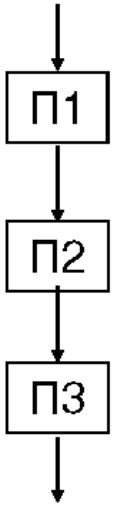
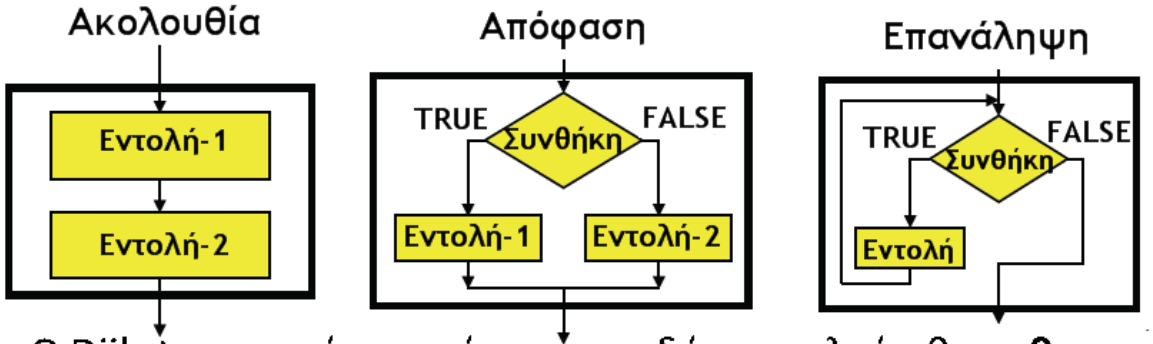
Ακολουθία προτάσεων

- Ο πιο απλός και συνηθισμένος τρόπος εκτέλεσης μιας ακολουθίας εντολών είναι ο ακολουθιακός: Η κάθε πρόταση εκτελείται μετά από την άλλη
- Οι γλώσσες δομημένου προγραμματισμού επιτρέπουν πιο ευέλικτες δομές ελέγχου ροής του προγράμματος
  - Δομή υπό-συνθήκη διακλάδωσης
  - Δομή επανάληψης



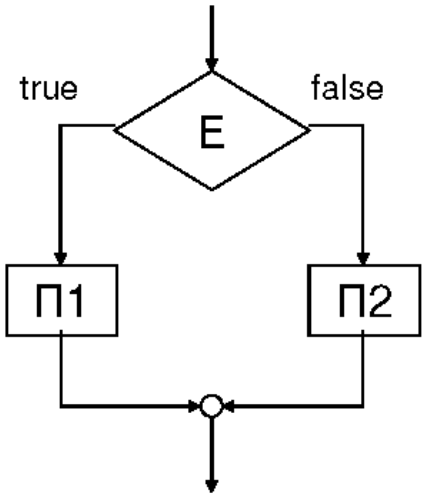
Δομές Ελέγχου

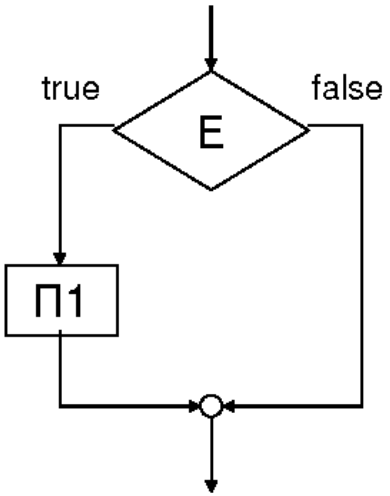
- Οι Boehm και Jacopini απέδειξαν ότι οποιοσδήποτε αλγόριθμος **μπορεί να εκφραστεί** με συνδυασμούς μόνον τριών διαφορετικών δομών ελέγχου :



- Ο Dijkstra ισχυρίστηκε ότι οποιοσδήποτε αλγόριθμος **θα πρέπει να εκφράζεται** με χρήση μόνον των τριών δομών ελέγχου (Η GOTO εντολή θα πρέπει να μην χρησιμοποιείται)
- Μια ψευδογλώσσα και οι γλώσσες προγραμματισμού παρέχουν μια ποικιλία εντολών για να εκφράσουν τις παραπάνω δομές.

