Προγραμματισμός Ι

Έλεγχος Προγράμματος

Πανεπιστήμιο Πελοποννήσου
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Νικόλαος Δ. Τσελίκας

Η εντολή if (Ι)

- Η εντολή **if** είναι μία από τις βασικότερες δομές ελέγχου ροής στη C, αλλά και στις περισσότερες γλώσσες προγραμματισμού
- Με την εντολή **if** γίνεται δυνατή η επιλεκτική εκτέλεση ενός τμήματος κώδικα, ανάλογα με την τιμή μίας συνθήκης
- Γενική σύνταξη της εντολής **if** (στην πιο απλή της μορφή):

```
if(συνθήκη)
{
      ... /* ομάδα εντολών */
}
```

Η εντολή if (II)

- Αν η συνθήκη είναι αληθής (true), τότε εκτελούνται οι εντολές που περικλείονται στα άγκιστρα {...}

```
int x = 3;
if(x != 0)
{
        printf("Yes\n");
}
```

- Αν η συνθήκη δεν είναι αληθής, δηλαδή αν η συνθήκη είναι ψευδής (false), τότε το μπλοκ των εντολών που περικλείεται στα άγκιστρα παρακάμπτεται και συνεπώς δεν εκτελείται

```
int x = -3;
if(x == 0)
{
        printf("Yes\n");
}
```

Παρατηρήσεις (Ι)

- Αν το μπλοκ εντολών περιέχει μόνο μία εντολή, τότε τα άγκιστρα μπορούν να παραλειφθούν

- Π.χ.

```
int x = 3;
if(x > 0)
      printf("x is positive\n");
```

- Αν, βέβαια, το μπλοκ εντολών περιέχει περισσότερες από μία εντολές, τότε τα άγκιστρα είναι απαραίτητα

```
int x = 3;
if(x > 0)
{
      printf("x is positive\n");
      printf("In other words, x is greater than zero\n");
}
```

Παρατηρήσεις (ΙΙ)

ΠΡΟΣΟΧΗ!!!

- Μην βάζετε το ελληνικό ερωτηματικό ; στο τέλος της **if** εντολής, γιατί ουσιαστικά το ερωτηματικό τερματίζει στο σημείο εκείνο την εντολή **if**

- Π.χ. τί εμφανίζει το παρακάτω παράδειγμα ???

```
int x = -3;
if(x > 0);
      printf("x is positive\n");
```

- και τί αυτό ???

```
int x = 3;
if(x > 0);
      printf("x is positive\n");
```

- Στην οθόνη εμφανίζεται το μήνυμα **x is positive** ανεξάρτητα από την τιμή της μεταβλητής **x**

Παρατηρήσεις (III)

ΠΡΟΣΟΧΗ!!!

- Μην συγχέετε τον τελεστή ελέγχου ισότητας == (διπλό ίσον) με τον τελεστή εκχώρησης = (μονό ίσον)

- Το παρακάτω πρόγραμμα εμφανίζει στην οθόνη **Yes**, παρόλο που η αρχική τιμή της μεταβλητής **x** είναι **-10** (αφού η συνθήκη **x=-20** είναι πάντα αληθής, δεδομένου ότι πρόκειται για ανάθεση μη μηδενικής τιμής)

```
int x = -10;
if(x = -20)
        printf("Yes\n");
```

- Για να είχαμε «σωστό χειρισμό» στη συνθήκη **if**, η συνθήκη θα έπρεπε να γραφεί ως **if(x == -20)**, δηλαδή με διπλό ίσον και όχι με μονό

Παρατηρήσεις (IV)

ΠΡΟΣΟΧΗ!!!

