Κεφάλαιο ΙΙΙ:

Οι εντολές ελέγχου της ροής ενός προγράμματος.

3.1 Εντολές ελέγχου της ροής.

Στο παρόν κεφάλαιο ασχολούμαστε με την σύνταξη των εντολών της C οι οποίες εισάγουν λογική και ελέγχουν την ροή εκτέλεσης σε ένα πρόγραμμα. Οι εντολές με τις οποίες θα ασχοληθούμε είναι οι ακόλουθες:

- **if-else και else-if**
- **switch**
- **for**
- **while**
- **do-while**
- **break** και **continue**
- **goto** και **ετικέτες**

3.2 Οι εντολές if-else και else-if.

Η βασική εντολή στη C με την οποία μπορούμε να αναπτύξουμε λογική στα προγράμματά μας είναι η εντολή **if**. Η ποιο απλή σύνταξη της εντολής **if** είναι η ακόλουθη:

```
if (παράσταση) {
        γραμμή_1;
        γραμμή_2;
        .
        .
}
```

Εάν η ***παράσταση*** είναι αληθής τότε εκτελούνται όλες οι γραμμές οι οποίες περικλείονται μέσα στις αγκύλες. Εάν η ***παράσταση*** δεν είναι αληθής τότε όλες οι γραμμές, οι οποίες περικλείονται μέσα στις αγκύλες, αγνοούνται και το πρόγραμμα συνεχίζει παρακάτω.

Εάν ο αριθμός των γραμμών προς εκτέλεση είναι 1 τότε μπορούν να παραληφθούν οι αγκύλες. Η χρησιμοποίηση των αγκυλών πάντως δεν απαγορεύεται ακόμη και εάν έχουμε μία μόνο γραμμή. Η μορφή της εντολής if σε αυτή την περίπτωση γίνεται απλά:

```
if (παράσταση)
        γραμμή_1;
```

Προσοχή! Καλό είναι για ένα αρχάριο προγραμματιστή ακόμη και εάν υπάρχει μία μόνο γραμμή μέσα στην if να χρησιμοποιεί τις αγκύλες.

Η γενική μορφή της εντολής **if-else** είναι η ακόλουθη:

```
if (παράσταση) {
        γραμμή_1;
        γραμμή_2;
        .
}
else {
        γραμμή_A;
        γραμμή_B;
        .
}
```

Εάν η ***παράσταση*** είναι αληθής τότε εκτελούνται όλες οι γραμμές οι οποίες περικλείονται από τις πρώτες αγκύλες, δηλαδή η γραμμή_1, η γραμμή_2 κτλ. Εάν η ***παράσταση*** δεν είναι αληθής τότε εκτελούνται όλες οι γραμμές οι οποίες περικλείονται από τις δεύτερες αγκύλες, δηλαδή η γραμμή_A, η γραμμή_B κτλ.

Στην ***παράσταση*** στην οποία αναφερόμαστε παραπάνω χρησιμοποιούμε τους συσχετικούς τελεστές, τους τελεστές ισότητας και τους λογικούς τελεστές τους οποίους συναντήσαμε στο κεφάλαιο ΙΙ. Η παράσταση μπορεί να είναι είτε ψευδής (να παίρνει δηλαδή την τιμή μηδέν), είτε αληθής (να λαμβάνει δηλαδή μη μηδενική τιμή). Ακολουθούν μερικά απλά παραδείγματα της εντολής if:

```
double a, b, c;
a=3.4;
b=7.8;
c=9.2;

if(a<b){                    /* Έλεγχος εάν a<b. Επειδή είναι αληθές θα εκτελεστεί η γραμμή */
   printf("a<b\n");         /* Η γραμμή εκτελείται και εκτυπώνεται το a<b  */
}

if(a==c){                   /* Έλεγχος εάν a=b. Επειδή είναι ψευδές δεν θα εκτελεστεί η γραμμή */
   printf("a=b \n");        /* Η γραμμή δεν εκτελείται, δεν εκτυπώνεται κάτι  */
}

If((a<b)&&(a<c)){                   /* Έλεγχος εάν a<b και ταυτόχρονα εάν a<c. Είναι αληθές */
   printf("a<b and a<c \n");        /* Η γραμμή εκτελείται και εκτυπώνεται το a<b and a<c  */
}

If((a<b)||(a>c)){                   /* Έλεγχος εάν a<b ή a>c. Είναι αληθές */
   printf("a<b or a>c \n");         /* Η γραμμή εκτελείται και εκτυπώνεται το a<b or a>c  */
}
```

