ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Σιβιτανίδειος

ΤΕΧΝΙΚΟΣ ΗΥ

Πίνακας περιεχομένων

Η Γλώσσα Προγραμματισμού C
(Μέρος 1 - Εισαγωγή)

Κεφ. 1.1 Σύντομη Ιστορία της C

Η γλώσσα προγραμματισμού C δημιουργήθηκε από τον Dennis Ritchie στα Bell Labs στο διάστημα από το 1969 και 1972 όταν αυτός και ο Ken Thompson ασχολούνταν με τον σχεδιασμό του λειτουργικού συστήματος Unix. Είναι διαδικαστική γλώσσα προγραμματισμού το οποίο σημαίνει ότι έχει δυνατότητες δομημένου προγραμματισμού και επιτρέπει τη χρήση υποπρογραμμάτων. Είναι γλώσσα υψηλού επιπέδου και μπορεί να λειτουργήσει σε διάφορες αρχιτεκτονικές υπολογιστών. Για να γίνεται αυτό είναι απαραίτητο να γίνεται μεταγλώττιση από τη γλώσσα προγραμματισμού στην εκάστοτε γλώσσα μηχανής. Η λειτουργία του αυτή εκτελείται από τον μεταγλωττιστή (compiler). Ένα από τα χαρακτηριστικά της γλώσσας C είναι ότι μπορεί να χρησιμοποιηθεί και ως γλώσσα προγραμματισμού χαμηλού επιπέδου, επιτρέποντας άμεση πρόσβαση στους πόρους του υπολογιστή.

Αποτελεί μια από τις πιο ευρύτερα χρησιμοποιούμενες γλώσσες προγραμματισμού και πολλές νεότερες γλώσσες έχουν επηρεαστεί από αυτήν. Ανάμεσα στις γλώσσες που έχουν επηρεαστεί είναι η C++, η Java, η Perl, η JavaScript, η PHP, η Python καθώς και του κελύφους C του Unix.

Ανάμεσα στα πλεονεκτήματα της C είναι:

- ✓ Διαθέτει ευελιξία
- ✓ Έχει απλό συντακτικό
- ✓ Διαθέτει φορητότητα και ο κώδικας της μπορεί να μεταγλωττιστεί από διάφορους μεταγλωττιστές
- ✓ Είναι αρθρωτή
- ✓ Είναι σχετικά μικρή και εύκολη στην εκμάθηση

