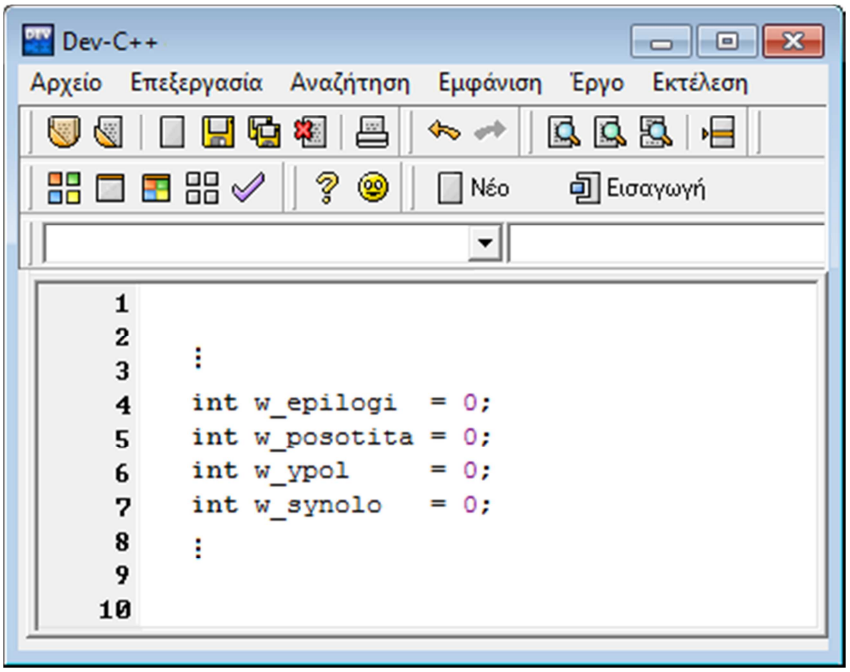
Αυτή η συνεχόμενη αλληλεπίδραση μπορεί να αναπαρασταθεί σχηματικά και από την επόμενη εικόνα. Προφανώς, ο κώδικας C++, της μηχανής θα πρέπει να επεξεργάζεται τις απαντήσεις του χρήστη και να προσαρμόζει κατάλληλα τις επόμενες προγραμματιστικές ενέργειες, βάση των απαντήσεων που λαμβάνει. Αυτός ο τρόπος προσέγγισης της κατάστρωσης ενός αλγορίθμου επίλυσης, γύρω από την διεπαφή του (ή τις διεπαφές του) είναι συνηθισμένη τακτική στον προγραμματισμό των ηλεκτρονικών υπολογιστών και των μηχανών.



Καθώς η ανωτέρω σχηματική απεικόνιση είναι ικανή να αποδώσει μια συνολική εποπτική εικόνα για την λειτουργικότητα της διεπαφής του συγκεκριμένου συστήματος, το αμέσως επόμενο βήμα είναι η κατάστρωση του σχεδίου (ή του λογικού διαγράμματος), με τα σημεία στα οποία θα πρέπει να τοποθετηθούν αυτές καθαυτές οι λειτουργικές εντολές της C++. Οι εντολές αυτές μπορεί να είναι εντολές για προβολή μηνυμάτων (**printf**), εντολές για έλεγχο (**if**), εντολές για υπολογισμούς, κ.λπ.

6.3. Η Δέσμευση των Απαραίτητων Μεταβλητών

Έχοντας καθορίσει από την προηγούμενη ενότητα τα σημεία στα οποία θα πρέπει να μπουν οι λειτουργικές εντολές, θα πρέπει να συνεχίσουμε με τον προσδιορισμό των περιοχών μνήμης, (των μεταβλητών) που χρειαζόμαστε. Αυτό μπορεί να σημαίνει ότι σίγουρα χρειαζόμαστε μεταβλητές για να αποθηκεύουμε στην μνήμη τις απαντήσεις που δίνει ο χρήστης στην μηχανή. Ο χρήστης λοιπόν λαμβάνει από το μηχάνημα τρεις ερωτήσεις, α) ερώτηση για τον κωδικό προϊόντος, β) ερώτηση για την ποσότητα και γ) ερώτηση για το υπόλοιπο των πόντων. Συνεπώς, χρειαζόμαστε μέσα στο πρόγραμμα 3 μεταβλητές για να συγκρατήσουμε αυτά τα δεδομένα. Να γνωρίζει δηλαδή το πρόγραμμα της C++, τον κωδικό προϊόντος, την ποσότητα και τους διαθέσιμους πόντους της κάρτας.



