Παραδείγματα:

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

```
metavliti=5;
printf(" %d", metavliti);
```

5

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

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

The value of variable is: 5

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

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

The value of variable is:

5

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

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

5 + 3 = 8

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

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

x = 5

y = 3

x + y = 8

Τι είναι Αλγόριθμος

Οι οδηγίες που δίνουμε με λογική σειρά, ώστε να εκτελέσουμε μια εργασία ή να επιλύσουμε ένα πρόβλημα, συνθέτουν έναν **Αλγόριθμο**. Για παράδειγμα, οι οδηγίες για την κατασκευή ενός χαρταετού μπορεί να αποτελέσουν έναν αλγόριθμο. ***Αλγόριθμο ονομάζουμε τη σαφή και ακριβή περιγραφή μιας σειράς ξεχωριστών οδηγιών - βημάτων, με σκοπό την επίλυση ενός προβλήματος.***

Αλγόριθμος μπορεί να είναι μια συνταγή μαγειρικής ή η βήμα προς βήμα περιγραφή της λύσης ενός μαθηματικού προβλήματος. Όταν σχεδιάζουμε έναν αλγόριθμο, πρέπει να είμαστε ιδιαίτερα προσεκτικοί, ώστε να βάζουμε με **λογική σειρά** τις **οδηγίες (instructions)** που θα μας οδηγήσουν στη λύση του προβλήματός μας. Αν, για παράδειγμα, δεν περιγράψουμε σωστά τα βήματα που πρέπει να ακολουθηθούν, ώστε να μαγειρέψει ένας άπειρος μάγειρας μια μακαρονάδα, τότε είναι πιθανό να μείνουμε νηστικοί.

Ιδιότητες ενός Αλγορίθμου

Τα βήματα που αποτελούν έναν αλγόριθμο ονομάζονται **οδηγίες** ή **εντολές**. Πρώτα απ' όλα, πρέπει να είμαστε σίγουροι ότι, αν υλοποιήσουμε τον αλγόριθμο, **κάποτε θα τελειώσει** επιτυγχάνοντας τον αρχικό σκοπό. Οι εντολές ενός αλγορίθμου πρέπει να έχουν **ακρίβεια** και **σαφήνεια**, ώστε να μην μπερδευτεί αυτός που θα υλοποιήσει τον αλγόριθμο και τις εκτελέσει με λανθασμένο τρόπο. Τέλος, οι εντολές ενός αλγορίθμου πρέπει να είναι **εκφρασμένες με απλά λόγια**, ώστε να είναι απόλυτα κατανοητές.

Οπότε οι ιδιότητες είναι οι ακόλουθες:

1. ***Αποτελεσματικότητα (στο τέλος πρέπει να προκύπτει ένα αποτέλεσμα, ένα έργο).***
2. ***Περατότητα (κάποτε θα τελειώσει)***
3. ***Καθοριστικότητα (οι εντολές πρέπει να έχουν ακρίβεια και σαφήνεια)***
4. ***Απλότητα (οι εντολές πρέπει να είναι εκφρασμένες με απλά λόγια)***

Βασικά χαρακτηριστικά γλωσσών προγραμματισμού

Όπως και οι φυσικές γλώσσες, έτσι και κάθε γλώσσα προγραμματισμού έχει ως βασικά χαρακτηριστικά:

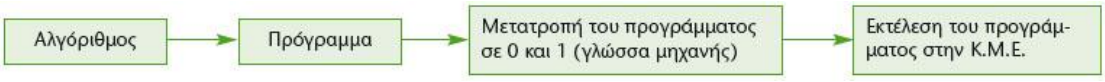
- το **αλφάβητο**,
- το **λεξιλόγιο** και
- το **συντακτικό**.