Η γενική μορφή της εντολής **if-else** συμπεριλαμβανομένης και της **else-if** είναι η ακόλουθη:

```
if (παράσταση_1) {
        γραμμή_1_1;
        γραμμή_1_2;
        .
}
else if(παράσταση_2)  {
        γραμμή_2_1;
        γραμμή_2_2;
        .
}
else if(παράσταση_3)  {
        γραμμή_3_1;
        γραμμή_3_2;
        .
}
.
.
.
.
else {
        γραμμή_Α;
        γραμμή_Β;
        .
}
```

Εάν η ***παράσταση_1*** είναι αληθής τότε εκτελούνται οι γραμμές: γραμμή_1_1, γραμμή_1_2 κτλ. Εάν η ***παράσταση_2*** είναι αληθής τότε εκτελούνται οι γραμμές: γραμμή_2_1, εγραμμή_2_2 κτλ. Εάν η ***παράσταση_3*** είναι αληθής τότε εκτελούνται οι γραμμές: γραμμή_3_1, γραμμή_3_2 κτλ. Εάν καμία από τις: ***παράσταση_1***, ***παράσταση_2, παράσταση_3*** κτλ. δεν αληθεύουν, τότε εκτελούνται οι γραμμές μετά το else, δηλαδή η γραμμή_Α, η γραμμή_Β κτλ.

Καταλαβαίνουμε πως η τελευταία σύνταξη είναι η γενικότερη και πως οι προηγούμενες αποτελούν υποπεριπτώσεις αυτής. Σημειώνουμε πως μπορούμε να έχουμε μεγάλο αριθμό από εντολές else if τη μία μετά από την άλλη.

Ας δούμε ένα απλό παράδειγμα. Να γραφεί ένα πρόγραμμα το οποίο να συγκρίνει μεταξύ τους δύο ακεραίους αριθμούς. Οι δύο ακέραιοι να εισαχθούν στο πρόγραμμα με την χρήση της συνάρτησης scanf(). Μια πιθανή λύση με την χρήση της εντολής if είναι το πρόγραμμα το οποίο εικονίζεται στο επόμενο σχήμα 3.1

```
#include<stdio.h>

int main(void)
{
  int a,b;

  printf("Dose toys akeraious a kai b:\n");
  scanf("%d %d%",&a,&b);

  if(a==b){
    printf("O akeraios a=%4d einai isos me ton akeraio b=%4d\n",a,b);
  }

  if(a>b){
    printf("O akeraios a=%4d einai megalyteros apo ton akeraio b=%4d\n",a,b);
  }

  if(a<b){
    printf("O akeraios a=%4d einai mikroteros apo ton akeraio b=%4d\n",a,b);
  }

  return 0;
}
-:--  example_if_1.c      (C Abbrev)--L23--All----------------------------------
```