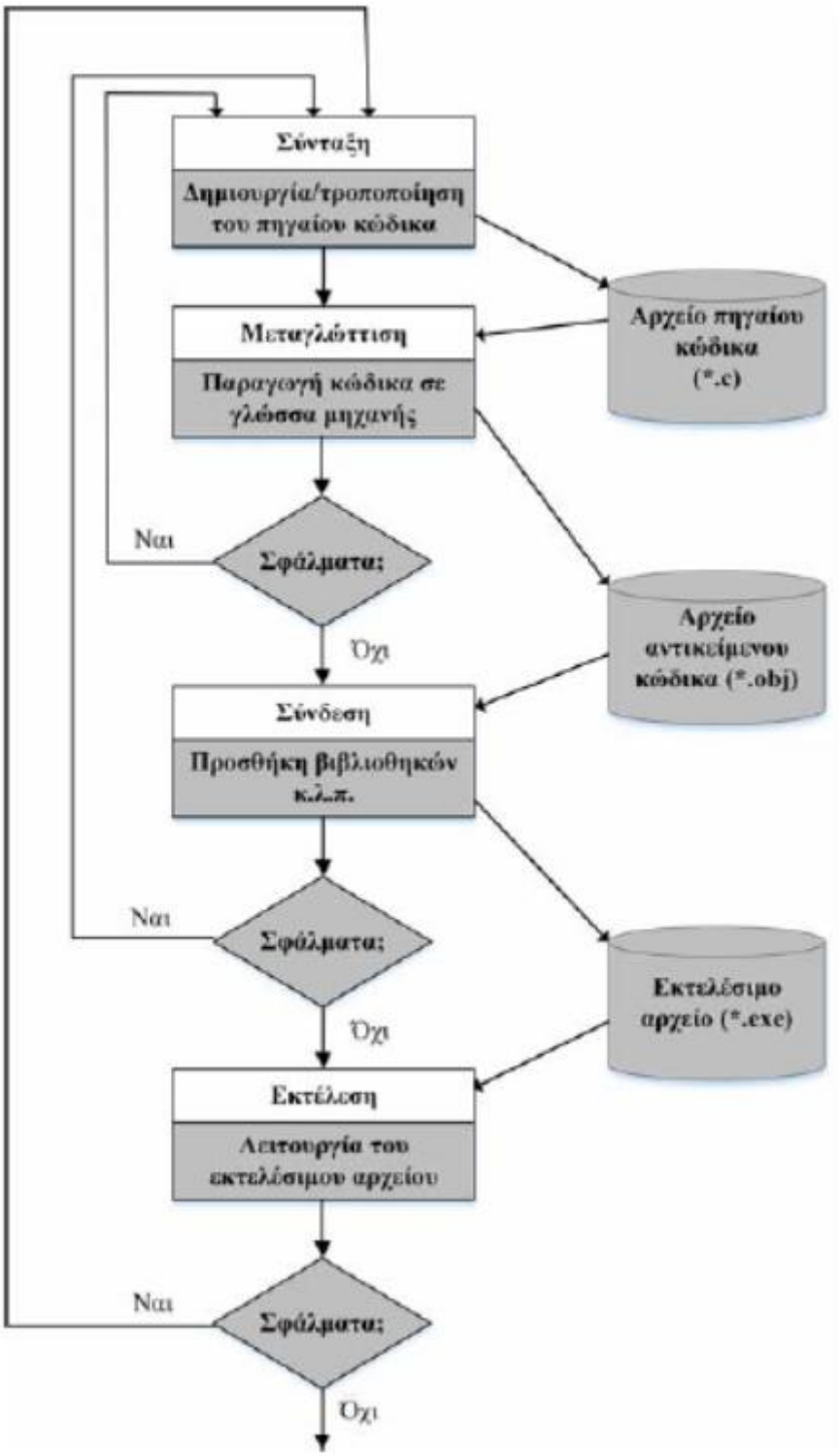
Κεφ. 1.2 Στάδια ανάπτυξης προγράμματος

Ένα πρόγραμμα αποτελείται από μια πεπερασμένη σειρά από εντολές. Η κάθε εντολή εκτελεί μια συγκεκριμένη λειτουργία σε κάθε γλώσσα προγραμματισμού. Αρχικά χρησιμοποιείται ένας συντάκτης κειμένου στον οποίο ο προγραμματιστής γράφει τον πηγαίο κώδικα (source code). Είναι δυνατόν να χρησιμοποιηθεί ο ενσωματωμένος συντάκτης κειμένου της γλώσσας C αλλά μπορεί να χρησιμοποιηθεί και οποιοσδήποτε συντάκτης κειμένου (notepad, word κλπ). Ο προγραμματιστής δημιουργεί ένα αρχείο το οποίο έχει κατάληξη .c.

Στην συνέχεια το πρόγραμμα πρέπει να περάσει από την φάση της μεταγλώττισης η οποία εκτελείται από τον μεταγλωττιστή. Ο μεταγλωττιστής παράγει ένα αρχείο το οποίο ονομάζεται αρχείο αντικειμένου (object file) το οποίο περιέχει τον κώδικα σε γλώσσα μηχανής και έχει κατάληξη .obj. Κατά την φάση της μεταγλώττισης, ανιχνεύονται τα συντακτικά σφάλματα που έχει κάνει ο προγραμματιστής ο οποίος έχει παραβιάσει τους συντακτικούς κανόνες της γλώσσας. Τα λάθη αυτά πρέπει να διορθωθούν για να εκτελεστεί το πρόγραμμα. Μόνο όταν διορθωθούν τα λάθη δημιουργείται το αρχείο αντικειμένου.

Στο επόμενο στάδιο, γίνεται η σύνδεση με την χρήση του συνδέτη ο οποίος αναζητά σε βιβλιοθήκες τις συναρτήσεις που χρησιμοποίησε ο προγραμματιστής. Ο συνδέτης παράγει ένα εκτελέσιμο αρχείο το οποίο έχει επέκταση .exe. Για να παραχθεί το αρχείο, θα πρέπει να ξεπεραστούν προβλήματα όπως για παράδειγμα η αδυναμία εντοπισμού μιας εξωτερικής μεταβλητής ή μιας συνάρτησης. Μόλις παραχθεί το .exe αρχείο, το πρόγραμμα μπορεί να εκτελεστεί στον υπολογιστή.

Στο τελικό στάδιο, εκτελείται το αρχείο .exe και γίνεται ο λογικός έλεγχος του προγράμματος. Δηλαδή, ελέγχεται αν το αποτέλεσμα του αλγορίθμου επιλύει σωστά το πρόβλημα για το οποίο φτιάχτηκε. Επίσης, ανιχνεύονται τα σημασιολογικά σφάλματα, με πιο χαρακτηριστικό παράδειγμα την διαίρεση ενός αριθμού με τον μηδέν, σφάλμα το οποίο δεν ανιχνεύεται κατά την μεταγλώττιση. Εφόσον, ο προγραμματιστής ανιχνεύσει κάποιο σφάλμα, τότε επιστρέφει στο πρώτο στάδιο και επαναλαμβάνει όλα τα στάδια. Η διαδικασία αυτή επαναλαμβάνεται μέχρι το πρόγραμμα να λειτουργήσει σωστά.


Σύνταξη
Δημιουργία/τροποποίηση του πηγαίου κώδικα
Αρχείο πηγαίου κώδικα (*.c)
Μεταγλώττιση
Παραγωγή κώδικα σε γλώσσα μηχανής
Ναι
Σφάλματα;
Όχι
Αρχείο αντικείμενου κώδικα (*.obj)
Σύνδεση
Προσθήκη βιβλιοθηκών κ.λ.π.
Ναι
Σφάλματα;
Όχι
Εκτελέσιμο αρχείο (*.exe)
Εκτέλεση
Λειτουργία του εκτελέσιμου αρχείου
Ναι
Σφάλματα;
Όχι

Κεφ. 2.Προγραμματισμός στην C

Κεφ. 2.1.1 Το πρώτο πρόγραμμα στη C

Το πιο απλό πρόγραμμα που μπορούμε να γράψουμε στην C, είναι το πρόγραμμα που δείχνει το μήνυμα “Hello World”. Ο πηγαίος κώδικας εμφανίζεται στις παρακάτω γραμμές.

```
1.  //Hello World C πρόγραμμα
2.  #include <stdio.h>
3.  int main()
4.  {
5.      printf(“Hello world\n”);
6.      return 0;
7.  }
```

- Γραμμή 1η: Περιέχει ένα σχόλιο του προγράμματος. Τα σχόλια μας βοηθάνε στην κατανόηση ενός προγράμματος και δεν επηρεάζουν καθόλου την εκτέλεση του. Τα σχόλια μπαίνουν μετά τα σύμβολα //. Σχόλια μπορούν να γραφούν και σε περισσότερες από μια γραμμές και είναι το κείμενο ανάμεσα σε /* και */.
- Γραμμή 2η: Γίνεται εισαγωγή του αρχείου επικεφαλίδας stdio.h (Standard Input Output Header). Το αρχείο περιλαμβάνει τις βασικές συναρτήσεις εισόδου (π.χ. πληκτρολόγιο) και εξόδου (π.χ. οθόνη). Για να μπορέσουμε να καλέσουμε την συνάρτηση printf() πρέπει να συμπεριλάβουμε το αρχείο stdio.h. Όλα τα αρχεία κεφαλίδας δηλώνονται πριν την main(), με την χρήση της οδηγίας include. Η γενική της μορφή είναι:

  #include <όνομα_αρχείου_κεφαλίδας.h>

  Η c περιέχει πολλές βιβλιοθήκες αλλά στο κάθε πρόγραμμα φροντίζουμε και εισάγουμε μόνο όσες βιβλιοθήκες χρειαζόμαστε.
- Γραμμή 3η: Η καρδιά κάθε προγράμματος είναι η συνάρτηση main(). Η συνάρτηση main() εκτελείται πάντα πρώτη και χρησιμοποιείται για να δείξει το σημείο έναρξης εκτέλεσης του προγράμματος. Μετά τη main() χρησιμοποιείται το αριστερό άγκιστρο ({) και μετά ακολουθούν οι εντολές του προγράμματος. Η main() τερματίζει την εκτέλεση της με το δεξί άγκιστρο (}). Η συνάρτηση πρέπει να επιστρέφει ένα αποτέλεσμα το οποίο είναι ένας ακέραιος αριθμός επειδή βάλαμε int main().
- Γραμμή 5η : Καλείται η συνάρτηση printf() η οποία τυπώνει το μήνυμα το οποίο το οποίο έχουμε γράψει. Η συνάρτηση printf() δέχεται ορίσματα τα οποία περικλείονται από τις παρενθέσεις. Το σύμβολο “\n” ανήκει στις ακολουθίες διαφυγής και σημαίνει «Πήγαινε στην επόμενη γραμμή». Ο χαρακτήρας '\' (backslash) στο τέλος των μηνυμάτων χρησιμοποιείται για να ειδοποιήσει τον μεταφραστή ότι οι χαρακτήρες που ακολουθούν δεν είναι εκτυπώσιμοι αλλά αποτελούν χαρακτήρες ελέγχου. Έτσι ο χαρακτήρας 'n' σημαίνει νέα γραμμή (new line, carriage return/line feed) με αποτέλεσμα ο κέρσορας της οθόνης να μετακινείται μια γραμμή προς τα κάτω

- Γραμμή 6η : Τερματίζεται η main(), και επιστρέφεται το 0 (μηδέν) που είναι ακέραιος αριθμός (int).

ΠΑΡΑΤΗΡΗΣΕΙΣ:

- Μετά από κάθε εντολή τοποθετείται ένα ελληνικό ερωτηματικό στο τέλος το οποίο δείχνει τον τερματισμό της εντολής.
- Η C είναι Case Sensitive γλώσσα, το οποίο σημαίνει ότι κάνει διάκριση ανάμεσα στα μικρά και τα κεφαλαία γράμματα. Η καλύτερη τακτική είναι να γράφουμε τον κώδικα πάντα με μικρά γράμματα.

Κεφ. 2.1.2 Σταθερές - Μεταβλητές

Για την αναπαράσταση των προσωρινών αποτελεσμάτων και των δεδομένων χρησιμοποιούνται οι σταθερές και οι μεταβλητές σε ένα πρόγραμμα. Και οι σταθερές και οι μεταβλητές χρησιμοποιούν κάποιο χώρο μνήμης στον οποίο αποθηκεύονται τα δεδομένα κατά την εκτέλεση του προγράμματος. Όταν κάποια δεδομένα δεν αλλάζουν ποτέ κατά τη διάρκεια της εκτέλεσης ενός προγράμματος τότε χρησιμοποιούνται οι σταθερές. Π.χ. όταν πρόκειται να χρησιμοποιήσουμε την τιμή του π στη γεωμετρία που είναι 3.14159, τότε αντί να χρησιμοποιούμε τον αριθμό κάθε φορά μπορούμε να χρησιμοποιήσουμε ένα όνομα σταθεράς (π.χ. το π). Οι σταθερές ορίζονται με την εντολή

const, π.χ. const int number=10;

Με την εντολή αυτή, αντί να χρησιμοποιούμε την ακέραια τιμή 10, μπορούμε να χρησιμοποιούμε το όνομα number. Μπορούμε να ορίσουμε μια σταθερά, χρησιμοποιώντας την οδηγία define η οποία αντιστοιχίζει ένα όνομα σε κάποια τιμή.

Π.χ. #define TRUE 1

σημαίνει ότι όταν χρησιμοποιηθεί στο πρόγραμμα η λέξη TRUE τότε στη θέση της μπαίνει η τιμή 1. Η define ορίζεται εκτός της main() και δεν χρειάζεται ερωτηματικό στο τέλος. Π.χ. σταθερές μπορούν να οριστούν με τους εξής τρόπους:

```
1. #Include <stdio.h>
2. // Δήλωση σταθερής τιμής με define
3. #define FPA 0.24
4. int main()
5. {
6.     //Δήλωση σταθερής τιμής με const
7.     const FPA = 0.21;
8. }
```

Αντίθετα, οι τιμές των μεταβλητών μπορούν να αλλάξουν κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Δεν μπορεί να αλλάξει ο τύπος των μεταβλητών. Η δήλωση των μεταβλητών γίνεται στην αρχή του προγράμματος. Οι μεταβλητές δηλώνονται στην αρχή των προγραμμάτων με εντολές του στυλ:

int num1, num2;

Η εντολή αυτή σημαίνει ότι υπάρχουν 2 ακέραιες μεταβλητές με τα ονόματα num1 και num2.

