0ι Λογικοί Τελεστές:

Πρόκειται για τελεστές που μας επιτρέπουν να συνδυάσουμε απλές εκφράσεις συσχετισμού και να δημιουργήσουμε πιό πολύπλοκες λογικές εκφράσεις. Εχουμε λοιπόν:

&& Λογικό "και" (AND) μας επιστρέφει 1 αν και οι δύο operands είναι όχι μηδέν, διαφορετικά 0)

| | Λογικό διαζευτικό "ή" (είτε - OR)

! Λογική άρνηση (NOT)

Το αποτέλεσμα μιάς έκφρασης με λογικούς τελεστές είναι είτε μηδέν (ψευδές), είτε 1 (αληθές).

(x > 10 && x < 20)

Η προτεραιότητα των τελεστών συσχέτισμού είναι μεγαλύτερη από αυτή των λογικών τελεστών ενώ η προτεραιότητα και των δύο (συσχετισμού και λογικών) είναι μικρότερη από αυτή των αριθμητικών τελεστών!

ΠΑΡΑΔΕΙΓΜΑ:Να γραφεί πρόγραμμα που να υπολογίζει αν το έτος που δίνουμε είναι δίσεκτο η όχι

```
#include <stdio.h>

void main()

{

        int year;

        do

        {

                printf("Give a year - positive integer number: ");

                scanf("%d",&year);

        }

        while (year<0); /* Αμυντικός προγραμματισμός

        if ((year%4==0 && year%100!=0) || (year%400==0))

                printf("to etos %d einai disekto",year);

        else

                printf("to etos %d den einai disekto",year);
```

}

υπενθύμιση

Δίσεκτα έτη θεωρούνται όσα διαιρούνται ακριβώς με το 4 αλλά όχι με το 100. Εξαίρεση αποτελούν τα έτη που διαιρούνται ακριβώς με το 400, που θεωρούνται επίσης δίσεκτα. Αν το έτος λοιπόν είναι δεν είναι δίσεκτο τότε η λογική έκφραση επιστρέφει την τιμή 0 (ψευδής).

ΠΑΡΑΔΕΙΓΜΑΤΑ

ΠΑΡ1: Δημιουργήστε ένα πρόγραμμα που θα ζητά από τον χρήστη δύο ακέραιους αριθμούς και θα

- εμφανίζει την λογική τους άρνηση
- ελέγχει αν κάποιος από αυτούς είναι αρνητικός
- ελέγχει αν και οι δύο είναι αρνητικοί

ΠΡΟΓΡΑΜΜΑ

```
#include <stdio.h>
int main(void)
{
int k, m;
printf("Dwse ton prwto akeraio:\n");
scanf ("%d",&k);
printf("Dwse ton deytero akeraio:\n");
scanf ("%d",&m);
printf("!%d = %d\n",k,!k);
printf("!%d = %d\n",m,!m);
printf("Kapoios arithmos einai arnhtikos -> %d\n",((k<0)||(m<0)));
printf("Kai oi dyo arithmoi einai arnhtikoi -> %d\n",((k<0)&&(m<0)));
getchar();
getchar();
return 0;
}
```

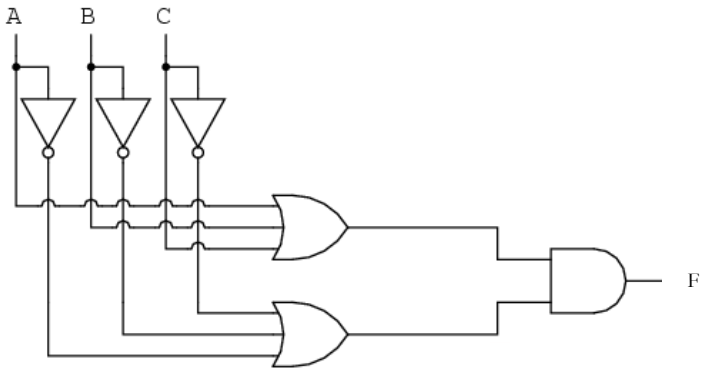
ΠΑΡ 2 :Δημιουργήστε ένα πρόγραμμα που θα ζητά από τον χρήστη δύο ακέραιους αριθμούς k (όπου 0<k<10) και m (όπου -10<m<0). Αν ο πρώτος αριθμός k βρίσκεται στο διάστημα 0<k<10 και ο δεύτερος αριθμός m βρίσκεται στο διάστημα -10<m<0 τότε θα εκτυπώνει το διπλάσιο του k και το τριπλάσιο του m. Σε διαφορετική περίπτωση θα τυπώνει το μήνυμα «Kapoios arithmos einai ektos oriwn».