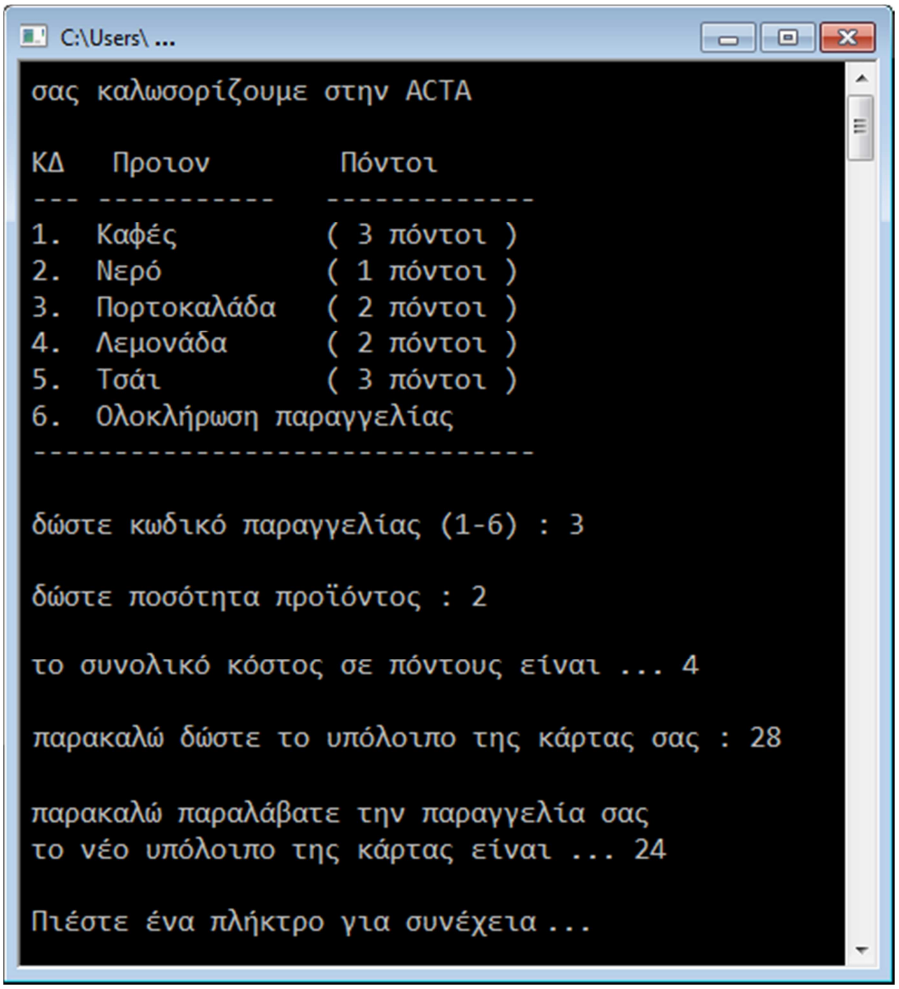
Ακολούθως, πρέπει να προχωρήσουμε και στον καθορισμό περιοχών μνήμης, (των μεταβλητών) που θα χρειαστούν, για να αποθηκεύονται στην μνήμη του υπολογιστή και τα αποτελέσματα των απαραίτητων υπολογισμών. Με βάση τις προδιαγραφές, οι υπολογισμοί αυτοί είναι δυο, α) ο υπολογισμός της ποσότητας (*) επί τους πόντους του προϊόντος και β) το υπόλοιπο της κάρτας (-) μείον, το αποτέλεσμα από τον προηγούμενο υπολογισμό. Συνεπώς, χρειαζόμαστε επιπλέον, μόνο 1 ακόμα μεταβλητή, αφού μπορούμε να χρησιμοποιήσουμε για τον δεύτερο υπολογισμό της αφαίρεσης, την μεταβλητή που δεσμεύσαμε ήδη για να την χρησιμοποιήσουμε για το υπόλοιπο των πόντων της κάρτας. Καθώς όλα τα υποδεχόμενα δεδομένα αλλά και όλοι οι υπολογισμοί δεν παράγουν αποτελέσματα με δεκαδικά ψηφία μπορούμε να προχωρήσουμε στην δέσμευση μεταβλητών ακέραιου τύπου **int**. Εάν ο συγκεκριμένος αυτόματος πωλητής λειτουργούσε με το νόμισμα του ευρώ, αντί για πόντους, τότε οι μεταβλητές οι οποίες συμμετέχουν στους υπολογισμούς θα έπρεπε να δηλωθούν με τύπο **float**.