Κεφ. 2.1.2.1 Ονομασία Μεταβλητών

Για την ονοματοδοσία των μεταβλητών ακολουθούνται οι εξής κανόνες:

- ✓ Ένα όνομα μεταβλητής μπορεί να περιέχει τα γράμματα του αλφαβήτου, τα ψηφία 0-9 και τον χαρακτήρα υπογράμμισης _
- ✓ Το όνομα πρέπει να ξεκινά με γράμμα ή με χαρακτήρα υπογράμμισης
- ✓ Δεν μπορούμε να χρησιμοποιήσουμε δεσμευμένη λέξη

Συμβουλή: Καλό είναι το όνομα της μεταβλητής να είναι ενδεικτικό της ιδιότητας της.

Κεφ. 2.1.2.2 Τύποι Μεταβλητών

Οι μεταβλητές της C ανήκουν σε 2 κατηγορίες, τους βαθμωτούς και τους συναθροιστικούς τύπους. Στους βαθμωτούς τύπους περιλαμβάνει:

- Τους **ακέραιους** που δηλώνονται με την λέξη int. Η δήλωση μιας ακέραιας μεταβλητής γίνεται με την εντολή:

  int number;

  Κατά την δήλωση της μεταβλητής υπάρχει δυνατότητα η μεταβλητή να αρχικοποιηθεί σε κάποια τιμή. Π.χ.

  int number=10;

  Για έναν υπολογιστή που διαθέτει 4 bytes για κάθε ακέραιο, το σύνολο των δυνατών τιμών που μπορεί να πάρει μια ακέραια μεταβλητή είναι 2^{32}. Στο πρότυπο C9 έχει προστεθεί ο τύπος long long int, στον οποίο οι μεταβλητές έχουν χώρο 64 bits. Επιπλέον, υπάρχει ο προσδιοριστής unsigned int για να χαρακτηρίσει μια μεταβλητή χωρίς πρόσημο.

  Για να εισάγουμε (με την συνάρτηση scanf()) ή να εμφανίσουμε (με την συνάρτηση printf()) μια ακέραια μεταβλητή χρησιμοποιούμε τον προσδιοριστή %d.

  Π.χ. για να διαβάσουμε μια ακέραια μεταβλητή, χρησιμοποιούμε την εντολή:

  scanf("%d", &number)

  Για την εμφάνιση μιας ακέραιας μεταβλητής, εκτός από την δεκαδική εμφάνιση της μπορούμε να χρησιμοποιήσουμε τους προσδιοριστές %o για εμφάνιση σε οκταδική μορφή ή %h για εμφάνιση σε δεκαεξαδική μορφή. Οι προσδιοριστές l (long), h (short), και u (unsigned) τοποθετούνται πριν από τους d, o, x.

  Τους **πραγματικούς**, οι οποίοι χωρίζονται στους αριθμούς διπλής ακρίβειας με κωδική λέξη double και στους αριθμούς κινητής υποδιαστολής που δηλώνονται με την λέξη float. Οι πραγματικοί αριθμοί διαθέτουν κλασματικό μέρος. Για να εισάγουμε (με την συνάρτηση scanf()) ή να εμφανίσουμε (με την συνάρτηση printf()) μια μεταβλητή κινητής υποδιαστολής χρησιμοποιούμε τον προσδιοριστή %f. Εάν πρόκειται να αναγνωσθεί αριθμός διπλής ακριβείας, απαιτείται η χρήση του προσδιοριστή %lf, ενώ στην ανάγνωση αριθμού εκτεταμένης ακριβείας ο προσδιοριστής %f αντικαθίσταται από τον προσδιοριστή %Lf.

- Τους **χαρακτήρες** που δηλώνονται με την λέξη char. Η δήλωση γίνεται ως εξής:

  char letter;

  ενώ όταν πρόκειται να δηλώσουμε κάποια τιμή στην μεταβλητή η τιμή μπαίνει μέσα σε εισαγωγικά, π.χ

  letter = 'AB'

  Για την εκτύπωση (printf()) ή το διάβασμα (scanf()) κάποιας τιμής χαρακτήρα χρησιμοποιείται ο προσδιοριστής %c. Π.χ. η εντολή

  printf("Το γράμμα είναι %c\n", letter);

  θα εκτυπώσει το μήνυμα Το γράμμα είναι και το περιεχόμενο της μεταβλητής letter. Για την εκτύπωση χαρακτήρων μπορεί να χρησιμοποιηθεί ο προσδιοριστής %d, επειδή κάθε χαρακτήρας του κώδικα ASCII αντιστοιχεί σε ένα οκταψήφιο δυαδικό αριθμό.
- Τους δείκτες
- Τον απαριθμητικό τύπο που δηλώνεται με την λέξη enum

Στην κατηγορία των συναθροιστικών τύπων ανήκουν οι πίνακες, οι δομές (struct) και οι ενώσεις (union).

Κεφ. 2.1.3 Λέξεις Κλειδιά

Η C χρησιμοποιεί ένα πλήθος από λέξεις οι οποίες υλοποιούν συγκεκριμένες λειτουργίες. Οι λέξεις αυτές είναι:

auto	double	int	struct	break	else	long
switch	case	enum	register	typedef	char	extern
return	union	const	float	short	unsigned	void
default	goto	sizeof	volatile	do	if	static
while						

Κεφ. 2.1.4 Είσοδος / Έξοδος Δεδομένων

Όλα τα προγράμματα πρέπει να δέχονται κάποια είσοδο ενώ παράγουν κάποια αποτελέσματα. Στην C, ο τύπος του στοιχείου προς εισαγωγή ή εμφάνιση δηλώνετε εξ' αρχής, μέσα στα εισαγωγικά. Οι ακόλουθοι συμβολισμοί χρησιμοποιούνται για τους αντίστοιχους τύπους δεδομένων:

Τύπος Μεταβλητής	**Συμβολισμός**
int	%d
char	%c
float	%f
double	%lf
string	%s

Κεφ. 2.1.3.1 Είσοδος Δεδομένων

Για την είσοδο δεδομένων χρησιμοποιείται η συνάρτηση scanf() και το πρότυπο της συνάρτησης υπάρχει στο αρχείο κεφαλίδας stdio.h. Με την εντολή:

scanf("%d", &number);

το πρόγραμμα θα διαβάσει μια ακέραια μεταβλητή την οποία θα την αποθηκεύσει στην number. Μπροστά από το όνομα της μεταβλητής υπάρχει το σύμβολο & το οποίο δηλώνει στον μεταγλωττιστή ότι πρόκειται για όνομα μεταβλητής και όχι θέση μνήμης. Με την scanf() υπάρχει η δυνατότητα να χρησιμοποιήσουμε τον προσδιοριστή πλάτους: Π.χ. στην εντολή

scanf("%4d", &number);

ακόμα και αν ο χρήστης πληκτρολογήσει τον αριθμό 123456789, στην μεταβλητής number θα αποθηκευτούν τα 4 πρώτα ψηφία, δηλαδή η μεταβλητή number θα έχει περιεχόμενο την τιμή 1234. Άλλη δυνατότητα της scanf() είναι να θέσει περιορισμούς στους χαρακτήρες που μπορεί να διαβάσει. Αυτό γίνεται με την χρήση του προσδιοριστή []. Π.χ. με τον προσδιοριστή %[A-D] επιτρέπει την καταχώρηση των χαρακτήρων 'A', 'B', 'C', 'D' ενώ ο προσδιοριστή %[^A-D] δεν επιτρέπει την ανάγνωση των χαρακτήρων αυτών. Είσοδος δεδομένων μπορεί να γίνει και με την συνάρτηση getch().

Κεφ. 2.1.3.2 Έξοδος Δεδομένων

Για την εμφάνιση αποτελεσμάτων ενός προγράμματος στη C χρησιμοποιείται η συνάρτηση printf() η οποία υπάρχει στο αρχείο κεφαλίδας stdio.h. Με την printf() μπορούν να χρησιμοποιηθούν κάποιοι μη εκτυπώσιμοι χαρακτήρες οι οποίοι επιτελούν κάποια λειτουργία. Οι μη εκτυπώσιμοι χαρακτήρες και οι αντίστοιχες ακολουθίες διαφυγής εμφανίζονται στον παρακάτω πίνακα:

Χαρακτήρας	Ακολουθία	Χαρακτήρας	Ακολουθία
Νέα γραμμή	\n	Μονό εισαγωγικό	\'
Οριζόντιος στηλοθέτης	\t	Διπλό εισαγωγικό	\''
Κατακόρυφος στηλοθέτης	\v	Πλάγια γραμμή	\\
Αλλαγή σελίδας	\f	Λατινικό ερωτηματικό	\?
Επαναφορά κεφαλής	\r	Οκταδικός αριθμός	\ooo
Οπισθοχώρηση μια θέση	\b	16δικός αριθμός	\xhhh
Συναγερμός	\a	Μηδενικός χαρακτήρας	\0

Ένα επιπλέον εργαλείο μορφοποίησης αποτελούν οι σημαίες (flags).

Σημαία	Λειτουργία
-	Αριστερή στοίχιση της εξόδου στο πεδίο πλάτους
+	Προσθήκη του προσήμου στις θετικές τιμές
#o	Προσθήκη του 0 μπροστά από οκταδικούς αριθμούς
#x	Προσθήκη του 0x μπροστά από δεκαεξαδικούς αριθμούς
#X	Προσθήκη του 0X μπροστά από δεκαεξαδικούς αριθμούς
0	Προσθήκη των απαιτούμενων μηδενικών μπροστά από την τιμή ώστε να καλυφθεί το πεδίο του πλάτους
Κενός χαρακτήρας	Προσθήκη του κενού χαρακτήρα μπροστά από τις μηδενικές τιμές

Εμφάνιση χαρακτήρων μπορεί να γίνει και με την συνάρτηση putchar().

Κεφ. 2.1.4 Τελεστές

Οι τελεστές χωρίζονται σε 3 κατηγορίες ανάλογα με τις μεταβλητές που μπορούν να χειριστούν ταυτόχρονα. Υπάρχουν οι μοναδιαίοι τελεστές (unary), οι δυαδικοί (binary), και οι τριαδικοί (ternary). Μια επιπλέον κατηγοριοποίηση είναι με βάση τη λειτουργεία που επιτελούν:

- **Αριθμητικοί τελεστές:** Οι αριθμητικοί τελεστές είναι δυαδικοί (άρα χρειάζονται 2 τελεστές για να λειτουργήσουν σωστά) και περιγράφονται στον παρακάτω πίνακα:

Τελεστής	Σύμβολο
Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/
Υπόλοιπο διαίρεσης ακεραίων	%

- **Σχεσιακοί τελεστές:** Οι σχεσιακοί τελεστές είναι και αυτοί δυαδικοί. Το αποτέλεσμα τους είναι είτε αληθές (1) είτε ψευδές (0) και περιγράφονται στον παρακάτω πίνακα.

Τελεστής	Σύμβολο
Ίσο	==
Διάφορο	!=
Μεγαλύτερο	>
Μεγαλύτερο ή ίσο	>=
Μικρότερο	<
Μικρότερο ή ίσο	<=

-

- **Λογικοί τελεστές:** Περιγράφονται στον παρακάτω πίνακα:

Λογική Πράξη	Σύμβολο
Λογική σύζευξη (και)	&&
Λογική διάζευξη (ή)	||
Λογική άρνηση (Όχι)	!