ΠΡΟΓΡΑΜΜΑ

```
#include <stdio.h>
int main(void)
{
int k, m, n;

printf("Dwse ton prwto akeraio k, opou 0<k<10:\n");
scanf ("%d",&k);
printf("Dwse ton deytero akeraio m, opou -10<m<0:\n");
scanf ("%d",&m);
if (k>0 && k<10 && m<0 && m>-10) {
  printf("O k anhkei sto 0<k<10 kai to diplasio toy k einai %d\n",2*k);
  printf("O m anhkei sto -10<m<0 kai to diplasio toy m einai %d\n",2*m);
}
else
   printf("Kapoios arithmos einai ektos oriwn");
getchar();
getchar();
return 0;
}
```

ΠΑΡ 3 :Δίνεται το πιο κάτω λογικό κύκλωμα:



a.Δώστε την αντίστοιχη λογική συνάρτηση που περιγράφει τη λειτουργία του κυκλώματος.
b.Γράψτε πρόγραμμα σε γλώσσα προγραμματισμού C, το οποίο να διαβάζει τις τιμές των εισόδων A, B και C και να εμφανίζει στην οθόνη την τιμή της εξόδου F του λογικού κυκλώματος

ΑΠΑΝΤΗΣΗ

Έξοδος F = (A OR B OR C) AND ((NOT A) OR (NOT B) OR (NOT C))

Πρόγραμμα c

```
#include <stdio.h>
int a, b, c, f; // χρησιμοποιούνται ακέραιοι για να αποθηκευτούν οι λογικές μεταβλητές

main()
{
	do {
		printf("\nType valye of A (0 - 1): ");
		scanf("%d",&a);
	} while ((a!=0) && (a!=1)); //ανάγνωση του a με έλεγχο αν η τιμή είναι 0 ή 1
	do {
		printf("\nType value for B (0 - 1): ");
		scanf("%d",&b);
	} while ((b!=0) && (b!=1)); //ανάγνωση του b με έλεγχο αν η τιμή είναι 0 ή 1
	do {
		printf("\nType value for C(0 - 1): ");
		scanf("%d",&c);
	} while ((c!=0) && (c!=1)); //ανάγνωση του c με έλεγχο αν η τιμή είναι 0 ή 1
	f = (a || b || c) && (!a || !b || !c);
	printf("\nh exodos einai f = %d", f);		// εμφάνιση της εξόδου f
}
```

Οι τελεστές επιπέδου bit:

Με την C μπορούμε να κάνουμε πράξεις σε επίπεδο **bit** με τους τελεστές:

<< Ολίσθηση (shift) προς αριστερά. Για παράδειγμα η εντολή **i << j** το **i** μετακινείται αριστερά **j** bits. Οι θέσεις των bits που αδειάζουν "γεμίζουν" με 0. Προσέξτε διότι δεν λαμβάνεται υπόψη το bit του προσήμου!

>> Ολίσθηση (shift) προς δεξιά. Αν η μεταβλητή είναι unsigned τότε τα κενά bits "γεμίζουν" με 0. Αν η μεταβλητή έχει πρόσημο τότε τα bits που αδειάζουν, είτε παίρνουν το bit του προσήμου (0 ή 1 - θετική ή αρνητική μεταβλητή), είτε "γεμίζουν" με 0. Εξαρτάται από τον compiler.