Το **αλφάβητο** μιας γλώσσας προγραμματισμού είναι το σύνολο των χαρακτήρων που χρησιμοποιούνται από τη γλώσσα.
Το **λεξιλόγιο** μιας γλώσσας είναι το σύνολο των λέξεων που αναγνωρίζει η γλώσσα και έχουν συγκεκριμένη και μοναδική σημασία. Στις γλώσσες προγραμματισμού το λεξιλόγιο είναι πολύ περιορισμένο (μερικές δεκάδες λέξεις), ώστε να μπορούμε να το μάθουμε εύκολα.
Το **συντακτικό** μιας γλώσσας προγραμματισμού είναι το σύνολο των κανόνων που πρέπει να ακολουθούμε, για να συνδέουμε λέξεις σε προτάσεις. Σε μια γλώσσα προγραμματισμού η σύνδεση λέξεων δημιουργεί ολοκληρωμένες εντολές προς τον υπολογιστή. Αν δεν ακολουθήσουμε αυστηρά το συντακτικό μιας γλώσσας, είναι αδύνατο για τον υπολογιστή να καταλάβει ποια εντολή του δίνουμε.

- Αν σε κάποια οδηγία έχουμε κάνει λάθος στο αλφάβητο, στο λεξιλόγιο ή στο συντακτικό τότε το πρόγραμμα που μετατρέπει τις οδηγίες μας σε σειρά από 0 και 1 θα μας δώσει ένα κατάλληλο μήνυμα λάθους, ώστε να μας βοηθήσει να διορθώσουμε το λάθος μας. Τα λάθη αυτά ονομάζονται **συντακτικά λάθη**.
- Τα προγράμματα που μετατρέπουν τις οδηγίες μας σε 0 και 1 μπορούν να χωριστούν σε δύο κατηγορίες:
- Στους μεταγλωττιστές και
- στους διερμηνείς.

Η διαφορά τους είναι ότι οι μεταγλωττιστές (compilers) θα ελέγξουν όλο το πρόγραμμα για συντακτικά λάθη και μετά θα το μετατρέψουν όλο σε μια κατάλληλη σειρά από 0 και 1, ώστε να μπορεί να εκτελεστεί από τον επεξεργαστή του υπολογιστή.

Αντίθετα οι διερμηνείς (interpreters) ελέγχουν μία οδηγία κάθε φορά, την εκτελούν και μετά ελέγχουν την επόμενη οδηγία.



Δεν πρέπει να ξεχνάμε ότι ο υπολογιστής εκτελεί πιστά, όποιες συντακτικά ορθές εντολές και αν του δώσουμε. Αν το αποτέλεσμα, που τελικά προκύπτει από την εκτέλεση του προγράμματος, δεν είναι το αναμενόμενο, τότε το πρόβλημα δε βρίσκεται στον τρόπο εκτέλεσης, αλλά στον αλγόριθμο που κατασκευάσαμε για τη λύση του προβλήματός μας. Στην περίπτωση αυτή λέμε ότι έχουμε κάνει ένα λογικό λάθος και πρέπει να ελέγξουμε ένα προς ένα τα βήματα - εντολές του αλγορίθμου μας, ώστε να διαπιστώσουμε, αν δίνουμε τις κατάλληλες εντολές με τη σωστή σειρά.

Ένα δεύτερο σημείο που πρέπει να γνωρίζουμε, όταν προγραμματίζουμε, είναι ότι για τον υπολογιστή τίποτα δεν είναι αυτονόητο.

Η C είναι σχετικά μικρή γλώσσα. Λίγες δεσμευμένες λέξεις απαρτίζουν το λεξιλόγιό της αλλά παρ΄όλα αυτά θεωρείται μία πανίσχυρη γλώσσα προγραμματισμού πάνω στην οποία αναπτύχθηκαν πολλά λειτουργικά συστήματα. Έχει καλά δομημένες εντολές ελέγχου και ισχυρούς τύπους δεδομένων.

Τα πλεονεκτήματά της είναι τα ακόλουθα:

• συμβατή με οποιοδήποτε μηχάνημα

• γλώσσα προγραμματισμού αρκετά σύντομη

• δομημένη γλώσσα προγραμματισμού: ένα πρόγραμμα αποτελείται από πολλές συναρτήσεις και πολλά ανεξάρτητα εξωτερικά προγράμματα. Δίνει την δυνατότητα επαναχρησιμοποίησης κομματιών κώδικα.