- Η εκχώρηση μίας μη μηδενικής τιμής σε μία μεταβλητή ισοδυναμεί με **αληθή συνθήκη**, ενώ η εκχώρηση μηδενικής τιμής ισοδυναμεί με **ψευδή συνθήκη**

- Π.χ. τι εμφανίζει το παρακάτω κομμάτι κώδικα;

```
int x = 0;
if(x = 0)
        printf("Yes\n");
```

- Δεν εμφανίζεται τίποτα, δεδομένου ότι η συνθήκη είναι ψευδής (έχουμε εκχώρηση της μηδενικής τιμής)

- Για να είχαμε «σωστό χειρισμό» στη συνθήκη `if`, η συνθήκη θα έπρεπε να γραφεί ως `if(x == 0)`, δηλαδή με διπλό ίσον και όχι με μονό

Παρατηρήσεις (V)

- Η έκφραση:

  `if(x)` είναι ισοδύναμη με `if(x != 0)`

- Η έκφραση:

  `if(!x)` είναι ισοδύναμη με `if(x == 0)`

- Η εντολή `if` μπορεί προαιρετικά να συμπληρώνεται με την εντολή `else`, όπως θα δούμε στη συνέχεια

Παρατηρήσεις (VI)

- Ορισμένοι προγραμματιστές προτιμούν να γράφουν πρώτα τη σταθερά, όταν θέλουν να συγκρίνουν την τιμή μιας μεταβλητής με μία σταθερά, π.χ. :

```
if(-20 == x) /* Αντί για if(x == -20) */
```

- Ο λόγος είναι ότι σε περίπτωση που ξεχάσουν το ένα «ίσον» (=), ο μεταγλωττιστής θα εμφανίσει μήνυμα λάθους, δεδομένου ότι δεν επιτρέπεται να ανατεθεί η τιμή μιας μεταβλητής σε μία σταθερά

- Ένα επίσης συνηθισμένο λάθος είναι η «ανάποδη γραφή» του τελεστή != σε κάποια σύγκριση, π.χ.:

```
#include <stdio.h>
int main(void)
{
      int i = 30;

      if(i =! 20) /* Instead of != */
            printf("One\n");
      else
            printf("Two\n");
      printf("%d\n", i);
      return 0;
}
```

Με την «ανάποδη γραφή» εφαρμόζεται ο τελεστής ! στη στεθερά 20, άρα το i γίνεται 0, άρα η συνθήκη ψευδής. Συνεπώς, το πρόγραμμα εμφανίζει Two και 0

Η εντολή if...else (Ι)

- Όταν θέλουμε να προσδιορίσουμε μία ομάδα εντολών που θα εκτελεστεί όταν μία συνθήκη είναι αληθής (true) και μία άλλη ομάδα εντολών που θα εκτελεστεί όταν η συνθήκη αυτή είναι ψευδής (false), τότε χρησιμοποιούμε την εντολή ελέγχου if...else

- Γενική σύνταξη της εντολής if...else:

```
if(συνθήκη)
{
      ... /* ομάδα εντολών Α */
}
else
{
      ... /* ομάδα εντολών Β */
}
```

Η εντολή if...else (ΙΙ)

- Όταν η συνθήκη είναι αληθής (true), τότε εκτελείται η ομάδα εντολών Α (δηλ. οι εντολές που περιέχονται ανάμεσα στα άγκιστρα του if), ενώ όταν η συνθήκη είναι ψευδής (false), τότε εκτελείται η ομάδα εντολών Β (δηλ. οι εντολές που περιέχονται ανάμεσα στα άγκιστρα του else)

- Π.χ.

```
int x = -3;
if(x > 0)
{
      printf("One\n");
}
else
{
      printf("Two\n");
}
```

- Το παραπάνω κομμάτι κώδικα εμφανίζει Two, αφού το x είναι ίσο με -3, που είναι μικρότερο ή ίσο του 0

Παρατηρήσεις

- Θυμηθείτε ότι στην περίπτωση της εντολής **if**, αν η ομάδα εντολών περιέχει μόνο μία εντολή, τότε τα άγκιστρα μπορούν να παραλειφθούν.

- Το ίδιο ισχύει και στην περίπτωση της εντολής **if...else**

- Δηλαδή, το προηγούμενο παράδειγμα θα μπορούσε να γραφεί και ως εξής:

```
int x = -3;
if(x > 0)
        printf("One\n");
else
        printf("Two\n");
```

- Αν, βέβαια, κάποια από τις ομάδες εντολών περιέχει περισσότερες από μία εντολές, τότε τα άγκιστρα είναι απαραίτητα στο συγκεκριμένο μπλοκ

Ένθετες if εντολές (Ι)

- Στη γενικότερη περίπτωση, τα μπλοκ εντολών των **if** και **else** εντολών επιτρέπεται να περιέχουν και άλλες **if** και **else** εντολές, οι οποίες με τη σειρά τους μπορεί να περιέχουν και άλλες, κ.ο.κ.