- **Τελεστές Ανάθεσης:** Χρησιμοποιούνται ως απλοποιημένες πράξεις και σκοπός τους είναι η πιο γρήγορη εκτέλεση πράξεων.

Πράξη	Σύμβολο	Παράδειγμα
x=x+2	+=	x+=2
x=x-2	-=	x-=2
x=x*2	*=	x*=2
x=x/2	/=	x/=2
x=x%2	%=	x%=2

- **Αυξητικοί και Αφαιρετικοί τελεστές:** Οι τελεστές αυτοί είναι μοναδιαίοι και η λειτουργία τους είναι η αύξηση ή η μείωση του τελεσταίου κατά μία μονάδα. Υπάρχουν δύο τρόποι χρήσης, είτε προθεματικά (prefix) είτε επιθεματικά (postfix) του τελεστέου. Η προθεματική χρήση δηλώνει την αύξηση ή μείωση του τελεσταίου πριν την αποτίμηση της έκφρασης ενώ αντίθετα η επιθεματική αφού εκτελεστεί.

Πράξη	Σύμβολο	Παράδειγμα (y=4)
x=x+1	++	x=y++; // x=4, y=5
		x=++y; //x=5,y=5
x=x-1	--	x=y--; // x=4, y=3
		x=--y; //x=3,y=3

Π.χ. έστω

X=10, y=15	x=10	y=15
++x;	x=11	y=15
y=--x;	x=10	y=5
y=x-- + y;	x=9	y=15
y = y – x--;	x=8	y=6

- **Τελεστές Διαχείρισης δυαδικών ψηφίων:** Οι τελεστές αυτοί επιτρέπουν την εκτέλεση πράξεων σε επίπεδο bit.

Τελεστής	Δράση
&	Λογικό AND
|	Λογικό OR
~	Συμπλήρωμα ως προς 1
^	Exclusive OR
<<	Ολίσθηση προς τα αριστερά
>>	Ολίσθηση προς τα δεξιά

Το αποτέλεσμα των συμβόλων αυτών φαίνεται στον παρακάτω πίνακα:

X	Y	X & Y	X | Y	X ^ Y	~X
Αληθές	Αληθές	Αληθές	Αληθές	Ψευδές	Ψευδές
Αληθές	Ψευδές	Ψευδές	Αληθές	Αληθές	Ψευδές
Ψευδές	Αληθές	Ψευδές	Αληθές	Αληθές	Αληθές
Ψευδές	Ψευδές	Ψευδές	Ψευδές	Ψευδές	Αληθές

Η ολίσθηση προς τα αριστερά μετατοπίζει όλα τα bits του αριστερού τελεστέου κατά τόσες θέσεις προς τα αριστερά όση είναι η τιμή του δεξιού τελεστέου και τα υπόλοιπα bits στο δεξί τμήμα του τελεστέου συμπληρώνονται με μηδενικά. Ουσιαστικά η ολίσθηση προς τα αριστερά πολλαπλασιάζει την τιμή του αριστερού τελεστέου κατά 2^n, όπου n είναι ο αριθμός των θέσεων ολίσθησης.

Η ολίσθηση προς τα δεξιά μετατοπίζει όλα τα bits του αριστερού τελεστέου κατά τόσες θέσεις προς τα δεξιά όση είναι η τιμή του δεξιού τελεστέου και τα υπόλοιπα bits στο αριστερό τμήμα του τελεστέου συμπληρώνονται με μηδενικά. Ουσιαστικά η ολίσθηση προς τα αριστερά διαιρεί την τιμή του αριστερού τελεστέου κατά 2^n, όπου n είναι ο αριθμός των θέσεων ολίσθησης.

Κεφ. 2.1.5 Μετατροπές τύπων

Είναι πολύ πιθανό ένας τελεστής να έχει τελεστέους διαφορετικού τύπου δεδομένων. Για να εκτελεστεί η λειτουργία του τελεστή, οι τελεστέοι μετατρέπονται σε ενιαίο τύπο είτε αυτόματα από τον υπολογιστή (έμμεση μετατροπή) είτε άμεσα από τον προγραμματιστή (ρητή μετατροπή).

Στις έμμεσες μετατροπές, ο τύπος χαμηλότερης ιεραρχίας μετατρέπεται στον τύπο υψηλότερης ιεραρχίας χωρίς να υπάρχει απώλεια πληροφορίας. Η κατάταξη των τύπων γίνεται ακολουθώντας τον εξής κανόνα:

char → int → long → float → double

Άρα σε μια πρόσθεση ενός int τελεστέου με έναν float τελεστή, ο int τελεστής μετατρέπεται σε float και εκτελείται η πρόσθεση.

Στις ρητές μετατροπές, ο προγραμματιστής χρησιμοποιεί τον τελεστή μετατροπής τύπου ή cast τελεστής και έχει την μορφή (τύπος δεδομένων) π.χ. (float). Ο cast τελεστής είναι μοναδιαίος.