6.4. Ο Αλγόριθμος με εντολές Προγραμματισμού C++

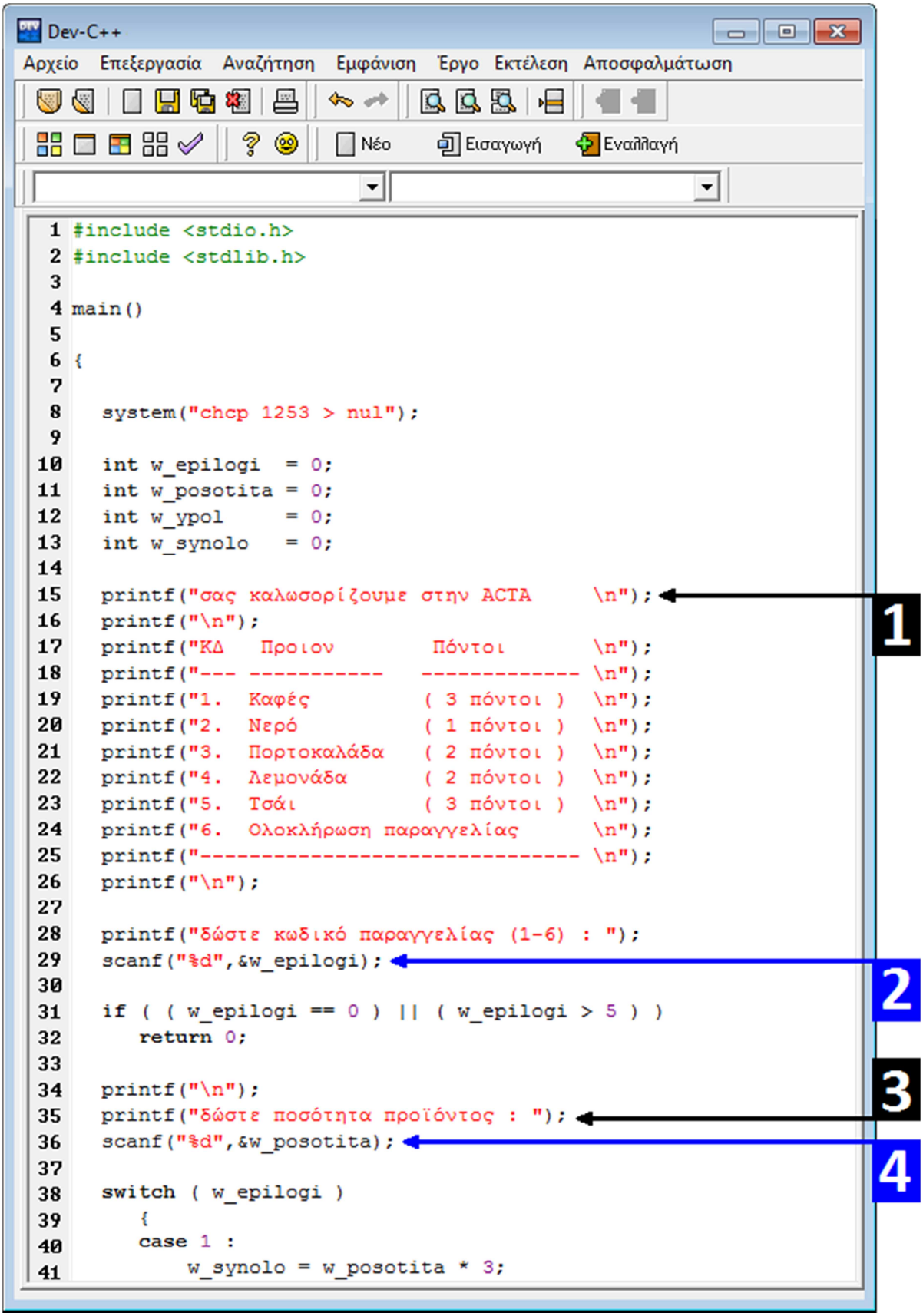
Έχοντας ολοκληρώσει την παρουσίαση των απαραίτητων στοιχείων που θα πρέπει να ληφθούν υπόψη ώστε να υλοποιηθεί ένας κατάλληλος αλγόριθμος υποστήριξης για την υποστήριξη ενός μηχανήματος αυτόματης πώλησης προϊόντων, καλούνται οι εκπαιδευόμενοι να προχωρήσουν στην προσπάθεια σύνταξης ενός σχετικού και πλήρως λειτουργικού προγράμματος C++.