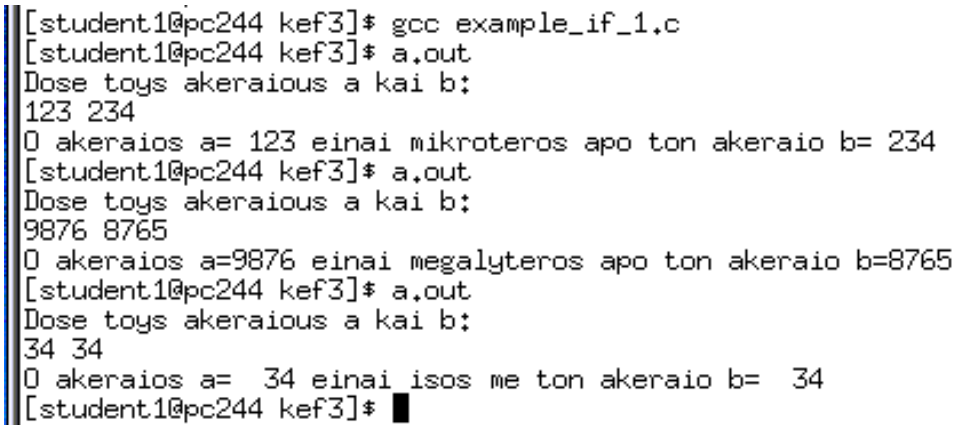
Σχήμα 3.1 Παράδειγμα με την εντολή if.

Μια δεύτερη πιθανή λύση με την χρήση της εντολής if-else και else-if είναι το πρόγραμμα το οποίο εικονίζεται στο επόμενο σχήμα 3.2

```
#include<stdio.h>

int main(void)
{
  int a,b;

  printf("Dose toys akeraious a kai b:\n");
  scanf("%d %d%",&a,&b);

  if(a==b)
    printf("O akeraios a=%4d einai isos me ton akeraio b=%4d\n",a,b);
  else if(a>b)
    printf("O akeraios a=%4d einai megalyteros apo ton akeraio b=%4d\n",a,b);
  else
    printf("O akeraios a=%4d einai mikroteros apo ton akeraio b=%4d\n",a,b);

  return 0;
}
-:--  example_if_2.c      (C Abbrev)--L18--All----------------------------------
```

Σχήμα 3.2 Παράδειγμα με την εντολή if-else και else-if.

Όποια από τις παραπάνω δύο λύσεις του προβλήματος ακολουθήσουμε παίρνουμε το αποτέλεσμα το οποίο εικονίζεται στο ακόλουθο σχήμα 3.3



Σχήμα 3.3 Εκτέλεση των προγραμμάτων τα οποία εικονίζονται στα σχήματα 3.1 και 3.2.

Τελειώνοντας με την εντολή if πρέπει να αναφέρουμε την ύπαρξη του **Τριαδικού Τελεστή (? :)**. Η γενική του σύνταξη είναι η ακόλουθη:

παρασταση_1 ? ***παράσταση_2*** : ***παράσταση_3***

η οποία αντιστοιχεί σε μια απλή εντολή if-else, όπως ακριβώς στην παρακάτω έκφραση:

if (*παράσταση_1*) {
    παράσταση_2;
}
else {
    παράσταση_3;
}

3.3 Η εντολή switch.

Στην προηγούμενη παράγραφο είδαμε πως η εντολή if χρησιμοποιείται όταν πρόκειται να παρθούν συνεχόμενες σχετικές αποφάσεις. Εάν υπάρχουν πολλές αποφάσεις που πρέπει να ληφθούν οι ένθετες εντολές if μπορεί να γίνουν περίπλοκες ο κώδικας μεγάλος και δυσανάγνωστος, πράγμα το οποίο μπορεί να οδηγήσει σε λάθη λογικής. Η C μας παρέχει την εντολή **switch** την οποία συνήθως χρησιμοποιούμε όταν ο αριθμός των αποφάσεων είναι μεγάλος. Επίσης η εντολή switch (η οποία δρα ως διακόπτης επιλογής) ενδείκνυται να χρησιμοποιείται όταν αναπτύσσουμε διάφορα μενού στα προγράμματά μας. Η γενική μορφή της switch είναι η ακόλουθη:

switch (*παράσταση*) {
    case *σταθερή-παράσταση_1* :
        γραμμή_1_1;
        γραμμή_1_2;
        .

```
            .
            break;
        case σταθερή-παράσταση_2 :
            γραμμή_2_1;
            γραμμή_2_2;
            .
            .
            break;
        case σταθερή-παράσταση_2 :
            γραμμή_2_1;
            γραμμή_2_2;
            .
            .
            break;

        .
        .
        .
        default :
            γραμμή_Α;
            γραμμή_Β;
            .
            .
            break;
}
```