Υπό-συνθήκη διακλάδωση





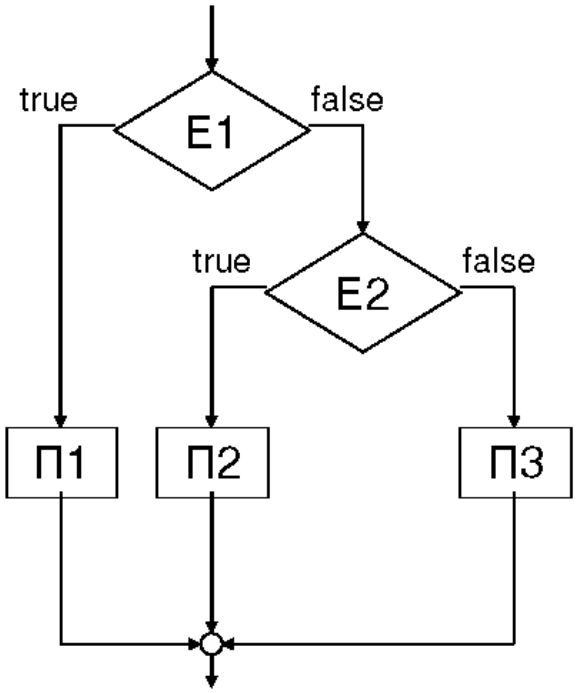
2 προτάσεις:

if E then Π1 else Π2

1 πρόταση:

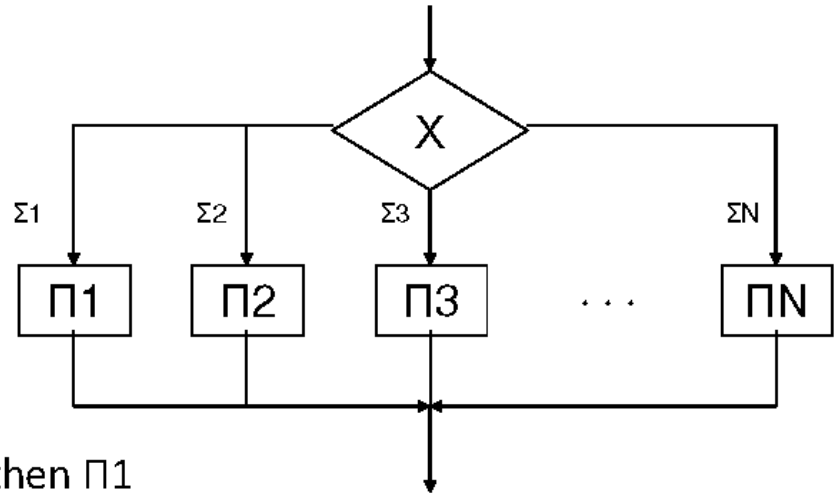
if E then Π1

Σύνθετη δομή επιλογής




```
if E1
    then Π1
else if E2
        then Π2
        else Π3
```

Δομή επιλογής (πολλαπλή)



If X=Σ1 then Π1
else if X=Σ2 then Π2
else if X=Σ3 then Π3
...
else if X=ΣN then ΠN

Προτάσεις ελέγχου ροής στη C

- Διακλάδωση υπό συνθήκη
  - if-else
  - switch case

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("x isn't zero\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("x is zero\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 **x equals 2**, αν και η αρχική τιμή της μεταβλητής **x** είναι 3

```
int x = 3;
if(x = 2)
        printf("x equals 2\n");
```

- Για να είχαμε «σωστό χειρισμό» στη συνθήκη **if**, η συνθήκη θα έπρεπε να γραφεί ως **if(x == 2)**, 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 = -3;
if(x = -2)
      printf("x equals -2\n");
```

- Και τι αυτό;

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

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

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 **A** (δηλ. οι εντολές που περιέχονται ανάμεσα στα άγκιστρα του **if**), ενώ όταν η συνθήκη είναι **ψευδής** (**false**), τότε εκτελείται η ομάδα εντολών **B** (δηλ. οι εντολές που περιέχονται ανάμεσα στα άγκιστρα του **else**)

- Π.χ.

```
int x = -3;
if(x > 0)
{
      printf("x is positive\n");
}
else
{
      printf("x is negative or zero\n");
}
```

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("x is positive\n");
else
      printf("x is negative or zero\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()
{
        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;
}
```

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()
{
        int a = 5;
      ? if(a != 5) X
            ? 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()
{
        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()
{
        int a;

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

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

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

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()
{
        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**

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()
{
        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;
```

Η εντολή switch (I)

- 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()
{
	int a;

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

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

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

		default:
			printf("Something else\n");
		break;
	}

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

Παρατηρήσεις (I)

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

- 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)

- Αν 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)

- Κάθε **switch** εντολή μπορεί να γραφτεί ισοδύναμα με χρήση πολλαπλών εντολών **if-else-if**

- 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

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

Α) 2

Β) 1

Γ) 0

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