Η ΓΛΩΣΣΑ C

Μάθημα 11:

Έλεγχος Ροής Προγράμματος

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Έλεγχος Προγράμματος

1. Η εντολή άμεσης επανάληψης continue
2. Η εντολή διακοπής επανάληψης break
3. Η εντολή μεταφοράς ελέγχου προγράμματος goto
4. Η εντολή ελέγχου πολλαπλής συνθήκης switch

Β. Προγραμματιστικές Τεχνικές

1. Ατέρμονες Βρόχοι για την κατασκευή menu
2. Άμεση Έξοδος από το πρόγραμμα
3. Εκτέλεση Εντολών Συστήματος

Γ.Ασκήσεις

Α. Έλεγχος Προγράμματος

1. Η εντολή άμεσης επανάληψης continue

- Ας θεωρήσουμε το σώμα μιας οποιασδήποτε εντολής επανάληψης (δηλαδή μέσα στα άγκιστρα μιας for, μιας while, ή μιας do…while).
  - Κάποιες φορές, ανάλογα με το αν ισχύει μια συνθήκη, δεν θέλουμε να εκτελεστούν οι επόμενες εντολές της επανάληψης, αλλά να ξεκινήσει άμεσα το επόμενο βήμα επανάληψης.
  - Στην περίπτωση αυτή, χρησιμοποιούμε την εντολή **continue**. Με την εντολή αυτή ξεκινά άμεσα το επόμενο βήμα επανάληψης.
  - Ο συνηθισμένος τρόπος χρήσης της continue (π.χ. Μέσα σε μια for) φαίνεται στον εξής κώδικα:

```
for (i=0; i<N; i++)
{
      ...
      if (συνθήκη)
         continue;
      ...
}
```

  - Μελετήστε το πρόγραμμα της επόμενης διαφάνειας που δείχνει τον τρόπο λειτουργίας της continue σε ένα βρόχο επανάληψης.

Α. Έλεγχος Προγράμματος

1. Η εντολή άμεσης επανάληψης continue

```
/* continue.c: Deixnei tin xrisi tis continue */

#include <stdio.h>

main()
{
    int i;

    for (i=0; i<10; i++)
    {
       printf("\nArithmos: %d",i);
       if (i%2==0)
           continue;
       printf(", kai to tetragono tou: %d",i*i);
    }
}
```

Α. Έλεγχος Προγράμματος
2. Η εντολή διακοπής επανάληψης break

- Ας θεωρήσουμε το σώμα μιας οποιασδήποτε εντολής επανάληψης (δηλαδή μέσα στα άγκιστρα μιας for, μιας while, ή μιας do…while).
  - Κάποιες φορές, ανάλογα με το αν ισχύει μια συνθήκη, θέλουμε να διακοπεί άμεσα η επανάληψη.
  - Στην περίπτωση αυτή, χρησιμοποιούμε την εντολή break. Με την εντολή αυτή διακόπτεται άμεσα η επανάληψη και πηγαίνουμε στην πρώτη εντολή μετά την επανάληψη.
  - Ο συνηθισμένος τρόπος χρήσης της break (π.χ. Μέσα σε μια for) φαίνεται στον εξής κώδικα:

```
for (i=0; i<N; i++)
{
      ...
      if (συνθήκη)
         break;
      ...
}
```

  - Μελετήστε το πρόγραμμα της επόμενης διαφάνειας που δείχνει τον τρόπο λειτουργίας της break σε ένα βρόχο επανάληψης.

Α. Έλεγχος Προγράμματος
2. Η εντολή διακοπής επανάληψης break

```
/* break.c: Deixnei tin xrisi tis break */

#include <stdio.h>

#define N 5

main()
{
    int i;
    int A[N]= {5, 3, 2, 4, 8 }; // Pinakas dedomenwn
    int x=2; // Pros anazitisi stoixeio

    /* Psaxnoume gia to stoixeio x ston pinaka A */
    for (i=0; i<N; i++)
    {
        if (A[i]==x)
           break;
    }
    if (i<N)
         printf("\nVrethike to stoixeio %d sti thesi %d",x,i);
    else
         printf("\nDen Vrethike to stoixeio");
}
```

