ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2017-2018

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ

ΜΑΘΗΜΑ: ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΤΗΡΙΟ 1 - ΣΗΜΕΙΩΣΕΙΣ

Εισαγωγή

Ένα πρόγραμμα σε C περιλαμβάνει μια ή περισσότερες συναρτήσεις οι οποίες είναι ένα σύνολο εντολών που στοχεύουν στην εκτέλεση μιας συγκεκριμένης εργασίας. Η ακόλουθη εικόνα παρουσιάζει τη γενική δομή ενός προγράμματος σε C.

```
main()
{
        Statement 1;
        Statement 2;
        .............
        .............
        Statement N;
}
Function1()
{
        Statement 1;
        Statement 2;
        .............
        .............
        Statement N;
}
Function2()
{
        Statement 1;
        Statement 2;
        .............
        .............
        Statement N;
}
...................
...................
FunctionN()
{
        Statement 1;
        Statement 2;
        .............
        .............
        Statement N;
}
```

Όπως κάθε γλώσσα προγραμματισμού έτσι και η C περιλαμβάνει ένα σύνολο **δεσμευμένων λέξεων** που απεικονίζονται στον ακόλουθο πίνακα:

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Επίσης, η C προσφέρει ένα σύνολο **τύπων δεδομένων (data types)** των μεταβλητών ως ακολούθως:

Data Type	Size in Bytes	Range	Use
char	1	-128 to 127	To store characters
int	2	-32768 to 32767	To store integer numbers
float	4	3.4E-38 to 3.4E+38	To store floating point numbers
double	8	1.7E-308 to 1.7E+308	To store big floating point numbers

Για διευκόλυνση των προγραμματιστών, ο κάθε τύπος δεδομένων περιλαμβάνει κάποιες υποκατηγορίες που είναι οι εξής:

Data Type	Size in Bytes	Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int	2	0 to 65535
signed int	2	-32768 to 32767
short int	2	-32768 to 32767
unsigned short int	2	0 to 65535
signed short int	2	-32768 to 32767
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
signed long int	4	-2147483648 to 2147483647
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Τα ακόλουθα παραδείγματα μας δείχνουν τον τρόπο δήλωσης και αρχικοποίησης μεταβλητών:

```
int emp_num;
float salary;
char grade;
double balance_amount;
unsigned short int acc_no;
int emp_num = 7;
float salary = 9800.99
char grade = 'A';
double balance_amount = 100000000;
```

Οι σταθερές αρχικοποιούνται ως εξής:

```
const float pi = 3.14;
```

Τέλος, οι ακόλουθοι πίνακες παρουσιάζουν τους τελεστές που μπορεί να χρησιμοποιηθούν στη γλώσσα C:

Operation	Operator	Syntax	Comment	Result
Multiply	*	a * b	result = a * b	27
Divide	/	a / b	result = a / b	3
Addition	+	a + b	result = a + b	12
Subtraction	-	a - b	result = a - b	6
Modulus	%	a % b	result = a % b	0

Operator	Meaning	Example
<	Less than	3 < 5 gives 1
>	Greater than	7 > 9 gives 0
<=	Less than or equal to	100 <= 100 gives 1
>=	Greater than equal to	50 >=100 gives 0

Operator	Meaning
==	Returns 1 if both operands are equal, 0 otherwise
!=	Returns 1 if operands do not have the same value, 0 otherwise

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

A	! A
0	1
1	0

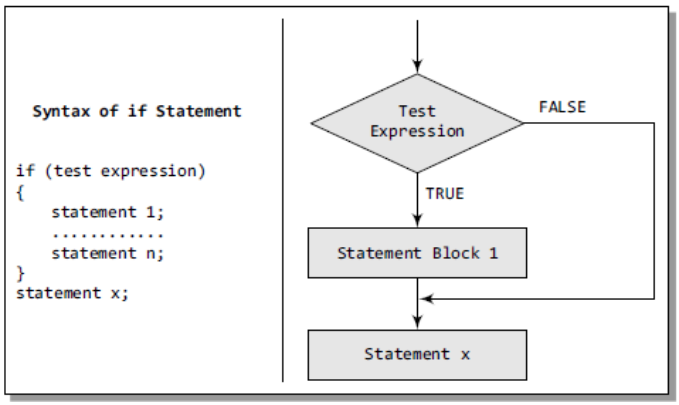
A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

Τέλος, το ακόλουθο παράδειγμα μας παρουσιάζει ένα πρόγραμμα C το οποίο μας υπολογίζει το εμβαδόν ενός κύκλου.