Κεφ. 2.1.6 Εκφράσεις

Οι τελεστές χρησιμοποιούνται για την δημιουργία εκφράσεων. Κάθε έκφραση έχει μια τιμή και αποτελείται από έναν ή περισσότερους τελεστές μαζί με έναν ή περισσότερους τελεστέους. Για να γίνει ο υπολογισμός μιας έκφρασης, οι τελεστές ταξινομούνται σε επίπεδα προτεραιότητας. Επιπλέον, η ύπαρξη περισσότερων τελεστών στο ίδιο επίπεδο προτεραιότητας επιβάλλει τον προσδιορισμό της κατεύθυνσης εφαρμογής, με την κατεύθυνση από τα αριστερά προς τα δεξιά να είναι ευρύτερα χρησιμοποιούμενη. Ένας τελεστής καλείται αριστερής προσεταιριστικότητας, όταν σε εκφράσεις που περιέχουν πολλά στιγμιότυπα του τελεστή η ομαδοποίηση γίνεται από τα αριστερά προς τα δεξιά. Π.χ., όλοι οι αριθμητικοί τελεστές είναι τελεστές αριστερής προσεταιριστικότητας. Το αποτέλεσμα είναι ο υπολογισμός 8-4-2 να υπολογίζεται ως (8-4)-2. Αντίθετα ο τελεστής ανάθεσης και η ύψωση σε δύναμη είναι τελεστές δεξιάς προσεταιριστικότητας. Έτσι στην έκφραση counter1=counter2=5, εφαρμόζεται πρώτα ο δεξιός τελεστής ανάθεσης με αποτέλεσμα η μεταβλητή counter2 να αποκτά την τιμή 5 και στη συνέχεια η μεταβλητή counter1 να παίρνει την τιμή της μεταβλητής counter2. Η προτεραιότητα και το είδος προσεταιριστικότητας των τελεστών εμφανίζεται στον παρακάτω πίνακα:

Τελεστές (με φθίνουσα σειρά προτεραιτότητας)	**Είδος προσεταιριστικότητας**
() [] ->	Από αριστερά προς τα δεξιά
! ~ ++ -- &	Από δεξιά προς τα αριστερά
Sizeof()	
* / %	Από αριστερά προς τα δεξιά
+ -	
<< >>	
< <= > >=	
== ! =	
&	
^	
|	
&&	
||	
?:	Από δεξιά προς τα αριστερά
= += -= *= %= &= ^= |=	
<<= >>=	Από αριστερά προς τα δεξιά

Κεφ. 3.Δομή Ελέγχου Ροής

Κεφ. 3.1 Η δομή ελέγχου if

Τα προγράμματα είναι πιθανό να χρειάζεται να αλλάξουν την ροή τους παίρνοντας αποφάσεις μέσω ελέγχου κάποιας συνθήκης. Η εντολή if ελέγχει την ορθότητα μιας συνθήκης. Αν η συνθήκη είναι αληθής τότε εκτελείται το αμέσως επόμενο σύνολο εντολών, διαφορετικά το πρόγραμμα προσπερνάει μια εντολή και συνεχίζει με την επόμενη.
Π.χ. στην εντολή

```
if (x<0)
    x +=1;
```

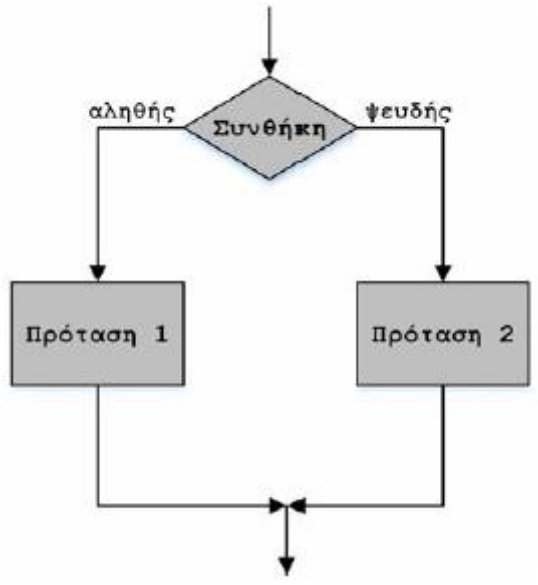
Στο παράδειγμα αυτό ελέγχεται η τιμή του x. Αν το x είναι αρνητικός, τότε εκτελείται η εντολή x+=1. Αν το x είναι θετικός, τότε η εντολή αυτή θα προσπεραστεί και δεν θα εκτελεστεί. Στην περίπτωση που πρέπει να εκτελεστούν περισσότερες από μια εντολές τότε χρησιμοποιούμε τα {}. Π.χ.

```
if (x<0) {
    scanf("%d", &x);
    printf("x= %d \n",x);
    x +=1;
}
```

Η προηγούμενη μορφή της if είναι η πιο απλή μορφή της. Πολύ συχνά, η if χρησιμοποιείται στην γενική της μορφή που είναι:

```
if (συνθήκη)
{
    Πρόταση 1;
}
else
{
    Πρόταση 2;
}
```

Σε μια τέτοια δομή, το διάγραμμα ροής είναι:



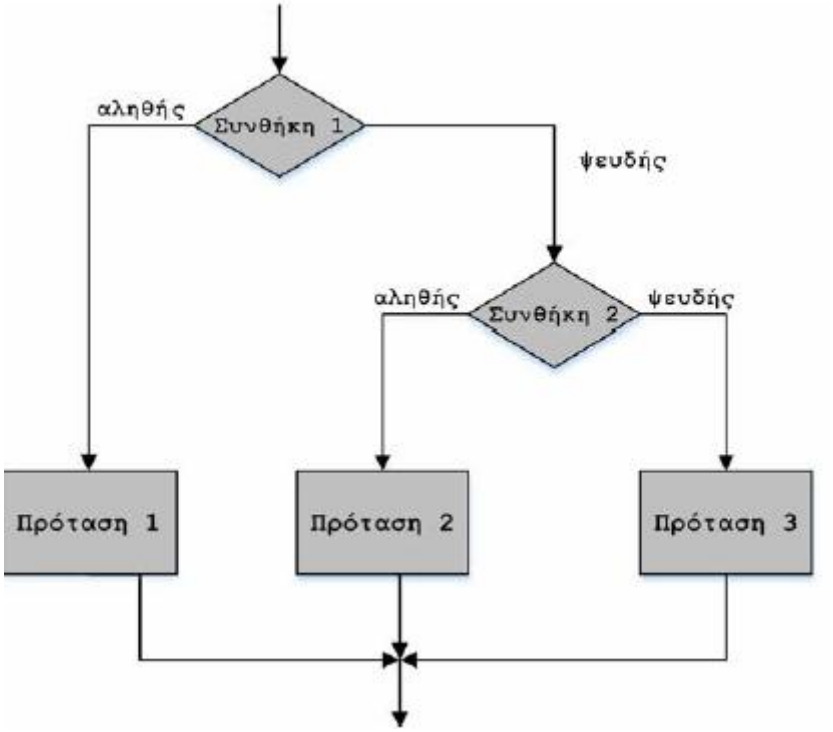
Η if ελέγχει την συνθήκη. Αν η συνθήκη είναι αληθής, τότε εκτελείται η ομάδα εντολών 1. Αν η συνθήκη είναι ψευδής, θα εκτελεστεί η ομάδα εντολών 2. Στην περίπτωση