Όταν θέλουμε να γράψουμε και να εκτελέσουμε ένα πρόγραμμα, η διαδικασία που πρέπει να ακολουθήσουμε είναι η εξής:

- 1. γράψιμο εντολών προγράμματος
- 2. αποθήκευση εντολών σε αρχείο πηγαίου κώδικα με επέκταση .c
- 3. μετάφραση (compilation) του πηγαίου κώδικα σε γλώσσα μηχανής. Εάν ο κώδικας δεν έχει λάθη θα δημιουργηθεί το εκτελέσιμο αρχείο.

Βιβλιοθήκη #include<stdio.h>

Η εντολή αυτή εισάγει στο πρόγραμμα μας τις έτοιμες συναρτήσεις του header file (στην ουσία βιβλιοθήκη συστήματος) με όνομα stdio.h το οποίο αφορά εντολές εισόδου / εξόδου.

int main() {…}

Κάθε πρόγραμμα στην C πρέπει να έχει 1 και μόνο 1 βασική συνάρτηση με το όνομα main. Εφόσον είναι συνάρτηση, θα πρέπει να έχει αρχικά έναν τύπο επιστροφής (int), το όνομά της main και αμέσως μετά παρενθέσεις. Τέλος, θα πρέπει να περιέχει ένα ζευγάρι { } μέσα στα οποία γράφουμε τον κώδικά της.

• Άλλες βιβλιοθήκες: , string.h, signal.h, ….

• Οι πιο μοντέρνες γλώσσες (4ης γενιάς – Java) έχουν ένα τεράστιο σύνολο από βιβλιοθήκες. Αυτό τις καθιστά ουσιαστικά πιο ισχυρές για την επίλυση προβλημάτων

Κάθε πρόγραμμα περιέχει απαραιτήτως μία και μόνο μία συνάρτηση που καλείται `main()`. Ασχέτως με το πόσες συναρτήσεις υπάρχουν σ'ένα πρόγραμμα, η `main()` είναι εκείνη στην οποία περνάει ο έλεγχος από το λειτουργικό σύστημα όταν εκτελείται το πρόγραμμα, μ'άλλα λόγια είναι η πρώτη συνάρτηση που εκτελείται, οπουδήποτε κι αν βρίσκεται εντός του προγράμματος. Η `main()` μπορεί να καλεί άλλη συνάρτηση η οποία μπορεί να καλεί κάποια άλλη κ.ο.κ. Η μορφή ενός προγράμματος που περιέχει μόνο μία συνάρτηση, που προφανώς θα ονομάζεται `main()`, είναι:

```
main()          ← όνομα συνάρτησης (οι παρενθέσεις θα μπορούσαν να έχουν παραμέτρους)
{               ← οριοθέτης
   εντολή1
   εντολή2      ← σώμα της συνάρτησης
      ...
   εντολήn
}               ← οριοθέτης
```

Θα γνωρίσουμε τώρα με συντομία τη συνάρτηση `scanf()` που χρησιμοποιείται για είσοδο δεδομένων στο πρόγραμμα, από το πληκτρολόγιο. Η μορφή της, που θα περιγραφεί αναλυτικά στο Κεφάλαιο 5, είναι παρόμοια με αυτή της `printf()`. Χρησιμοποιεί τους ίδιους καθοριστές μορφής, για την ανάγνωση τιμών, με αυτούς της `printf()`.

Στο παρακάτω πρόγραμμα φαίνεται μια τυπική χρήση της `scanf()`.

```
main()
{
  int i;

  printf("Δώσε την ηλικία σου σε χρόνια: ");
  scanf("%d",&i);
  printf("Είσαι %d ετών",i);
}
```

Όλες οι μεταβλητές πρέπει να δηλώνονται πριν χρησιμοποιηθούν. Μία δήλωση καθορίζει τον τύπο και περιέχει μία λίστα μεταβλητών αυτού του τύπου, όπως φαίνεται παρακάτω:

τύπος λίστα_μεταβλητών;