Στις επόμενες σελίδες παρουσιάζεται η ολοκληρωμένη υλοποίηση ενός ενδεικτικού προγράμματος C++. Δίπλα σε κάθε εντολή που αφορά σημείο της διεπαφής του συστήματος, υπάρχει υποβοηθητική αριθμητική παράθεση όπως αυτή καθορίστηκε από την προηγούμενη σχηματική απεικόνιση (βλ. ετικέτες, **_1_**, **_2_**, **_3_**, **_4_**, **_5_**, **_6_** και **_7_**). Εκτός, από τις εντολές (**printf**), για την προβολή των μηνυμάτων, τις εντολές (**scanf**), για την υποδοχή των απαντήσεων, χρησιμοποιούνται επιπλέον, μια εντολή (**switch**), μια εντολή (**if…else…**) και αρκετές εντολές για υπολογισμούς. Ένα λεπτό σημείο βρίσκεται στις γραμμές 31 και 32 όπου ο κώδικας ελέγχει εάν ο χρήστης ζητά να μην προχωρήσει το πρόγραμμα με την επιλογή **6** (ή

κάτι εκτός των διαθέσιμων επιλογών). Στην περίπτωση αυτή, με την εντολή **return 0**, το πρόγραμμα διακόπτει την λειτουργία του, στο σημείο αυτό.