Πρώτα υπολογίζεται η ***παράσταση*** και στη συνέχεια εκτελείται η περίπτωση (**case**) της οποίας η ***σταθερή-παράσταση*** ταυτίζεται με αυτή. Όλες οι παραστάσεις case πρέπει να είναι διαφορετικές. Εάν καμία περίπτωση δεν ικανοποιείται τότε εκτελείται η **default**. Στο τέλος κάθε εντολής case και της default αναγράφεται η εντολή **break** η οποία τερματίζει την εντολή switch και το πρόγραμμα συνεχίζει μετά από τέλος αυτής. Η χρήση της εντολής break είναι **απαραίτητη** γιατί οι case εξυπηρετούν απλώς ως επιγραφές.

Ας δούμε ένα παράδειγμα. Να γραφεί ένα πρόγραμμα το οποίο ως είσοδο να έχει έναν ακέραιο από το 1 έως το 7 και ως έξοδο να τυπώνει την αντίστοιχη ημέρα της εβδομάδας. Μια πιθανή λύση του προβλήματος εικονίζεται στο σχήμα 3.4. Η εκτέλεση και τα αποτελέσματα του προγράμματος του σχήματος 3.4 εικονίζονται στο σχήμα 3.5.

```
#include<stdio.h>

int main(void)
{
  int day;
  printf("Doste thn hmera tis ebdomadas apo to 1 eos to 7\n");
  scanf("%d",&day);

    switch(day){
    case 1:
      printf("H ptoti mera tis ebdomadas einai h Kiriaki.\n");
      break;
    case 2:
      printf("H deyteri mera tis ebdomadas einai h Deytera.\n");
      break;
    case 3:
      printf("H triti mera tis ebdomadas einai h Triti.\n");
      break;
    case 4:
      printf("H tetarti mera tis ebdomadas einai h Tetarti.\n");
      break;
    case 5:
      printf("H pempti mera tis ebdomadas einai h Pempti.\n");
      break;
    case 6:
      printf("H ekti mera tis ebdomadas einai h Paraskeyi.\n");
      break;
    case 7:
      printf("H ebdomi mera tis ebdomadas einai to Sabato.\n");
      break;
    }
  return 0;
}
-:--  example_case.c      (C Abbrev)--L33--All--------------------------------
```

Σχήμα 3.4 Παράδειγμα με την εντολή case.

```
[student1@pc244 kef3]$ gcc example_case.c
[student1@pc244 kef3]$ a.out
Doste thn hmera tis ebdomadas apo to 1 eos to 7
2
H deyteri mera tis ebdomadas einai h Deytera.
[student1@pc244 kef3]$ a.out
Doste thn hmera tis ebdomadas apo to 1 eos to 7
5
H pempti mera tis ebdomadas einai h Pempti.
[student1@pc244 kef3]$ a.out
Doste thn hmera tis ebdomadas apo to 1 eos to 7
7
H ebdomi mera tis ebdomadas einai to Sabato.
[student1@pc244 kef3]$
```

Σχήμα 3.5 Η εκτέλεση του προγράμματος το οποίο εικονίζεται στο σχήμα 3.4.

3.4 Η εντολή for.

Συνεχίζουμε με τις εντολές που αφορούν την δημιουργία βρόγχων μέσα σε ένα πρόγραμμα. Με την λέξη βρόγχος εννοούμε πως ένα σύνολο γραμμών (που ανήκει στον βρόγχο) μπορεί να εκτελείται επανειλημμένα όσες φορές το επιθυμεί ο προγραμματιστής.