όπου *τύπος* πρέπει να είναι ένας αποδεκτός τύπος δεδομένου της C, ενώ η *λίστα μεταβλητών* μπορεί να αποτελείται από ένα ή περισσότερα αναγνωριστικά χωρισμένα με κόμμα. Παραδείγματα δηλώσεων μεταβλητών είναι:

```
int i,j,k;
char ch;
```

Υπάρχουν τρεις θέσεις σε ένα πρόγραμμα της C όπου μπορούν να δηλωθούν μεταβλητές. Η πρώτη θέση είναι έξω από όλες τις συναρτήσεις, συμπεριλαμβανομένης της συνάρτησης `main()`. Οι μεταβλητές που δηλώνονται κατ'αυτόν τον τρόπο καλούνται *εξωτερικές* (external) και έχουν *καθολική* (global) εμβέλεια. Έτσι μπορούν να χρησιμοποιούνται από οποιοδήποτε τμήμα του προγράμματος. Η δεύτερη θέση όπου μπορούν να δηλωθούν μεταβλητές είναι μέσα σε μία συνάρτηση. Οι μεταβλητές που δηλώνονται κατ'αυτόν τον τρόπο έχουν *τοπική* (local) εμβέλεια και μπορούν να χρησιμοποιούνται μόνο από αυτή τη συνάρτηση. Η τελευταία θέση όπου μπορούν να δηλωθούν μεταβλητές είναι στη δήλωση των *τυπικών παραμέτρων* μίας συνάρτησης. Εκτός του ότι διαβιβάζουν τιμές στη συνάρτηση, αυτές οι παράμετροι ενεργούν όπως οι υπόλοιπες τοπικές μεταβλητές.

Μία μεταβλητή μπορεί να λάβει *αρχική τιμή* κατά τη δήλωσή της, χρησιμοποιώντας το σύμβολο = και μία σταθερά μετά το όνομα της μεταβλητής. Η γενική μορφή της απόδοσης αρχικής τιμής κατά τη δήλωση, είναι:

```
τύπος  αναγνωριστικό_μεταβλητής = σταθερά;
```

Μερικά παραδείγματα φαίνονται παρακάτω:

```
char ch='C';
int i=0;
float eps=1.0E-5;
```

Ονομασίες σε προγράμματα

• Για σταθερές, μεταβλητές, συναρτήσεις, κτλ.

• Κανόνες Σύνταξης:

1. Αποτελούνται από γράμματα, δεκαδικά ψηφία και underscores (_).

2. Δεν μπορεί να αρχίζουν με δεκαδικό ψηφίο.

3. Σύμβολα όπως &, #, $ δεν επιτρέπονται.

4. Το όνομα μίας μεταβλητής δεν μπορεί να περιέχει κενό.

5. Δεσμευμένες λέξεις (θα εξηγηθούν αργότερα) δεν μπορούν να χρησιμοποιηθούν ως ονομασίες για κάτι άλλο.

6. Ονομασίες που ορίζονται σε κάποια από τις βασικές βιβλιοθήκες δεν πρέπει να ξαναορίζονται (πχ printf())

Η C είναι CASE SENSITIVE, δηλαδή κεφαλαία και μικρά γράμματα θεωρούνται διαφορετικά:

- – foo και FOO είναι δυο διαφορετικά ονόματα
- – One, one και ONE είναι όλες διαφορετικές μεταβλητές.

Χρησιμοποιείτε αυτοεπεξηγηματικά ονόματα

– Π.χ. η ονομασία age είναι πολύ καλύτερη από την ονομασία Α για μία μεταβλητή στην οποία θα καταχωρούνται ηλικίες.

Βιβλιογραφία

Πηγή: Διαδίκτυο

Νάστου, Π. *Εισαγωγή στις γλώσσες προγραμματισμού με τη γλώσσα C*. Πανεπιστήμιο Αιγαίου, Τμήμα Μαθηματικών, Σάμος.

Η ΓΛΩΣΣΑ C http://www.ba.teiwest.gr/Nea%20Mathimata/c/LECTURE%2003.PDF

«Η Γλώσσα Προγραμματισμού C», Αθανάσιος Ε. Κουτσονικόλας, Ιούνιος 2021.