- Όταν υπάρχει μία **if** εντολή μέσα σε μία άλλη, τότε αυτή η **if** εντολή ονομάζεται **ένθετη** ή **φωλιασμένη (nested)**

- Παράδειγμα με δύο ένθετες **if** εντολές

```
#include <stdio.h>
int main(void)
{
        int a = 10, b = 20, c = 30;

        if(a > 5)
        {
                if(b == 20)
                        printf("1 ");

                if(c == 40)
                        printf("2 ");
                else
                        printf("3 ");
        }
        else
                printf("4\n");

        return 0;
}
```

Αφού το a είναι μεγαλύτερο του 5, το πρόγραμμα εμφανίζει: 1 3

Ένθετες if εντολές (ΙΙ)

- Στην περίπτωση που ένα πρόγραμμα περιέχει ένθετες if εντολές, ο κανόνας είναι ότι κάθε else εντολή συνδέεται με την αμέσως προηγούμενη if εντολή που υπάρχει στην ίδια ομάδα εντολών (δηλ. ανάμεσα στα ίδια άγκιστρα), αρκεί αυτή να μη σχετίζεται με άλλη else εντολή

```
#include <stdio.h>
int main(void)
{
        int a = 5;
      ? if(a != 5)
        {
            ? if(a-2 > 5)
                    printf("One\n");
              else
                    printf("Two\n");
        }
        return 0;
}
```

- Όταν γίνεται χρήση ένθετων εντολών if προτείνεται η χρήση των αγκίστρων, για να είναι πιο ξεκάθαρη η σχέση μεταξύ των εντολών else και if (ιδιαίτερα στην περίπτωση που στο πρόγραμμά σας χρησιμοποιείτε μεγάλο αριθμό από if και else εντολές)

Ένθετες if εντολές (III)

- Στο διπλανό πρόγραμμα, η εντολή `else printf("3\n");` αντιστοιχεί στην πλησιέστερη `if` εντολή, που είναι η `if(c == 40)`
- Όμως, η τελική εντολή `else printf("4\n");` δεν αντιστοιχίζεται με την πλησιέστερη `if` εντολή, που είναι η `if(b == 20)`, γιατί δεν ανήκουν στο ίδιο μπλοκ
- Η εντολή αυτή συνδέεται με την εντολή `if(a > 5)`
- Άρα, η ποια είναι η έξοδος του προγράμματος ???

```
#include <stdio.h>
int main(void)
{
        int a = 10,b = 20,c = 30;

        if(a > 5)
        {
                if(b == 20)
                        printf("1\n");

                if(c == 40)
                        printf("2\n");
                else
                        printf("3\n");
        }
        else
                printf("4\n");

        return 0;
}
```

Έξοδος: 1 3

Προτεινόμενη σύνταξη ένθετων if εντολών

- Μία πολύ συνηθισμένη χρήση των ένθετων εντολών **if** στηρίζεται στην ακόλουθη σύνταξη:

```
if(συνθήκη_A)
{
      ... /* ομάδα εντολών A */
}
else if(συνθήκη_B)
{
      ... /* ομάδα εντολών B */
}
else if(συνθήκη_C)
{
      ... /* ομάδα εντολών C */
}
.
.
.
else
{
      ... /* ομάδα εντολών N */
}
... /* επόμενες εντολές του προγράμματος. */
```

- Βάσει αυτής της σύνταξης, όταν βρεθεί μία συνθήκη που να είναι αληθής, τότε εκτελείται η ομάδα εντολών που σχετίζεται με αυτή και οι υπόλοιπες **else if** συνθήκες αγνοούνται

- Δηλαδή, η εκτέλεση του κώδικα συνεχίζει με την πρώτη εντολή που υπάρχει μετά την τελευταία **else** εντολή

Παράδειγμα

```
#include <stdio.h>
int main(void)
{
        int a;

        printf("Enter number: ");
        scanf("%d", &a);

        if(a == 1)
                printf("One\n");
        else if(a == 2)
                printf("Two\n");
        else
                printf("Other\n");

        printf("End\n");
        return 0;
}
```