Οι εντολές που σχετίζονται με τους βρόγχους είναι η **for**, η **while** και η **do-while**. Ξεκινούμε με την πλέον δημοφιλέστερη εντολή αυτού του είδους, την εντολή **for**, της οποίας η γενική σύνταξη είναι η ακόλουθη:

```
for (παράσταση_1; παράσταση_2; παράσταση_3){
        γραμμή_1;
        γραμμή_2;
        .
        .
}
```

Το κυρίως σώμα της for αποτελείται από ένα σύνολο γραμμών (γραμμή_1, γραμμή_2 κτλ.) οι οποίες εκτελούνται ανάλογα με την λογική που προσδιορίζουν οι παραστάσεις: ***παράσταση_1, παράσταση_2*** και ***παράσταση_3***. Ποιο συγκεκριμένα η ακριβής εκτέλεση της παραπάνω εντολής έχει ως ακολούθως:

- Πρώτα εκτελείται η ***παράσταση_1***.
- Στη συνέχεια ελέγχεται εάν η ***παράσταση_2*** είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Στη συνέχεια εκτελείται η ***παράσταση_3***.
- Ελέγχεται πάλι εάν η ***παράσταση_2*** είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Εκτελείται πάλι η ***παράσταση_3***.
- Ελέγχεται πάλι εάν η ***παράσταση_2*** είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Εκτελείται πάλι η ***παράσταση_3*** και ελέγχεται εάν η ***παράσταση_2*** είναι αληθής ή ψευδής. Αυτό συνεχίζεται έως ότου κάποια στιγμή η ***παράσταση_2*** γίνει ψευδής οπότε και τερματίζεται η for.

Με απλά λόγια η εντολή for αποτελεί ένα *βρόγχο* ό οποίος εκτελείται **n** φορές (όπου n ακέραιος αριθμός). Ο αριθμός n εξαρτάται από την λογική_που προσδιορίζουν οι παραστάσεις: ***παράσταση_1, παράσταση_2*** και ***παράσταση_3***.

Παράδειγμα: Ας δούμε ένα απλό παράδειγμα με την εντολή for. Να αναπτύξετε ένα πρόγραμμα το οποίο να παράγει και να τυπώνει τους ακεραίους αριθμούς από το 1 έως το 10.

```
#include<stdio.h>

int main(void){
    int i;                          /* Δήλωση ακεραίας μεταβλητής i */

    for(i=1;  i<=10;  i++){         /* Εντολή for. Το i λαμβάνει τιμές από 1 έως 10 με βήμα 1 */
       printf("%d \n",i);           /* Εκτύπωση της ακεραίας μεταβλητής i */
    }

    return 0;                       /* Τερματισμός του προγράμματος */
}
```

Στο παραπάνω απλό πρόγραμμα το σώμα της εντολής for (δηλαδή η γραμμή *printf("%d \n",i);*) εκτελείται συνολικά δέκα φορές. Η μεταβλητή i αρχικοποιείται στην τιμή 1 και ελέγχει την εκτέλεση της εντολής for. Κάθε φορά αυξάνει κατά ένα. Το for τερματίζεται όταν πλέον δεν ισχύει η παράσταση i<=10. Ποιο αναλυτικά η λογική που προσδιορίζει την εντολή for καθορίζεται από τις ακόλουθες τρεις παραστάσεις:

- *παράσταση_*1**: i=1** (Αρχικοποίηση της μεταβλητής i στην τιμή 1).
- *παράσταση_*2**: i<=10** (Έλεγχος εάν η τιμή της μεταβλητής i είναι μικρότερη ή ίση του 10).
- *παράσταση_*3**: i++** (Η μεταβλητή i αυξάνει την τιμή της κατά ένα).