Α. Έλεγχος Προγράμματος

3. Η εντολή μεταφοράς ελέγχου προγράμματος goto

- Έχουμε το δικαίωμα να καθορίσουμε ρητά στο πρόγραμμα να μεταβεί σε ένα συγκεκριμένο σημείο με την εντολή goto.
- Για να την χρησιμοποιήσουμε γράφουμε μια ετικέτα (καθορίζουμε ένα «σημάδι» στο πρόγραμμα μας) με το συντακτικό:

```
ΟνομαΕτικέτας:
```

  - Το όνομα το καθορίζουμε εμείς ακολουθούμενο από άνω-κάτω τελεία.
- Και έπειτα οπουδήποτε στην συνάρτηση που γράφουμε μπορούμε να γράψουμε την εντολή:

```
goto ΟνομαΕτικέτας;
```

  - Και αυτόματα το πρόγραμμα μας θα μεταβεί στο σημείο που έχουμε γράψει την ετικέτα και θα εκτελέσει τις εντολές από εκείνο το σημείο.
- Δείτε στο πρόγραμμα της επόμενης διαφάνειας πως χρησιμοποιούμε την goto για να γράψουμε με έμμεσο τρόπο μία επανάληψη.

Α. Έλεγχος Προγράμματος

3. Η εντολή μεταφοράς ελέγχου προγράμματος goto

```
/* goto.c: Deixnei tin xrisi tis goto */

#include <stdio.h>

main()
{
    int i=0;

label: // To onoma tis etiketas
    printf("\ni=%d",i);

    if (i<5)
    {
       i++;
       goto label;
    }
}
```

Α. Έλεγχος Προγράμματος

3. Η εντολή μεταφοράς ελέγχου προγράμματος goto

- Αν και η goto φαίνεται να λύνει εύκολα πολλά προβλήματα, εύκολα μπορεί να γίνει κατάχρησή της και το πρόγραμμα να μην είναι πλέον εύκολα αναγνώσιμο.
- Πολλές ετικέτες θα κάνουν δυσνόητη την ροή του προγράμματος, ενώ προγραμματιστικές τεχνικές που έχουμε μάθει καταργούνται με την χρήση της goto.
- Εύκολα ο κώδικας μπορεί να εκτραπεί σε κώδικα-μακαρόνι στον οποίο κάνουμε όλην την δουλειά με ετικέτες και goto.
- Για όλους αυτούς τους λόγους:

Είναι δείγμα **κακού** προγραμματισμού η χρήση της goto.

Μην την χρησιμοποιείτε ποτέ, ακόμη κι όταν φαίνεται ότι η χρήση της θα σας λύσει τα χέρια!

Α. Έλεγχος Προγράμματος
4. Η εντολή ελέγχου πολλαπλής συνθήκης switch

- Η εντολή switch κάνει ευκολότερο τον προγραμματισμό, όταν θέλουμε να εκτελέσουμε διαφορετικές ενέργειες ανάλογα με την τιμή που έχει μια μεταβλητή.

- Ξέρουμε ήδη να το κάνουμε αυτό βέβαια με διαδοχικές if…else if…else.
  - Ωστόσο η εντολή switch είναι πιο εύκολα αναγνώσιμη και χρησιμοποιείται κατά κόρον, ιδίως όταν οι τιμές που έχουμε να ελέγξουμε είναι περισσότερες από 2.

- Ας δούμε το παράδειγμα της επόμενης διαφάνειας. Αριστερά βλέπουμε διαδοχικές if…else που εκτελούν διαφορετικές εντολές ανάλογα με την τιμή μιας ακέραιας μεταβλητής x:
  - Δεξιά βλέπουμε πως συντάσσουμε την switch:
    - Μέσα σε παρένθεση γράφουμε το όνομα της μεταβλητής.
    - Έπειτα με διαδοχικά case Τιμη : επιλέγουμε ποιες γραμμές κώδικα θα τρέξουν ανάλογα με την τιμή της μεταβλητής.
    - Το τελικό else γράφεται στην switch με την εντολή default
    - Στο τέλος κάθε case γράφουμε break.