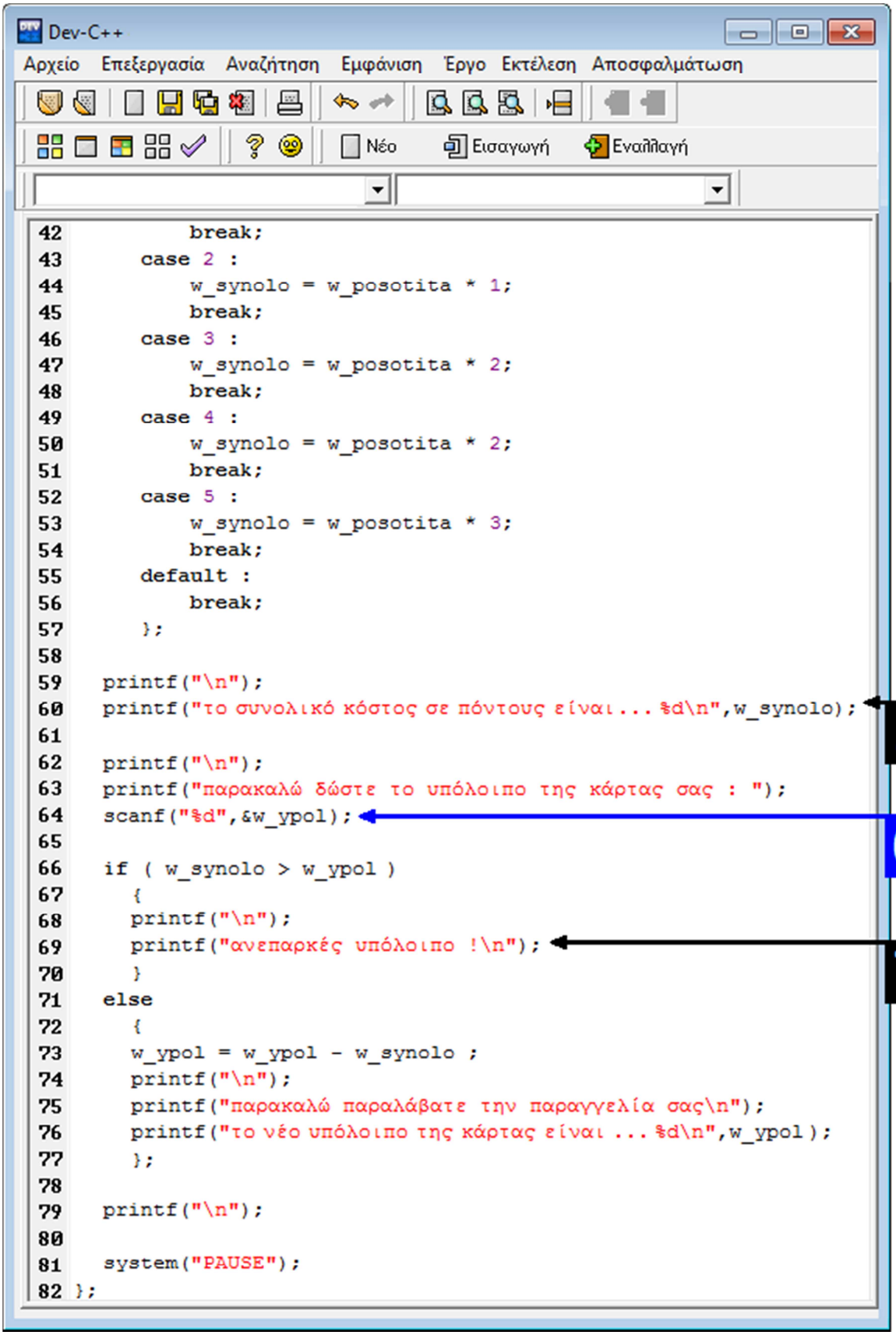


Το αποτέλεσμα της ολοκληρωμένης λειτουργικότητας του προγράμματος.




```
#include <stdio.h>
#include <stdlib.h>

main()

{

   system("chcp 1253 > nul");

   int w_epilogi  = 0;
   int w_posotita = 0;
   int w_ypol     = 0;
   int w_synolo   = 0;

   printf("σας καλωσορίζουμε στην ACTA     \n");
   printf("\n");
   printf("ΚΔ   Προιον         Πόντοι       \n");
   printf("---- -----------    ------------ \n");
   printf("1.   Καφές          ( 3 πόντοι )  \n");
   printf("2.   Νερό           ( 1 πόντοι )  \n");
   printf("3.   Πορτοκαλάδα    ( 2 πόντοι )  \n");
   printf("4.   Λεμονάδα       ( 2 πόντοι )  \n");
   printf("5.   Τσάι           ( 3 πόντοι )  \n");
   printf("6.   Ολοκλήρωση παραγγελίας      \n");
   printf("------------------------------- \n");
   printf("\n");

   printf("δώστε κωδικό παραγγελίας (1-6) : ");
   scanf("%d",&w_epilogi);

   if ( ( w_epilogi == 0 ) || ( w_epilogi > 5 ) )
      return 0;

   printf("\n");
   printf("δώστε ποσότητα προϊόντος : ");
   scanf("%d",&w_posotita);

   switch ( w_epilogi )
      {
      case 1 :
          w_synolo = w_posotita * 3;
```