Προσοχή! Η *παρασταση_2* καθορίζει τον τερματισμό των επαναλήψεων. Ο προγραμματιστής θα πρέπει να είναι προσεκτικός σε αυτό το σημείο. Πιθανό λάθος στην *παράσταση_2* μπορεί να οδηγήσει σε ατέρμονα βρόγχο, δηλαδή σε άπειρο αριθμό επαναλήψεων. Αυτό συμβαίνει όταν η *παράσταση_2* είναι πάντοτε αληθής. Σε αυτή την περίπτωση το πρόγραμμα θα εγκλωβιστεί μέσα στον βρόγχο και θα τρέχει συνέχεια. Ο χρήστης σε αυτή την περίπτωση πρέπει να τερματίσει το πρόγραμμα πατώντας στο πληκτρολόγιο **ctrl-c**. Για παράδειγμα το ακόλουθο πρόγραμμα θα τρέχει επ' άπειρον τυπώνοντας ακεραίους αριθμούς ξεκινώντας από το 1:

```
#include<stdio.h>

int main(void){
    int i;                          /* Δήλωση ακεραίας μεταβλητής i */

    for(i=1;  i>0;  i++){           /* Ατέρμονας βρόγχος! Η παράσταση i>0 είναι πάντοτε αληθής */
       printf("%d \n",i);           /* Εκτύπωση της ακεραίας μεταβλητής i */
    }

    return 0;
}
```

Σε περίπτωση που κάποιος προγραμματιστής επιθυμεί την δημιουργία ατέρμονα βρόγχου μπορεί να χρησιμοποιήσει την ακόλουθη σύνταξη:

```
for (  ;  ;  ){
    γραμμή_1;
```

```
        γραμμή_2;
        .
        .
    }
```

Σε αυτή την περίπτωση όμως πρέπει στο σώμα της εντολής for να συμπεριλάβει τις κατάλληλες γραμμές κώδικα οι οποίες κάτω από τις κατάλληλες συνθήκες θα τερματίζουν τον βρόγχο (χρειάζεται για παράδειγμα κάποια εντολή if() συνοδευόμενη από την εντολή break).

3.5 Η εντολή while.

Η εντολή **while** όπως και η for χρησιμοποιείται για την κατασκευή βρόγχων μέσα σε ένα πρόγραμμα. Η σύνταξη της εντολής while είναι η ακόλουθη:

```
while (παράσταση){
    γραμμή_1;
    γραμμή_2;
    .
    .
}
```

Η εκτέλεσή της είναι πολύ απλή. Όσο η ***παράσταση*** είναι αληθής εκτελείται το σώμα της εντολής (δηλαδή η γραμμή_1, γραμμή_2 κτλ.). Αυτό συνεπάγεται πως ο προγραμματιστής πρέπει με κατάλληλη λογική να ελέγχει τη έξοδο από τον βρόγχο.

Παράδειγμα: Ας δούμε το απλό παράδειγμα που αναπτύξαμε με την εντολή for, πως μπορεί να γραφεί κάνοντας χρήση της while. Να αναπτύξετε ένα πρόγραμμα το οποίο να παράγει και να τυπώνει τους ακεραίους αριθμούς από το 1 έως το 10.

```
#include<stdio.h>

int main(void){
    int i;                      /* Δήλωση ακεραίας μεταβλητής i */

    i=1;                        /* Αρχικοποίηση της μεταβλητής i στο 1 */
    while(i<=10){               /* Εντολή while. Έλεγχος εάν  i<=10  */
        printf("%d \n",i);      /* Εκτύπωση της ακεραίας μεταβλητής i */
        i++;                    /* Αύξηση της μεταβλητής i κατά 1 */
    }

    return 0;                   /* Τερματισμός του προγράμματος */
}
```

Όπως παρατηρούμε πάλι χρησιμοποιούμε μια ακέραια μεταβλητή i με την τιμή της οποίας ελέγχεται πλήρως ο βρόγχος. Η μεταβλητή i αρχικοποιείται στο 1. Ο βρόγχος του while εκτελείται εάν i<=10, ενώ πριν την επόμενη επανάληψη η τιμή της μεταβλητής i αυξάνεται κατά 1.