που κάποια ομάδα εντολών αποτελείται από 1 εντολή, τότε τα άγκιστρα είναι περιττά. Ωστόσο, καλό είναι να τοποθετούνται πάντα.

Η ομάδα εντολών 2 μπορεί να περιέχει και άλλη δομή if σε περίπτωση που πρέπει να ελεγχθεί κάποια συνθήκη. Σε μια τέτοια περίπτωση η δομή if έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
        Πρόταση 1;
}
else if (συνθήκη 2)
{
        Πρόταση 2;
}
else
{
        Πρόταση 3;
}
```

Στην δομή αυτή, ελέγχεται η συνθήκη. Αν είναι αληθής τότε εκτελείται η Πρόταση 1. Αν είναι ψευδής, τότε ελέγχεται η συνθήκη 2. Αν είναι Αληθής, τότε εκτελείται η Πρόταση 2 διαφορετικά η Πρόταση 3. Σε αυτή την περίπτωση, το διάγραμμα ροής παίρνει την μορφή:



Κεφ. 3.2 Υποθετικός τελεστής

Ο υποθετικός τελεστής αποτελείται από 2 σύμβολα (? :). Είναι ο μοναδικός τριαδικός τελεστής στην C και η σύνταξη του είναι η ακόλουθη:

Έκφραση 1 ? έκφραση 2 : έκφραση 3

Ουσιαστικά η τιμή της έκφρασης αυτής είναι η έκφραση 2 αν η έκφραση 1 είναι αληθής, διαφορετικά έχει την τιμή της έκφρασης 3. Ουσιαστικά, η έκφραση 1 είναι συνθήκη ελέγχου. Π.χ.

(x > z) ? x : z

Η έκφραση αυτή έχει την τιμή x αν η έκφραση x > z είναι αληθής, διαφορετικά έχει την τιμή z.

Ουσιαστικά είναι μια σύντμηση της if, αλλά εφόσον είναι και ο ίδιος expression έχουμε την δυνατότητα να τον χρησιμοποιήσουμε με ακόμη περισσότερη ευχέρεια.

Παράδειγμα

```
#include<stdio.h>
int main()
{
        int x = 5, y = 10, max;
        max = (x > y) ? x : y;
        // στην μεταβλητή max θα εκχωρηθεί το y
        //αφού είναι μεγαλύτερο του x
        printf("The maximum number is %d.\n", max);
}
```

Κεφ. 3.3 Δομή επιλογής switch

Όπως η if έτσι και η switch μας επιτρέπει την διακλάδωση του προγράμματος σε διαφορετικά block κώδικα ανάλογα με την περίπτωση που ισχύει. Επειδή όμως οι περιπτώσεις της switch είναι σταθερές και όχι λογικές εκφράσεις, μπορεί να ισχύει μόνο μια περίπτωση, και άρα να εκτελεστεί μόνο ένα τμήμα κώδικα.

Η switch συντάσσεται ως εξής:

Σύνταξη

```
switch (μεταβλητή){
    case σταθερά1:  εντολή_Α;
            break;
    case σταθερά2:  εντολή_Β;
            break;
    case σταθερά3:  εντολή_Γ;
            break;
    default:    εντολή_Δ;
}
```

Παράδειγμα

```
switch (x){
    case 1: x++;
        break;
    case 2: x--;
        break;
    case 3: x+=x;
        break;
    default:printf("%d",x);
}
```

Όπως φαίνεται και στο παράδειγμα, αν το x έχει π.χ. την τιμή 2 τότε εκτελείται η εντολή x--; και επειδή μετά βάλαμε την δεσμευμένη λέξη **break**; η ροή του προγράμματος διακόπτεται και συνεχίζει μετά το τέλος της switch. Αν η **break** δεν υπήρχε τότε ο κώδικας θα εκτελούσε την εντολή και μετά όλες τις παρακάτω, δηλαδή αν αντιστοιχούσε στην 2, τότε θα εκτελούνταν και η 3 και η default. Σημειώστε ότι η default εκτελείται αν δεν προκύψει εγκυρότητα κάποιας άλλης περίπτωσης.

Παράδειγμα

```c
#include<stdio.h>
int main()
{
	int day;
	printf("Give me a number from 1 to 7: ");
	scanf("%d", &day);
	switch (day) {
	case 1: printf("The first day of a week is Monday.\n");    break;
	case 2: printf("The second day of a week is Tuesday.\n");  break;
	case 3:	printf("The third day of a week is Wednesday.\n"); break;
	case 4:	printf("The fourth day of a week is Thursday.\n"); break;
	case 5:	printf("The fifth day of a week is Friday.\n");    break;
	case 6:	printf("The sixth day of a week is Saturday.\n");  break;
	case 7:	printf("The seventh day of a week is Sunday.\n");  break;
	default: printf("You gave a wrong number.\n");
	}
	return 0;
}
```