Αν ο χρήστης εισάγει 1, το πρόγραμμα εμφανίζει One

Αν ο χρήστης εισάγει 2, το πρόγραμμα εμφανίζει Two

Αν εισαχθεί οποιοσδήποτε ακέραιος εκτός των 1 ή 2, το πρόγραμμα εμφανίζει Other

Σε κάθε περίπτωση, το πρόγραμμα συνεχίζει με την εκτέλεση της τελευταίας printf() και εμφανίζει End

Παρατηρήσεις

Σημειώστε ότι η τελική **else** εντολή **δεν είναι υποχρεωτικό** να υπάρχει

- Αν δεν υπάρχει, και καμία συνθήκη δεν είναι αληθής, τότε – πολύ απλά – το πρόγραμμα δεν κάνει τίποτα

- Ποια θα ήταν η έξοδος του προηγούμενου παραδείγματος αν δεν υπήρχε η τελική **else** εντολή (βλ. δίπλα) ενώ ο χρήστης εισήγαγε την τιμή 3 ???

```
#include <stdio.h>
int main(void)
{
        int a;

        printf("Enter number: ");
        scanf("%d", &a);

        if(a == 1)
                printf("One\n");
        else if(a == 2)
                printf("Two\n");

        printf("End\n");
        return 0;
}
```

Έξοδος: **End**

Παράδειγμα

- Γράψτε ένα πρόγραμμα που να διαβάζει δύο δεκαδικούς αριθμούς (π.χ., a και b) και εμφανίζει τη λύση (αν υπάρχει) της εξίσωσης:

a*x + b = 0

```
#include <stdio.h>
int main(void)
{
	double a, b;

	printf("Enter numbers: ");
	scanf("%lf%lf", &a, &b);

	if(a == 0)
	{
		if(b == 0)
			printf("Infinite solutions\n");
		else
			printf("There is no solution !!!\n");
	}
	else
		printf("The solution is %f\n", -b/a);
	return 0;
}
```

Ο τελεστής ?: (Ι)

- Ο τελεστής ?: επιτρέπει την εκτέλεση μίας από δύο ενέργειες, σύμφωνα με την τιμή μίας έκφρασης και η σύνταξή του είναι:

```
exp1 ? exp2 : exp3;
```

- Σε μία εντολή με τον τελεστή ?: αν η έκφραση **exp1** είναι αληθής, τότε θα εκτελεστεί η έκφραση που ακολουθεί το ερωτηματικό ? (δηλαδή η **exp2**), αλλιώς θα εκτελεστεί η έκφραση που ακολουθεί την άνω-κάτω τελεία : (δηλαδή η **exp3**)

- Π.χ.

```
#include <stdio.h>
int main(void)
{
        int b = 20;
        (b > 10) ? printf("One\n") : printf("Two\n");
        return 0;
}
```

- Ο τελεστής ?: χρησιμοποιείται συνήθως για να υποκαταστήσει την εντολή **if**, όταν αυτή έχει απλή μορφή

Ο τελεστής ?: (ΙΙ)

- Η τιμή μίας έκφρασης με τον τελεστή ?: είναι ίση με την τιμή της έκφρασης που εκτελείται **τελευταία**
- Ποια είναι η τιμή της μεταβλητής **max** στην παρακάτω έκφραση ;

```
max = (a > b)? a : b;
```

- Η παραπάνω έκφραση είναι ισοδύναμη με:

```
if(a > b)
        max = a;
else
        max = b;
```

- Γενικότερα, η έκφραση:

```
exp1 ? exp2 : exp3;
```

Είναι ισοδύναμη με:

```
if(exp1)
     exp2;
else
     exp3;
```

Ο τελεστής ?: (ΙΙΙ)

- Η έκφραση μετά την την άνω-κάτω τελεία : (δηλαδή η exp3) μπορεί να αντικατασταθεί από άλλη έκφραση που χρησιμοποιεί τον τελεστή ?:

- Π.χ.

```
k = exp1 ? exp2 : add1 ? add2 : add3;
```

Η παραπάνω έκφραση είναι ισοδύναμη με:

```
if(exp1)
        k = exp2;
else if(add1)
        k = add2;
else
        k = add3;
```

Παράδειγμα

- Τι κάνει το παρακάτω πρόγραμμα???

```
#include <stdio.h>
int main(void)
{
        int a;

        printf("Enter number: ");
        scanf("%d", &a);
        (a > 10) ? printf("One\n") : printf("Two\n");
        return 0;
}
```

Το πρόγραμμα διαβάζει έναν ακέραιο και αν είναι μεγαλύτερος του 10 εμφανίζει **One**, αλλιώς εμφανίζει **Two**

Η εντολή switch (Ι)

- Η εντολή ελέγχου **switch** χρησιμοποιείται εναλλακτικά έναντι της **if-else-if** δομής, όταν επιθυμούμε να ελέγξουμε μία έκφραση για όλες τις δυνατές τιμές που αυτή η έκφραση μπορεί να πάρει και να χειριστούμε την κάθε περίπτωση με διαφορετικό τρόπο

- Γενική σύνταξη της εντολής **switch**:

```
switch(έκφραση)
{
      case σταθερά_1:
      /* ομάδα εντολών που θα εκτελεστεί αν η τιμή της
      έκφρασης είναι ίση με τη σταθερά_1. */
      break;

      case σταθερά_2:
      /* ομάδα εντολών που θα εκτελεστεί αν η τιμή της
      έκφρασης είναι ίση με τη σταθερά_2. */
      break;
      ...
      case σταθερά_n:
      /* ομάδα εντολών που θα εκτελεστεί αν η τιμή της
      έκφρασης είναι ίση με τη σταθερά_n. */
      break;

      default:
      /* ομάδα εντολών που θα εκτελεστεί αν η τιμή της
      έκφρασης δεν είναι ίση με καμία από τις προηγούμενες
      σταθερές. */
      break;
}
```

Η εντολή switch (II)

- Η έκφραση που ελέγχεται πρέπει να είναι ακέραιη μεταβλητή ή έκφραση
- Οι τιμές των σταθερά_1, σταθερά_2, ... , σταθερά_n πρέπει και αυτές να είναι ακέραιες σταθερές με διαφορετικές τιμές μεταξύ των
- Τα «βήματα» κατά την εκτέλεση της εντολής **switch**:

1. Η τιμή της έκφρασης συγκρίνεται διαδοχικά με κάθε μία από τις σταθερά_1, σταθερά_2, ..., σταθερά_n
   - Αν βρεθεί μία ίδια τιμή, τότε εκτελούνται οι εντολές που ακολουθούν το αντίστοιχο **case** και στη συνέχεια γίνεται τερματισμός της εντολής **switch** μέσω της εντολής **break** (λεπτομέρειες για την εντολή **break** σε επόμενο μάθημα...)
   - Αν δεν βρεθεί ίδια τιμή, τότε εκτελούνται οι εντολές που ακολουθούν το **default** και στη συνέχεια γίνεται τερματισμός της εντολής **switch** μέσω της εντολής **break**
2. Και στις δύο περιπτώσεις, η εκτέλεση του κώδικα συνεχίζει με την πρώτη εντολή που υπάρχει μετά το άγκιστρο κλεισίματος της **switch** εντολής

Παράδειγμα

```c
#include <stdio.h>
int main(void)
{
	int a;

	printf("Enter number: ");
	scanf("%d", &a);

	switch(a)
	{
		case 1:
			printf("One\n");
		break;

		case 2:
			printf("Two\n");
		break;

		default:
			printf("Other\n");
		break;
	}
	printf("End\n");
	return 0;
}
```

Παρατηρήσεις (Ι)

- Η ύπαρξη της **default** περίπτωσης στην εντολή **switch** **δεν είναι υποχρεωτική** (όπως **δεν ήταν υποχρεωτική** και η ύπαρξη της εντολής **else** στην εντολή **if**)

- Η **default** περίπτωση μπορεί να βρίσκεται οπουδήποτε μέσα σε μία εντολή **switch** (π.χ. να είναι πρώτη ή να βρίσκεται ανάμεσα στα **case**), αν όμως υπάρχει, προτείνεται να βρίσκεται στο τέλος, δηλ. μετά από κάθε **case**