Σημειώνουμε εδώ πως οι εντολές for και while είναι ισοδύναμες. Αυτό φαίνεται παρακάτω όπου γίνεται σύγκριση των δύο εντολών. Είναι θέμα γούστου ποια από τις δύο εντολές μπορεί να χρησιμοποιήσει κάποιος.

Η εντολή for

for (*παράσταση_1*; *παράσταση_2*; *παράσταση_3*){
    γραμμή_1;
    γραμμή_2;
    .
    .
}

Η εντολή while

παράσταση_1;
while(*παράσταση_2*){
    γραμμή_1;
    γραμμή_2;
    .
    παράσταση_3;
}

3.6 Η εντολή do-while.

Η εντολή **do-while** είναι η τρίτη εντολή της C με την οποία μπορούμε να δημιουργήσουμε βρόγχους σε ένα πρόγραμμα. Η σύνταξη της εντολής do-while είναι η ακόλουθη:

```
do{
    γραμμή_1;
    γραμμή_2;
    .
    .
}while(παράσταση);
```

Η εκτέλεσή της είναι πολύ απλή. Το σώμα της εντολής (δηλαδή η γραμμή_1, γραμμή_2 κτλ.), εκτελείται μία φορά και στη συνέχεια ελέγχεται εάν η ***παράσταση*** είναι αληθής ή ψευδής. Εάν η παράσταση είναι αληθής το σώμα της εντολής ξαναεκτελείται έως ότου η παράσταση γίνει ψευδής. Αυτό συνεπάγεται πως ο προγραμματιστής πρέπει με κατάλληλη λογική να ελέγχει τη έξοδο από τον βρόγχο.

Παράδειγμα: Ας δούμε το απλό παράδειγμα που αναπτύξαμε με την εντολή for και την εντολή while πως μπορεί να γραφεί κάνοντας χρήση της do-while. Να αναπτύξετε ένα πρόγραμμα το οποίο να παράγει και να τυπώνει τους ακεραίους αριθμούς από το 1 έως το 10.

```
#include<stdio.h>

int main(void){
    int i;                  /* Δήλωση ακεραίας μεταβλητής i */
    i=1;                    /* Αρχικοποίηση της μεταβλητής i στο 1 */
```

```
    do{                         /* Αρχή της εντολής do-while  */
        printf("%d \n",i);      /* Εκτύπωση της ακεραίας μεταβλητής i */
        i++;                    /* Αύξηση της μεταβλητής i κατά 1 */
    }while(i<=10);              /* Τέλος της εντολής do-while. Έλεγχος εάν  i<=10  */

    return 0;                   /* Τερματισμός του προγράμματος */
}
```

Όπως παρατηρούμε πάλι χρησιμοποιούμε μια ακέραια μεταβλητή i με την τιμή της οποίας ελέγχεται πλήρως ο βρόγχος. Η μεταβλητή i αρχικοποιείται στο 1. Ο βρόγχος του while εκτελείται πάντοτε μία φορά και στο τέλος ελέγχεται εάν i<=10. Η τιμή της μεταβλητής i αυξάνεται κατά 1 πριν ακριβώς από τον έλεγχο.

Σημειώνουμε εδώ πως η εντολή do-while δεν είναι ακριβώς ισοδύναμη με τις εντολές for και while διότι το σώμα της εντολής εκτελείται πάντοτε μία τουλάχιστον φορά. Αυτό συμβαίνει γιατί ο έλεγχος αληθείας της *παράστασης* γίνεται στο τέλος της εντολής και όχι στην αρχή όπως συμβαίνει στις εντολές for και while. Η εντολή do-while γενικά χρησιμοποιείται πολύ λιγότερο από τις εντολές for και while.

3.7 Οι εντολές break και continue.