Το πρόγραμμα C++, υποστήριξης μηχανήματος αυτόματης πώλησης, 1/2



Το πρόγραμμα C++, υποστήριξης μηχανήματος αυτόματης πώλησης, 2/2

Σύνοψη κεφαλαίου

Στο τελευταίο αυτό κεφάλαιο δόθηκε η ευκαιρία στους εκπαιδευόμενος να αναπτύξουν με προσωπική προσπάθεια μια ολοκληρωμένη απλή εφαρμογή με βασικές εντολές προγραμματισμού στην γλώσσα C++. Η ανάπτυξη περιελάμβανε την εφαρμογή στην πράξη, ενός προγράμματος το οποίο περιελάμβανε μια πλήρη διεπαφή, την κατασκευή βασικών αλγοριθμικών δομών επιλογής, καθώς και την δημιουργία εντολών υπολογισμού με χρήση κατάλληλου τύπου μεταβλητών.

Ερωτήσεις αξιολόγησης

Ερώτηση-1: Είναι σωστό, ότι ένα πρόγραμμα έχει πάντα μόνο μια διεπαφή;

Ερώτηση-2: Είναι λάθος, ότι ένα πρόγραμμα μπορεί να έχει πολλές διεπαφές;

Ερώτηση-3: Σε ποια περίπτωση θα χρησιμοποιούσατε τύπο μεταβλητής **int**;

Ερώτηση-4: Σε ποια περίπτωση θα χρησιμοποιούσατε τύπο μεταβλητής **float**;

Ερώτηση-5: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, ποιος είναι ο λόγος ύπαρξης των εντολών στις γραμμές, 26, 34 και 62;

Ερώτηση-6: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, ποιος είναι ο λόγος ύπαρξη της εντολής στην γραμμή, 32;

Ερώτηση-7: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, ποιος είναι ο λόγος ύπαρξη της εντολής στην γραμμή, 81;

Ερώτηση-8: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, στις γραμμές, 28, 35 και 63, υπάρχουν εντολές **printf**, οι οποίες στο τέλος τους δεν έχουν την χαρακτήρα **\n**, για ποιον λόγο συμβαίνει αυτό;

Ερώτηση-9: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, εάν αφαιρέσουμε την εντολή **break** από την γραμμή, 48, τι θα συμβεί στο πρόγραμμα, στοχαστείτε και περιγράψτε, τι θα συμβεί;

Ερώτηση-10: Στην ολοκληρωμένη υλοποίηση του ενδεικτικού προγράμματος C++, εάν αφαιρέσουμε την εντολή **break** από την γραμμή, 51, τι θα συμβεί στο πρόγραμμα, στοχαστείτε και περιγράψτε, τι θα συμβεί;

Απαντήσεις ερωτήσεων αξιολόγησης

Απάντηση-1: Όχι, είναι λάθος.

Απάντηση-2: Όχι, είναι σωστό.

Απάντηση-3: Συμβουλευτείτε την ενότητα 6.3. του κεφαλαίου.

Απάντηση-4: Συμβουλευτείτε την ενότητα 6.3. του κεφαλαίου.

Απάντηση-5: Για λόγους κομψότητας, για να αφήνει μια κενή γραμμή.

Απάντηση-6: Εάν ο έλεγχος μπει μέσα στο **if**, με την εντολή **return 0**, το πρόγραμμα διακόπτει την λειτουργία του.

Απάντηση-7: Για να σταματά τα πρόγραμμα και να μπορούμε να δούμε και το τελευταίο μήνυμα.

Απάντηση-8: Ο λόγος που γίνεται αυτό, είναι για να «κολλά» δίπλα στο μήνυμα της **printf**., η μεταβλητή με την **scanf**, που ακολουθεί.

Απάντηση-9: Η «μπίλια» θα πέσει και στο επόμενο **case : 4**, αλλά δεν θα δημιουργήσει πρόβλημα καθώς και η επιλογή 3 και η επιλογή 4, έχουν ίδιο ποσό σε πόντους και άρα δεν θα δημιουργηθεί λογικό λάθος.