Code 1

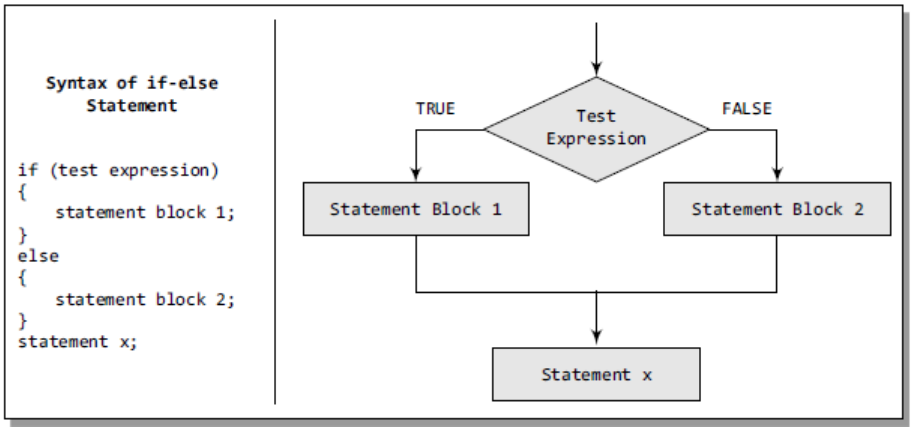
```
#include <stdio.h>
#include <conio.h>
int main()
{
        float radius;
        double area;
        clrscr();
        printf("\n Enter the radius of the circle : ");
        scanf("%f", &radius);
        area = 3.14 * radius * radius;
        printf(" \n Area = %.2lf", area);
        return 0;
}
```

Σύντομη Ανασκόπηση των Εντολών της C



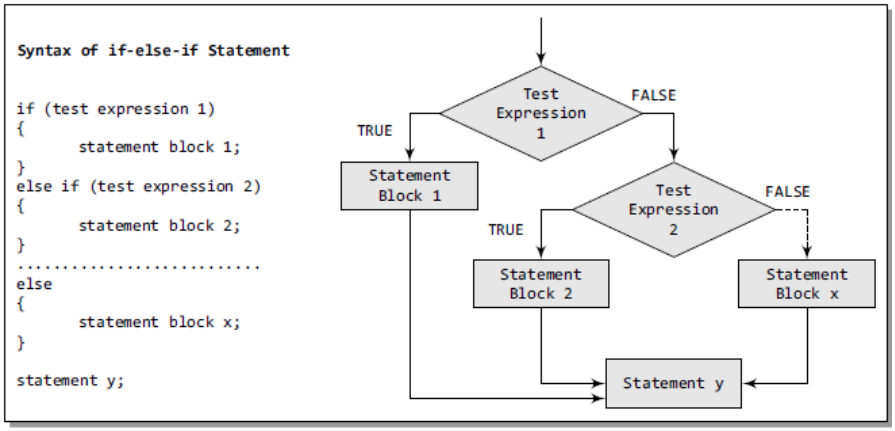
Code 2

```
#include <stdio.h>
int main()
{
        int x=10;
        if (x>0) x++;
                printf("\n x = %d", x);
        return 0;
}
```



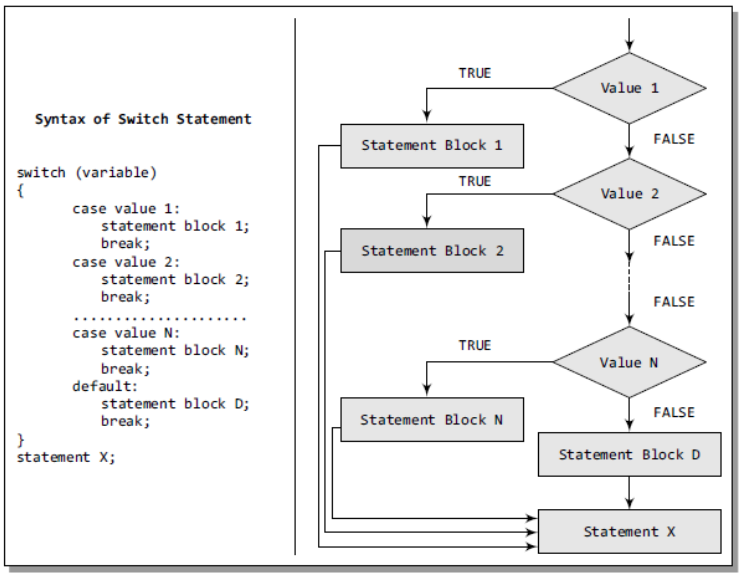
Code 3

```
#include <stdio.h>
int main()
{
        int a;
        printf("\n Enter the value of a : ");
        scanf("%d", &a);
        if(a%2==0)
                printf("\n %d is even", a);
        else
                printf("\n %d is odd", a);
        return 0;
}
```