Α. Έλεγχος Προγράμματος

4. Η εντολή ελέγχου πολλαπλής συνθήκης switch

```
if (x==1)
{
   εντολέςΑ;
}
else if (x==2)
{
   εντολέςΒ;
}
else if (x==3)
{
   εντολέςΓ;
}
else if (x==4)
{
   εντολέςΔ;
}
else
{
   εντολέςΕ;
}
```

```
switch(x)
{
case 1:
   εντολέςΑ;
   break;
case 2:
   εντολέςΒ;
   break;
case 3:
   εντολέςΓ;
   break;
case 4:
   εντολέςΔ;
   break;
default:
   εντολέςΕ;
}
```

Α. Έλεγχος Προγράμματος
4. Η εντολή ελέγχου πολλαπλής συνθήκης switch

- Η εντολή switch μπορεί να χρησιμοποιηθεί με τύπους δεδομένων που αποτιμώνται σε ακέραια τιμή (π.χ. int και char) και όχι με πραγματικές τιμές (π.χ. float ή double)

- Προσέξτε ιδιαίτερα την χρήση της break στο τέλος κάθε case.
  - Είναι απαραίτητη! Αν δεν την γράψουμε, τότε η εκτέλεση θα συνεχιστεί και στο επόμενο case, μέχρι να συναντήσει μία break, ή μέχρι να φτάσει στην τελευταία εντολή της switch που είναι μέσα στα βασικά άγκιστρα.
  - Υπόψη ότι αυτή η break δεν έχει καμία σχέση με την break που μάθαμε για την διακοπή ενός βρόχου. Ο μεταγλωττιστής καταλαβαίνει σε τι αναφέρεται το break ανάλογα με το πεδίο εφαρμογής της στον κώδικα.

- Μελετήστε το πρόγραμμα της επόμενης διαφάνειας στο οποίο ο προγραμματιστής δεν έβαλε τις απαραίτητες break στο τέλος κάθε case!
  - Δείτε ότι το πρόγραμμα μεταβαίνει στα επόμενα case εωσότου συναντήσει την πρώτη break.

Α. Έλεγχος Προγράμματος

4. Η εντολή ελέγχου πολλαπλής συνθήκης switch

```
/* wrong_switch.c: Deixnei mia kaki xrisi tis switch xwris ta aparaitita break */
#include <stdio.h>
main()
{
    int x;

    scanf("%d",&x);

    switch(x)
    {
    case 1:
         printf("\nEisagate 1");
    case 2:
         printf("\nEisagate 2");
    case 3:
         printf("\nEisagate 3");
    case 4:
         printf("\nEisagate 4");
    default:
         printf("\nEisagate arithmo >=5");
    }
}
```

Β. Προγραμματιστικές Τεχνικές
1. Ατέρμονες Βρόχοι για την κατασκευή menu

- Η εντολή switch χρησιμοποιείται σε συνδυασμό με έναν ατέρμονα βρόχο για την κατασκευή ενός menu για να διαχειριζόμαστε με διαφορετικούς τρόπους τις επιλογές του χρήστη.

- Δείτε το πρόγραμμα
  - Είναι ο συνηθισμένος τρόπος με τον οποίο ένα πρόγραμμα C διαχειρίζεται διαφορετικές επιλογές που εισάγει ο χρήστης. Απομένει να γράψουμε τις συναρτήσεις που εκτελούνται σε κάθε διαφορετική επιλογή.

- Χρησιμοποιούμε έναν ατέρμονα βρόχο – δηλαδή μια επανάληψη που δεν τελειώνει ποτέ. Συγκεκριμένα, η συνθήκη while(1) επιτυγχάνει πάντα, άρα επαναλαμβάνεται συνεχώς η εκτέλεση του βρόχου εωσότου με την εκτέλεση της εντολής exit() τερματίσουμε το πρόγραμμα.

Β. Προγραμματιστικές Τεχνικές

1. Ατέρμονες Βρόχοι για την κατασκευή menu

```
/* menu.c: Kataskeyi menu me tin switch */

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

void func1();
void func2();
void func3();

main()
{
    int epilogi;

    while(1)
    {
       printf("\nMenu");
       printf("\n------------");
       printf("\n1-Epilogi 1");
       printf("\n2-Epilogi 2");
       printf("\n3-Epilogi 3");
       printf("\n4-Exit");
       printf("\n------------");
       printf("\nEpilogi? ");
       scanf("%d",&epilogi);

       switch(epilogi)
       {
       case 1:
            func1();
            break;
       case 2:
            func2();
            break;
       case 3:
            func3();
            break;
       case 4:
            printf("\nExodos apo to
programma");
            exit(0);
            break;
       default:
            printf("\nLanthasmeni Epilogi");
       }
       system("pause");
    }
}
```