~ Δυαδική άρνηση (όπου 1 μπαίνει 0 και όπου 0 μπαίνει 1 - one's complement). Υπάρχει ένας μόνο όρος

& Δυαδικό "και" σε επίπεδο bit (συξευξη / AND). Δηλαδή η εντολή **var = var & 0177**, βάζει 0 σε όλα τα bits της μεταβλητής num, εκτός από τα 7 δεξιά bits αυτής. Προσέξτε ο τελεστής εδώ απαιτεί δύο όρους. Υπάρχει και ο τελεστής της διευθυνσης (&) με τον ίδιο συμβολισμό που όταν μπαίνει μπροστά σε μία μεταβλητή αναφέρεται στην διεύθυνση της.

| Δυαδικό διαζευτικό "ή" σε επίπεδο bit (διάζευξη / OR). Η εντολή **var = var | ON**, όπου υπάρχει 1 στις θέσεις των bits της ON, μπαίνει αντίστοιχα 1 στις θέσεις των bits της var.

^ Αποκλειστική διάζευξη "ή" (exclusive OR) σε επίπεδο bit. Δηλαδή, η εντολή **var1 ^ var2** στις θέσεις που τα bits και των δύο μεταβλητών έχουν την ίδια τιμή μπαίνει 0, ενώ στις θέσεις που τα bits έχουν διαφορετική τιμή μπαίνει 1.

Οι τελεστές αυξησης/μείωσης:

Πρόκειται γιά τους τελεστές:

+ + που κάνει αύξηση μιας μεταβλητής κατά 1 μονάνδα π.χ **i++;**

- - που κάνει μείωση μιας μεταβλητής κατά 1 π.χ **i-- ;**

Εχει σημασία αν οι τελεστές αύξησης/μείωσης τεθούν πρίν (++i) ή μετά (i--) την μεταβλητή.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>
main()
{
 int a=3, b=3;
 int syna, synb;
 syna=a++;
 synb=++b;
 printf ("a= %d, syna= %d, b= %d, synb=%d\n", a, syna, b, synb);
}
```

το αποτέλεσμα που θα πάρουμε από την εντολή **printf** θα είναι:
a=4, syna=3, b=4, synb=4
και αυτό διότι στην εντολή **syna=a++;** γίνεται χρήση της τιμής της μεταβλητής **a,** πριν γίνει η αύξηση κατά μία μονάδα, ενώ στην **synb=++b;** μετά την αύξηση της μεταβλητής **b** κατά μία μονάδα.

Οι τελεστές εκχώρησης:

+= Πρόσθεση σε μεταβλητή. Παράδειγμα: **x+=1**; είναι το ιδίο όπως x=x+1

-= Αφαίρεση από μεταβλητή. Παράδειγμα: **x-=1**; (όπως x=x-1)

*= Πολλαπλασιασμός μεταβλητής. Παράδειγμα: **x*=3**; (όπως x=x*3)

/= Διαίρεση μεταβλητής. Παράδειγμα: **x/=2**; (όπως x=x/2)

%= Το υπόλοιπο. Παράδειγμα: **alpha %=2** (οπως alpha=alpha %2)

>>= Ολίσθηση δεξιά. Παράδειγμα: **beta >>= 3** (τα bits της beta ολίσθησαν δεξιά κατα 3 θέσεις)

<<= Ολίσθηση αριστερά. Παράδειγμα: **beta <<= 3** (τα bits της beta ολίσθησαν αριστερά κατα 3 θέσεις)

&= Σύζευξη (AND). Παράδειγμα:**var &= 0177** (όπως var = var & 0177)

|= Διάζευξη (OR). Παράδειγμα: **var | = 0177**

^= Αποκλειστική διάζευξη (exclusive OR). Παράδειγμα: **var | = 0177**

Ο τελεστής διευθυνσης (&):

Μας δίνει την διευθύνση της μνήμης που αποθηκευεται μιά μεταβλητή. Αν για παράδειγμα έχω μία ακέραια μεταβλητή ALPHA ο συμβολισμός &ALPHA μας δίνει την διεύθυνση της μεταβλητής στην μνήμη. Ετσι αν η μεταβλητή ALPHA περιέχει τον αριθμό 32 (alpha=32) και αποθηκεύεται στην διεύθυνση 4294967246, τότε η εκτέλεση της εντολής:

printf ("Value of ALPHA:%d, Address of ALPHA:%u \n", ALPHA, &ALPHA);

θα μας δώσει:

Value of ALPHA:32, Address of ALPHA:4294967246

Προσέξτε, για να εμφανιστεί σωστά η διεύθυνση χρησιμοποιήθηκε ο προσδιοριστής %u (χωρίς πρόσημο ακέραιος).

Ο τελεστής συνθήκης η τριαδικός τελεστής :

Γενική μορφή: **έκφραση ? εντολή-1 : εντολή-2**
Αν η (λογική) **έκφραση** είναι αληθής τότε αναπτύσσεται η **εντολή-1**, διαφορετικά αναπτύσσεται η **εντολή-2**.
Ουσιαστικά πρόκειται για μία συμπαγής μορφή της εντολής **if-else**:

```
if (έκφραση)
      εντολή-1
else
      εντολή-2
```

ΠΑΡΑΔΕΙΓΜΑΤΑ

ΠΑΡ1.

```
num = (x<=5) ? 1 : 0;
```

Η μεταβλητή **num** θα πάρει την τιμή **1** αν το **x** είναι μικρότερο ή ίσο του **5**, διαφορετικά θα πάρει την τιμή **0.**

ΠΑΡ2.

```
 #include<stdio.h>
int main()
{
        int x=5,y=10,max;
        max = (x > y)? x : y; // στην μεταβλητή max θα εκχωρηθεί το y
αφού είναι μεγαλύτερο του x
        printf("The maximum number is %d.\n",max);
}
```

Προτεραιότητες μεταξύ των τελεστών της C:

Ιεραρχία	Τελεστές	Φορά
1	( ) [ ] . ->	Από αριστερά πρός δεξιά
2	! ~ ++ -- &(διεύθυνση) *(indirection) (type) sizeof	Από δεξιά πρός αριστερά
3	*(πολλαπλασιασμός) / %	Από αριστερά πρός δεξιά
4	+ -	Από αριστερά πρός δεξιά
5	<< >>	Από αριστερά πρός δεξιά
6	< <= > >=	Από αριστερά πρός δεξιά
7	== !=	Από αριστερά πρός δεξιά
8	& (AND)	Από αριστερά πρός δεξιά
9	^	Από αριστερά πρός δεξιά
10	|	Από αριστερά πρός δεξιά
11	&&	Από αριστερά πρός δεξιά
12	||	Από αριστερά πρός δεξιά
13	? :	Από δεξιά πρός αριστερά
14	= += -= *= /= %= &= ^= |= <<= >>=	Από δεξιά πρός αριστερά
15	,	Από αριστερά πρός δεξιά

Η C υποστηρίζει τις παρακάτω δομές ελέγχου:

Εντολή if-else:

Οι εντολές **if** και **if-else** υπάρχουν σχεδόν σε όλες τις γλώσσες προγραμματισμού. Χρησιμοποιούνται για να ελέγξουν να ισχύει ή όχι κάποια συνθήκη. Στην **C** η γενική σύνταξη της εντολής είναι:

```
if (<ΣΥΝΘΗΚΗ>)
        Ενότητα -Α
else
        Ενότητα-Β
```

Η **<ΣΥΝΘΗΚΗ>** μπορεί να είναι λογική έκφραση, έκφραση συσχετισμού, αποτέλεσμα κάποιας πράξης, είτε ακόμα και κάποια μεταβλητή. Η **Ενότητα-Α** και η **Ενότητα-Β** μπορεί να περιλαμβάνουν μία εντολή ή πολλές εντολές (block) που περικλείονται σε άγκιστρα ({,}). Οι εντολές (ή η εντολή) της **Ενότητας-Α** εκτελούνται αν η **<ΣΥΝΘΗΚΗ>** είναι αληθής (όχι 0), ενώ οι εντολές (ή η εντολή) της **Ενότητας-Β** εκτελείται αν η **<ΣΥΝΘΗΚΗ>** είναι ψευδής (0-μηδέν). Το **else** είναι προαιρετικό και όταν υφίσταται αναφέρεται στο πλησιέστερο πριν από αυτό **if** που δεν έχει **else**. Κάθε ενότητα είναι δυνατόν να περικλείει και άλλες **if-else** ενότητες (blocks) εντολών. Παραδείγματα:

Προγραμμα -Α