Code 4

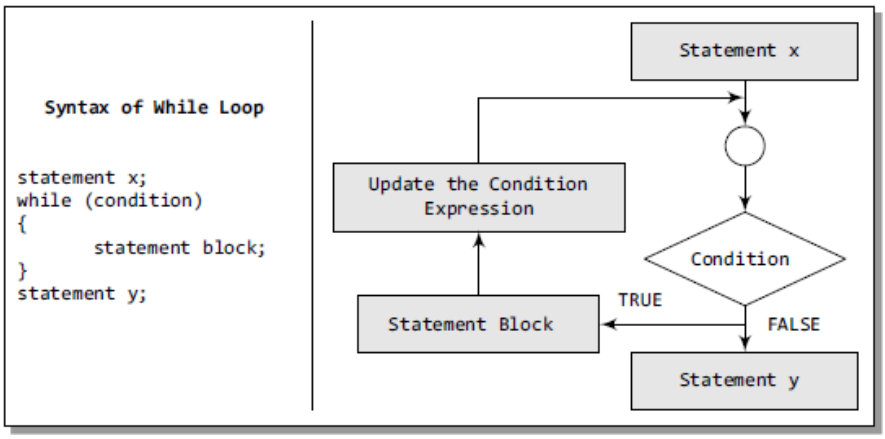
```
#include <stdio.h>
int main()
{
        int num;
        printf("\n Enter any number : ");
        scanf("%d", &num);
        if(num==0)
                printf("\n The value is equal to zero");
        else if(num>0)
                printf("\n The number is positive");
        else
                printf("\n The number is negative");
        return 0;
}
```



Code 5

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

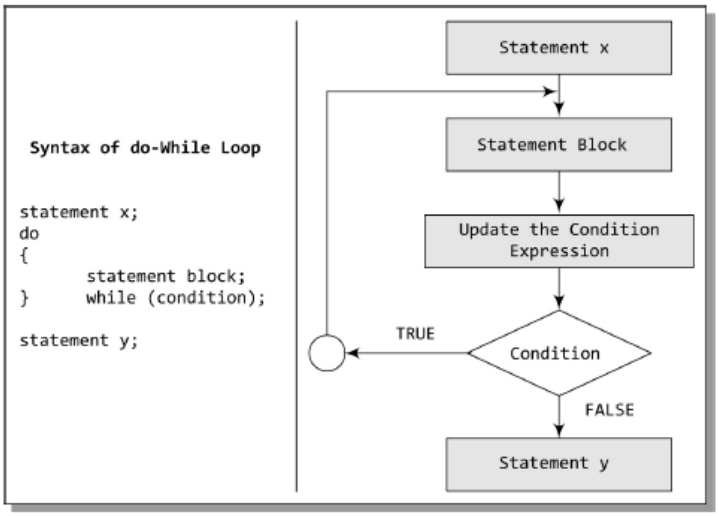
```
        char ch;
        printf("\n Enter any character : ");
        scanf("%c", &ch);
        switch(ch)
        {
                case ‘A’:
                case ‘a’:
                        printf("\n %c is VOWEL", ch);
                        break;
                case ‘E’:
                case ‘e’:
                        printf("\n %c is VOWEL", ch);
                        break;
                case ‘I’:
                case ‘i’:
                        printf("\n %c is VOWEL", ch);
                        break;
                case ‘O’:
                case ‘o’:
                        printf("\n %c is VOWEL", ch);
                        break;
                case ‘U’:
                case ‘u’:
                        printf("\n %c is VOWEL", ch);
                        break;
                default: printf("\n %c is not a vowel", ch);
        }
return 0;
}
```