Η εντολή **break** χρησιμοποιείται όταν θέλουμε να σταματήσουμε την εκτέλεση ενός βρόγχου και να βγούμε από αυτόν. Δείτε το ακόλουθο παράδειγμα:

```
int i;                      /* Δήλωση ακέραιας μεταβλητής i */
for(i=1; i<=10; ++i){       /* Εντολή for. Το i  ξεκινά από 1 έως 10 με βήματα του 1 */

    if(i==6){               /* Μόλις το i πάρει την τιμή εκτελείται το break */
        break;
    }

    printf("%d \n",i);      /*Εκτύπωση του i */
}
```

Στο παραπάνω παράδειγμα ο βρόγχος έχει προγραμματιστεί να τρέξει δέκα φορές (το i ξεκινά από 1 έως 10 με βήματα του 1). Κάθε φορά εκτυπώνεται η τιμή της μεταβλητής i. Μόλις όμως το i γίνει ίσο με το 6 εκτελείται το break το οποίο βρίσκεται μέσα στην if(). Αυτό έχει ως αποτέλεσμα τον τερματισμό της εκτέλεσης του βρόγχου και την έξοδο από αυτόν. Ο παραπάνω κώδικας λοιπόν θα εκτυπώσει τις τιμές από το 1 έως και το 5.

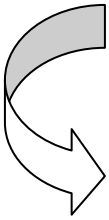
Η εντολή **continue** χρησιμοποιείται όταν θέλουμε να επιστρέψουμε στην κορυφή του βρόγχου αγνοώντας τις υπόλοιπες γραμμές του σώματός του. Το παρακάτω παράδειγμα κάνει κατανοητή την χρήση του continue:

```
int i;                          /* Δήλωση ακέραιας μεταβλητής i */
for(i=1; i<=10; ++i){           /* Εντολή for. Το i  ξεκινά από 1 έως 10 με βήματα του 1 */
    if((i==4)||(i==7)){         /* Μόλις το i πάρει τις τιμές 4 ή 7 εκτελείται το continue */
        continue;
    }
    printf("%d \n",i);          /*Εκτύπωση του i */
}
```

Στο παραπάνω παράδειγμα ο βρόγχος έχει προγραμματιστεί να τρέξει δέκα φορές (το i ξεκινά από 1 έως 10 με βήματα του 1). Κάθε φορά εκτυπώνεται η τιμή της μεταβλητής i. Μόλις όμως το i γίνει ίσο με το 4 ή το 7 εκτελείται το continue το οποίο βρίσκεται μέσα στην if(). Αυτό έχει ως αποτέλεσμα την επιστροφή κάθε φορά στην κορυφή του βρόγχου. Ο παραπάνω κώδικας λοιπόν θα εκτυπώσει στη σειρά τις ακόλουθες τιμές: 1 2 3 5 6 8 9 και 10. Δεν θα εκτυπωθούν δηλαδή οι τιμές 4 και 7.

3.8 Η εντολή goto και οι ετικέτες.

Η εντολή **goto** αποτελεί κατάλοιπο από άλλες γλώσσες προγραμματισμού (όπως η Fortran) και σήμερα αποφεύγουμε να την χρησιμοποιούμε. Ο λόγος είναι απλός. Η goto επιτρέπει την μετάβαση σε όποιο σημείο του κώδικα επιθυμούμε. Αυτός ο τρόπος προγραμματισμού σήμερα δεν είναι αποδεκτός μιας και μπορεί πολύ εύκολα να χαθεί η λογική μέσα στο πρόγραμμά μας. Παραθέτουμε εδώ την εντολή για λόγους πληρότητας. Η σύνταξή της είναι η ακόλουθη:



```
γραμμή_προγράμματος;
γραμμή_προγράμματος;
γραμμή_προγράμματος;
goto sinexeia_edw;              /* Μετάβαση στην ετικέτα sinexeia_edw */
γραμμή_προγράμματος;
γραμμή_προγράμματος;
γραμμή_προγράμματος;
sinexeia_edw:                   /* Ετικέτα με όνομα: sinexeia_edw */
γραμμή_προγράμματος;
γραμμή_προγράμματος;
```