```
int num;

.....

if (num = = 2)
  printf("The num is now: 2\n");

if (num < 5)
  printf("The num is now %d, less than
5\n",num);
else
  printf("The num is now %d, greater than
4\n",num);
```

Προγραμμα -Β

```
int num;

.....

if (num = = 2)
  printf("The num is now: 2\n");
else if (num < 5)
          printf("the num is now less than 5\n");
       else
          printf("the num is now  greater than
4\n");
```

Προγραμμα -Γ

```
int a,b,c;

.....

if (a>b)
     if (a>c)
          printf ("max is a:%d", a);
     else
          printf ("max is c:%d", c);
 else
      if (b>c)
           printf ("max is b:%d",b);
      else
           printf("max is c:%d",c);
```

Προγραμμα -Δ

```
int a,b,c;

.....

if (a>b  &&  a>c)
          printf ("max is a:%d", a);
if (b>a  &&  b > c)
          printf ("max is b:%d", b);
if (c> a  &&  c > b)
           printf ("max is c:%d",c);
```

Εντολή switch:

Η γενικευμένη μορφή της εντολής είναι:

```
switch (έκφραση)
{
   case σταθερά1:
                εντολή1;
                break;
   case σταθερά2:
                εντολή2;
                break;
   default:
            εντολήN;
            break;
}
```

Εκτελείται η εντολή της οποίας η σταθερά ταιριάζει με την τιμή της έκφρασης. Η εκτέλεση του προγράμματος μέσα σε ένα **switch** συνεχίζεται στο επόμενο **case**, εκτός αν δεν μεσολαβεί κάποια από τις εντολές **break**, **exit** ή **return**. Αν η τιμή δεν ταιριάζει με καμία σταθερά, τότε εκτελείται η εντολή στο block της **default** (που είναι προαιρετικό).

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

printf("Enter an integer: ");
scanf("%d", &a);
/* Check values of a */
switch (a)
{
case 1:
   puts("You entered 1");
   break;
case 2:
   puts("You entered 2");
   break;
case 3:
   puts("You entered 3");
   break;
case 4:
   puts("You entered 4");
   break;
case 5:
   puts("You entered 5");
   break;
case 6:
   puts("You entered 6");
   break;
default:
   puts("You enter an integer less than 1 or more than 6");
}
}
```

ΠΑΡΑΔΕΙΓΜΑ

Δημιουργήστε ένα πρόγραμμα που να υλοποιεί μια υποτυπώδη αριθμομηχανή που μπορεί να εκτελέσει τις πράξεις της πρόσθεσης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης δύο πραγματικών αριθμών με χρήση της εντολής switch.

ΠΡΟΓΡΑΜΜΑ

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

float num1, num2;
char op;
printf("MENOU EPILOGWN\n");
printf("Dwse + gia thn prothsesi\n");
printf("Dwse * gia thn pollaplasiasmo\n");
printf("Dwse / gia diairesi\n");
printf("Dwse - gia afairesi\n");
printf("Dwse <arithmo1> praksi <arithmo2>: ");
scanf("%f %c %f", &num1, &op, &num2 );
switch (op)
{
  case '+': case 'A': printf ( " = %f", num1 + num2 );
            break;
  case '-': case 'B':  printf ( " = %f", num1 - num2 );
            break;
  case '*': case 'C': printf ( " = %f", num1 * num2 );
            break;
  case '/': case 'D': printf ( " = %f", num1 / num2 );
            break;
  default: printf ( "Lathos eisodos\n\n" );
}
getchar();
getchar();
return 0;
}
```

Εντολή break:

Η εντολή **break** προκαλεί τον τερματισμό μιάς εντολής **switch** ή και μιάς επανάληπτικής διαδικασίας (που προκλήθηκε με εντολή **for** ή την **while** - δείτε την επόμενη ενότητα). Η εκτέλεση του προγράμματος συνεχίζεται μετά την εντολή **switch** ή την ενολή επανάληψης.

```
while(...)
{

. . .

break;

. . .

}

. . . /* Η εκτέλεση του προγράμματος συνεχίζεται μετά το break */
```