Code 6

```
#include <stdio.h>
```

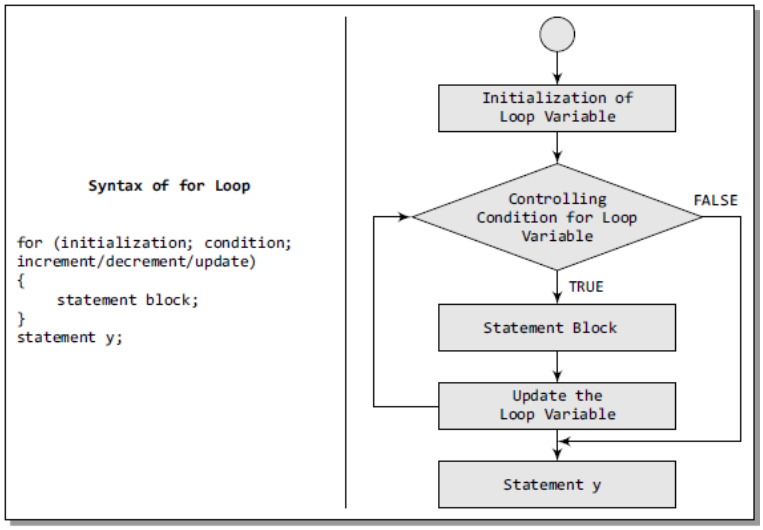
```
int main()
{
        int n, m, i, sum =0;
        printf("\n Enter the value of m : ");
        scanf("%d", &m);
        i=m;
        printf("\n Enter the value of n : ");
        scanf("%d", &n);
        while(i<=n)
        {
                sum = sum + i;
                i = i + 1;
        }
        printf("\n The sum of numbers from %d to %d = %d", m, n, sum);
        return 0;
}
```



Code 7

```
#include <stdio.h>
int main()
{
        int n, i = 0, sum =0;
        float avg = 0.0;
        printf("\n Enter the value of n : ");
        scanf("%d", &n);
        do
        {
                sum = sum + i;
                i = i + 1;
        } while(i<=n);
```

```
    avg = (float)sum/n;
    printf("\n The sum of first %d numbers = %d",n, sum);
    printf("\n The average of first %d numbers = %.2f", n, avg);
    return 0;
}
```



Code 8

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int flag = 0, i, num;
    clrscr();
    printf("\n Enter any number : ");
    scanf("%d", &num);
    for(i=2; i<num/2;i++)
    {
        if(num%i == 0)
        {
            flag =1;
            break;
        }
    }
    if(flag == 1)
        printf("\n %d is a composite number", num);
    else
        printf("\n %d is a prime number", num);
    return 0;
}
```

Code 9

```
#include <stdio.h>
int main()
{
        int i = 0;
        while(i<=10)
        {
                if (i==5)
                        break;
                printf("\t %d", i);
                i = i + 1;
        }
        return 0;
}
```

Code 10

```
#include <stdio.h>
int main()
{
        int i;
        for(i=0; i<= 10; i++)
        {
                if (i==5)
                        continue;
        printf("\t %d", i);
        }
        return 0;
}
```

Συναρτήσεις

Μια συνάρτηση είναι ένα 'μαύρο κουτί' το οποίο περιλαμβάνει εντολές που επιτελούν κάποιες ενέργειες. Η δήλωση μιας συνάρτησης στη C γίνεται ως εξής:

```
return_data_type function_name(data_type variable1, data_type variable2,..)
{
        .............
        statements
        .............
        return(variable);
}
```

Ενώ η κλήση της:

```
variable_name = function_name(variable1, variable2, ...);
```

Code 11

```
#include <stdio.h>
int evenodd(int); //FUNCTION DECLARATION
```

```
int main()
{
	int num, flag;
	printf("\n Enter the number : ");
	scanf("%d", &num);
	flag = evenodd(num); //FUNCTION CALL
	if (flag == 1)
		printf("\n %d is EVEN", num);
	else
		printf("\n %d is ODD", num);
	return 0;
}

int evenodd(int a) // FUNCTION HEADER
{
	// FUNCTION BODY
	if(a%2 == 0)
		return 1;
	else
		retun 0;
}
```

Για το πέρασμα παραμέτρων στις συναρτήσεις υπάρχουν δύο τρόποι:

A. Call by value. Η συνάρτηση που καλέι περνά μόνο τις τιμές των μεταβλητών. Οποιαδήποτε αλλαγή γίνεται σε αντίγραφα των μεταβλητών και όχι στις ίδιες τις μεταβλητές.

B. Call by reference. Η συνάρτηση που καλεί περνά τις διευθύνσεις των μεταβλητών. Οποιαδήποτε αλλαγή γίνεται στις ίδιες τις μεταβλητές.

Code 12

```
#include <stdio.h>
void add(int n);
int main()
{
	int num = 2;
	printf("\n The value of num before calling the function = %d", num);
	add(num);
	printf("\n The value of num after calling the function = %d", num);
	return 0;
}
void add(int n)
{
	n = n + 10;
	printf("\n The value of num in the called function = %d", n);
}
```

Code 13