Απάντηση-10: Η «μπίλια» θα πέσει και στο επόμενο **case : 5**, και θα δημιουργηθεί λογικό λάθος, καθώς η επιλογή 3 έχει κόστος 2 πόντους, ενώ η επιλογή 4, έχει 3 πόντους. Ο χρήστης δηλαδή, θα ζητά λεμονάδα και θα χρεώνεται με τσάι.

Γλωσσάριο σημαντικών όρων

bit

Byte

Preprocessing directive

Έλεγχος συνθήκης

Έξοδος αλγορίθμου

Όνομα μεταβλητής

Ακέραιος

Ακολουθία ενεργειών

Ακολουθίες διαφυγής

Αλγοριθμικές δομές επανάληψης

Αλγοριθμικές δομές επιλογής

Αλληλεπίδραση

Αλφάβητο γλώσσας

Αλφαριθμητικά

Αντικειμενοστραφής προγραμματισμός

Απλή ακολουθία

Απόφαση

Απόφαση σε συνθήκη

Αριθμητικοί τελεστές

Αρχεία επικεφαλίδων

Αρχικοποίηση μεταβλητής

Βιβλιοθήκες

Γραμματική γλώσσας

Γραπτές προδιαγραφές

Δήλωση τροποποιητή

Δεδομένα

Δεσμευμένες λέξεις

Διάγραμμα ροής

Διαδικασιακός προγραμματισμός

Διαδίκτυο

Διακόπτης

Διεπαφή

Δομές πολλαπλών επιλογών

Δομή

Δομή επανάληψης υπό συνθήκη

Δομή επιλογής

Δομημένος προγραμματισμός

Είσοδος αλγορίθμου

Εγκατάσταση γλώσσας

Εκτέλεση κώδικα

Εκτελέσιμος κώδικας

Εμβέλεια μεταβλητής

Εμφωλιασμένες δομές επιλογής

Ενημερωτικό μήνυμα

Εντολή

Εντολή if

Εντολή if ... else ...