Β. Προγραμματιστικές Τεχνικές
2. Άμεση Έξοδος από το Πρόγραμμα

- Στο προηγούμενο πρόγραμμα χρησιμοποιήσαμε την συνάρτηση:

```
void exit(int message)
```

  - Ορισμένη στην βιβλιοθήκη συναρτήσεων stdlib.h

- Η οποία όταν κληθεί τερματίζει άμεσα το πρόγραμμα.
  - Μέσω του ορίσματός της μπορούμε να επιστρέψουμε μια τιμή στο λειτουργικό σύστημα.
    - Όλα τα προγράμματα που τρέχουν στον υπολογιστή καλούνται από το λειτουργικό σύστημα.
    - Έτσι όπως μια συνάρτηση επιστρέφει μία τιμή στην συνάρτηση που την κάλεσε, έτσι και η συνάρτηση exit επιστρέφει μέσω ορίσματος μια τιμή στο λειτουργικό σύστημα.
  - Κατά σύμβαση το όρισμα 0, αντιστοιχεί σε επιτυχή έξοδο από το πρόγραμμα, ενώ το όρισμα 1, αντιστοιχεί σε ανεπιτυχή έξοδο από το σύστημα.

Β. Προγραμματιστικές Τεχνικές
3. Εκτέλεση Εντολών Συστήματος

- Έχουμε το δικαίωμα να εκτελέσουμε μια εντολή του συστήματος με την εντολή

```
void system(char *entoli)
```

  - Ορισμένη στην βιβλιοθήκη συναρτήσεων stdlib.h
- Η εντολή αυτή τρέχει στο λειτουργικό σύστημα την εντολή που δέχεται ως όρισμα ως συμβολοσειρά.
- Αν και ξεφεύγει από τους στόχους της παρουσίασης αυτής η αναφορά στις εντολές συστήματος (διότι εξαρτάται και από το λειτουργικό μας σύστημα), μία συνηθισμένη χρήση της είναι η εντολή:

```
system("pause");
```

  - Η οποία καλεί την εντολή συστήματος pause που κάνει τον υπολογιστή να «παγώσει» μέχρι να πατήσουμε Enter.

- και η εντολή:

```
system("cls");
```

  - που «καθαρίζει» την οθόνη

Γ. Ασκήσεις

1. Τροποποίηση Παλαιότερου Προγράμματος

Στο «Μάθημα 9 Εισαγωγή στην Είσοδο/Έξοδο Δεδομένων – Εφαρμογή 1» κατασκευάσαμε ένα πρόγραμμα το οποίο είχε 4 επιλογές ανάλογα με την είσοδο του χρήστη.

- Τροποποιήστε το πρόγραμμα ώστε να χρησιμοποιεί την switch αντί για διαδοχικές if…else.

Γ. Ασκήσεις

2. Κατασκευή Μενού Επιλογών

Στο « Μάθημα 8: Δείκτες – Εφαρμογή 5» κατασκευάσαμε ένα πρόγραμμα το οποίο εκτελούσε συνήθεις πράξεις σε πίνακες.

- Τροποποιήστε το πρόγραμμα ώστε να χρησιμοποιεί την switch στην κατασκευή του μενού αντί για διαδοχικές if…else.
- Να εμφανίζεται επίσης κατάλληλο μήνυμα αν ο χρήστης εισάγει λάθος είσοδο.

Γ. Ασκήσεις
3. Κατασκευή Μενού Επιλογών

Συνεχίζουμε την τροποποίηση της προηγούμενης εφαρμογής ως εξής:

- Μετά από κάθε επιλογή του χρήστη, να «παγώνει» το πρόγραμμα με χρήση της εντολής system(“pause”) ώστε να μην εμφανίζεται αμέσως εκ νέου το μενού επιλογών. Να δίνεται μια προτροπή στον χρήστη να πατήσει Enter για να συνεχίσει.
- Να καθαρίζει η οθόνη μόλις εκκινά η εκτύπωση του μενού με την εντολή system(“cls”)