- Σε περίπτωση που δεν υπάρχει η **default** περίπτωση και η τιμή της έκφρασης δεν είναι ίση με κάποια από τις τιμές των σταθερά_1, σταθερά_2, ..., σταθερά_n, τότε γίνεται τερματισμός της εντολής **switch**, χωρίς να γίνει κάποια άλλη ενέργεια
  - Δηλαδή, η ροή του προγράμματος συνεχίζει με την εκτέλεση της πρώτης εντολής μετά το **switch**

Παρατηρήσεις (ΙΙ)

Η ύπαρξη της **break** δεν είναι υποχρεωτική, συνήθως, όμως, η παράλειψή της οδηγεί σε bug στον κώδικα

- Αν, όμως, τα μπλοκ εντολών που αντιστοιχούν σε δύο ή περισσότερες **case** περιπτώσεις **είναι κοινά**, τότε μπορεί να γίνει συνένωση των αντίστοιχων **case**

- Π.χ. αν τα μπλοκ εντολών για τις περιπτώσεις των σταθερά_1, σταθερά_2 και σταθερά_3 είναι κοινά, τότε τα αντίστοιχα **case** συνενώνονται ως εξής (έχουν, όπως βλέπουμε, κοινή **break**)

```
case σταθερά_1:
case σταθερά_2:
case σταθερά_3:
/* μπλοκ εντολών που θα εκτελεστεί αν η τιμή της έκφρασης
είναι ίση με σταθερά_1 ή σταθερά_2 ή σταθερά_3. */
break;
```

Παρατηρήσεις (III)

- Η τελευταία **break** σε μία εντολή **switch** μπορεί να παραλειφθεί, αφού ούτως ή άλλως η **switch** τερματίζεται με το άγκιστρο κλεισίματός της **}**
- Κάθε **switch** εντολή μπορεί να γραφτεί ισοδύναμα με χρήση πολλαπλών εντολών **if-else-if**
  - Σε περιπτώσεις, όμως, πολλαπλών **if-else-if** εντολών, η χρήση της **switch** οδηγεί σε πιο ευανάγνωστο κώδικα
- **ΜΕΙΟΝΕΚΤΗΜΑΤΑ** της **switch** έναντι της **if**:
  1. Η εντολή **switch** διαφέρει από την εντολή **if** στο ότι η **switch** κάνει έλεγχο μόνο για ισότητα (δηλαδή, για τιμές της έκφρασης που να είναι **ίσες** με σταθερές **case**), ενώ η συνθήκη σε μία **if** εντολή μπορεί να είναι οποιουδήποτε τύπου
  2. Οι τιμές της **έκφρασης** της **switch** και των συγκρινόμενων σταθερών πρέπει υποχρεωτικά να είναι ακέραιες

Παράδειγμα (Ι)

Ποια είναι η έξοδος του προγράμματος, αν ο χρήστης πληκτρολογήσει:

Α) 2

Β) 1

Γ) 0

```
#include <stdio.h>
int main(void)
{
        int a;

        printf("Enter number: ");
        scanf("%d",&a);

        switch(a)
        {
                case 1:
                        printf("One\n");

                case 2:
                        printf("Two\n");
                break;

                default:
                        printf("Something else\n");
                break;
        }
        printf("End\n");
        return 0;
}
```

Έξοδος:

Α)
```
Two
End
```

Β)
```
One
Two
End
```

Γ)
```
Something else
End
```

Παράδειγμα (II)

```c
#include <stdio.h>
int main(void)
{
	int month;

	printf("Enter month [1-12]: ");
	scanf("%d", &month);

	switch(month)
	{
		case 12:
		case 1:
		case 2:
			printf("Winter\n"); /* If the input value is 1, 2
or 12, the program outputs Winter. */
		break;

		case 3:
		case 4:
		case 5:
			printf("Spring\n");
		break;

		case 6:
		case 7:
		case 8:
			printf("Summer\n");
		break;

		case 9:
		case 10:
		case 11:
			printf("Autumn\n");
		break;

		default:
			printf("Error: Wrong month\n");
		break;
	}
	return 0;
}
```