Εντολή switch

Εξωτερική δομή επιλογής

Επίπεδα προτεραιότητας

Επεξεργασία δεδομένων

Επεξεργαστής κειμένου

Εργαλεία ανάπτυξης λογισμικού

Εσωτερική δομή επιλογής

Εύρος τιμών

Η επιστήμη της Πληροφορικής

Ηλεκτρονικός υπολογιστής Η/Υ

Θεμελιώδεις τύποι μεταβλητών

Καθορισμός ζητουμένων εξόδου

Καθοριστές τύπου

Κατάλληλη ονοματοδοσία

Κλασματικό μέρος

Κοινωνία της πληροφορίας

Κυβερνοχώρος

Κρίση λογισμικού

Κώδικας σε γλώσσα προγραμματισμού

Λέξεις κλειδιά

Λίστα ορισμάτων

Λεξιλόγιο γλώσσας

Λογικοί τελεστές

Λογικό διάγραμμα

Λογικός διακόπτης

Λογισμικό

Μέγεθος μεταβλητής

Μεθοδολογία ανάπτυξης λογισμικού

Μενού επιλογών

Μεταβλητές

Μεταγλώττιση κώδικα

Μεταφερσιμότητα

Μεταφραστής

Μορφοποιημένη προβολή δεδομένων

Μορφοποιημένη υποδοχή δεδομένων

Μορφοποιητές

Νήματα εκτέλεσης

Ολοκληρωμένο περιβάλλον ανάπτυξης

Ομοιάζουσες γλώσσες

Ορίσματα εισόδου

Ορίσματα εξόδου

Πίνακας κωδικοποίησης ASCII

Παγκόσμιος ψηφιακός μετασχηματισμός

Πεδίο

Περιγραφικά ονόματα μεταβλητών

Περιεχόμενο μεταβλητής

Περιοχή μνήμης

Πηγαίος κώδικας

Πληροφορίες

Πραγματικός αριθμός

Προ-μεταφραστής

Προγραμματισμός υπολογιστών

Προγραμματιστές

Προδιαγραφή

Προσδιοριστές

Προσεταιριστικότητα

Προσομοίωση

Προτεραιότητα

Προτεραιότητες τελεστών

Πρόγραμμα ηλεκτρονικού υπολογιστή

Πρότυπο C++20

Ροή εισόδου

Ροή εξόδου

Σημασιολογία γλώσσας

Σταθερές

Στηλοθέτηση κώδικα

Συγκριτικοί τελεστές

Συναρτήσεις

Συντακτικό γλώσσας

Σχεδίαση

Σχόλια

Σύνταξη προγράμματος

Τελεστέοι

Τελεστές

Τελεστής

Τελεστής διεύθυνσης

Τελεσταίος

Τεχνολογίες αιχμής

Τεχνολογία λογισμικού

Τροποποιητές τύπου

Τύπος μεταβλητής

Χαρακτήρας

Ψευδογλώσσα

Ψευδοκώδικας

Aristotle Certification Training & Assessment
Τεχνοβλαστός Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης

Βιβλιογραφία

Διαδικαστικός Προγραμματισμός (η γλώσσα C), Πάρις Μαστοροκώστας, Σ.Ε.Α.Β., 2015.

Beginning C: From Novice to Professional, Ivor Horton, 4th. ed, APRESS, 2006.

C PROGRAMMING: A Modern Approach, K.N.King, 2nd. ed, W.W.NORTON, 2008.

Programming language C++, ISO/IEC JTC1 SC22 WG21 N4860, I.S.O./I.E.C., 2020.

The Art of Computer Programming, Donald Knuth, Addison-Wesley, 1968.

THE C PROGRAMMING LANGUAGE, Brian W. Kernighan, Dennis M. Ritchie, Prentice-Hall International, Bell Labs, 1978.

THE C++ PROGRAMMING LANGUAGE, Bjarne Stroustrup, 3rd. ed, AT & T Labs, Addison-Wesley, 1997.

The development of the C programming language, Dennis M. Ritchie, in History of Programming Languages JJ, Addison-Wesley, 1996.

Οδηγίες για περαιτέρω μελέτη

Η εκμάθηση μιας γλώσσας προγραμματισμού και η υιοθέτηση κατάλληλων πρακτικών και μεθοδολογιών είναι μια σύνθετη διαδικασία η οποία αποκτάται κυρίως μέσα από την εμπειρία και το διαθέσιμο περιβάλλον εργασίας (εκπαιδευτικό, επαγγελματικό, κ.λπ.). Παρά την σημαντική πρόοδο που έχει επιτευχθεί, η δημιουργία των προγραμμάτων παρουσιάζει σημαντικά προβλήματα που σχετίζονται με την ποιότητα και την επάρκεια των εφαρμογών που κατασκευάζονται. Για την επιστήμη της Πληροφορικής, τον κλάδο της τεχνολογίας λογισμικού και τις γλώσσες προγραμματισμού, τα προβλήματα αυτά περιγράφονται με τον όρο, *κρίση λογισμικού* (software crisis).

Στα πλαίσια των προσπαθειών για περαιτέρω ενασχόληση και βελτίωση, καλούνται οι εκπαιδευόμενοι, μετά από την ολοκλήρωση της μελέτης του συγγράμματος,

- *Προγραμματισμός ΙΙ (ενδιάμεσο επίπεδο), Κατανοώντας τον Προγραμματισμό,*

και χρησιμοποιώντας το εγκατεστημένο, στον υπολογιστή τους ολοκληρωμένο περιβάλλον Dev-C++, της BloodShed, να συνεχίσουν με την δημιουργία, αποσφαλμάτωση και εκτέλεση των προγραμμάτων C++, τα οποία υπάρχουν σαν παραδείγματα στο πέμπτο κεφάλαιο του συγγράμματος.

Ακολούθως, μπορούν να συνεχίσουν την μελέτη τους και με το τελευταίο σύγγραμμα της σειράς, *Προγραμματισμός Ηλεκτρονικών Υπολογιστών & Μηχανών, της ACTA,*

- *Προγραμματισμός ΙΙΙ (προχωρημένο επίπεδο), Αξιοποιώντας τον Προγραμματισμό.*