```
#include <stdio.h>
void add(int *);
int main()
{
	int num = 2;
	printf("\n The value of num before calling the function = %d", num);
	add(&num);
	printf("\n The value of num after calling the function = %d", num);
	return 0;
}
void add(int *n)
{
	*n = *n + 10;
	printf("\n The value of num in the called function = %d", *n);
}
```

Δείκτες (pointers)

Κάθε μεταβλητή έχει ένα όνομα και μια τιμή. Μετά τη δήλωση της μεταβλητής, ένα τμήμα μνήμης δεσμεύεται για αυτή. Έτσι, κάθε μεταβλητή έχει τιμή και μια διεύθυνση μνήμης στην οποία έχει εκχωρηθεί η τιμή αυτή. Οι **δείκτες (pointers)** είναι μεταβλητές που περιέχουν τη διεύθυνση μνήμης μιας άλλης μεταβλητής. Η δήλωση των δεικτών γίνεται ως εξής:

```
data_type *ptr_name;
```

Παραδείγματα:

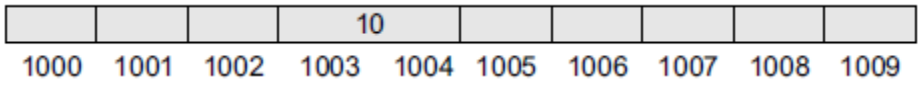
```
int *pnum;
char *pch;
float *pfnum;
```

Ακόμα και αν οι δείκτες 'δείχνουν' σε διαφορετικού τύπου δεδομένα, θα καταλάβουν τον ίδιο χώρο στη μνήμη αφού στην ουσία αποθηκεύουν διευθύνσεις μνήμης και όχι τιμές. Η χρήση του * ειδοποιεί το μεταγλωττιστή ότι πρόκειται για μια μεταβλητή τύπου δείκτη.

```
int x= 10;
int *ptr;
ptr = &x;
```

Στο παραπάνω παράδειγμα, το ptr δείχνει σε μια ακέραια μεταβλητή που καταλαμβάνει 2 bytes στη μνήμη. Το σύμβολο **&** ανακτά τη διεύθυνση μνήμης της μεταβλητής x.

Αν η μνήμη έχει την εξής μορφή:



Ο δείκτης θα πάρει την τιμή 1003.

Code 14

```
#include <stdio.h>
int main()
{
	int num, *pnum;
	pnum = &num;
	printf("\n Enter the number : ");
	scanf("%d", &num);
	printf("\n The number that was entered is : %d", *pnum);
	return 0;
```

```
}
```

Code 15

```
#include <stdio.h>
int main()
{
        int x=10;
        char ch = 'A';
        void *gp;
        gp = &x;
        printf("\n Generic pointer points to the integer value = %d", *(int*)gp);
        gp = &ch;
        printf("\n Generic pointer now points to the character= %c", *(char*)gp);
        return 0;
}
```

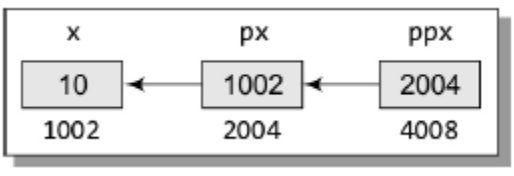
Code 16

```
#include <stdio.h>
void sum (int*, int*, int*);
int main()
{
        int num1, num2, total;
        printf("\n Enter the first number : ");
        scanf("%d", &num1);
        printf("\n Enter the second number : ");
        scanf("%d", &num2);
        sum(&num1, &num2, &total);
        printf("\n Total = %d", total);
        return 0;
}
void sum (int *a, int *b, int *t)
{
        *t = *a + *b;
}
```

Τέλος, στη C μπορούμε να χρησιμοποιήσουμε δείκτες στο να δείχνουν σε άλλους δείκτες. Η δήλωση γίνεται μέσω της χρήσης του συμβόλου * σε κάθε επίπεδο των δεικτών.

Παράδειγμα:

```
int x=10;
int *px, **ppx;
px = &x;
ppx = &px;
```



Σε αυτό το παράδειγμα, η εντολή

```
printf("\n %d", **ppx);
```

Θα τυπώσει το 10.

Διάφορα κοινά λάθη που συμβαίνουν με τους δείκτες έχουν ως ακολούθως:

```
int x, *px;
x=10;
*px = 20;
```

Ο δείκτης px δεν έχει οριστεί και με την τρίτη εντολή θα γράψει στα περιεχόμενα της αντίστοιχης θέσης μνήμης την τιμή 20.

```
int x, *px;
x=10;
px = x;
```

Η σωστή εντολή είναι px = &x;

```
int x=10, y=20, *px, *py;
px = &x, py = &y;
if(px<py)
printf("\n x is less than y");
else
printf("\n y is less than x");
```

Η σωστή εντολή είναι *px < *py