

Unit 1

Programming using C language
Dr. Nivedita Palia

Repetition Structures: Types

Allows the programmer to specify that an action is to be repeated ***while some condition remains true.***

There are three repetition structures in C: -

the **while loop**

the **for loop**

do-while loop.

For Loop

The syntax of for loop is

```
for(initialisation; condition; l teration)
{
    set of statements;
}
```

Eg: Program to print Hello 10 times

```
for(i=1;i<=10;i++)
{
    printf("Hello");
}
```

Important points related to for loop

1. No braces required for a single statement in the body of for loop
2. No semicolon after for statement
3. loop counter Increment within the body of for loop also , in spite of this the semicolon after condition is necessary
4. Initialisation is done before for loop, but still the semicolon before condition is necessary
5. increment and comparison done at same statement
6. Multiple initialisation and increment allow but single condition

Nesting of for loops

```
for(i=0;i<3;i++)  
{  
    for(j=0;j<2;j++)  
        printf("i=%d\tj=%d\n",i,j);  
}
```

Question

1. WAP to find the factorial of a given number.
2. WAP to check whether the given number is prime
3. Print n natural number and find its sum
4. find the value of one number raised to the power of another
5. write a program to print the following pattern

```
1                1
12              2 3
123            4 5 6
1234
```

While loop

The syntax for while loop

Initialisation; //part 1

while(condn) //part 2

{

statements;

iteration; (increment/decrement) //part3

}

Eg:

a=10; //part1

while(a !=1) //part 2

{

printf("%d",a);

a- -; //part 3

}

Do-While Loop – The Odd Loop

The syntax of do while loop

```
do  
{  
    set of statements  
}while(condn);
```

Eg:

```
i=1;  
do  
{  
    printf(“%d”,i); //1  
    i--;           //0  
}while(i!=0);
```


Do-while Vs while loop

- 1) Do-while will be executed minimum once, but while loop can have minimum execution as 0 times.
- 2) Do-while is an **exit** control loop, but while are **entry** control loop.
- 3) While loop is not terminated by semi colon. But do-while is terminated by semicolon.
- 4) Variable initialisation happens before the loop in case of while loop, but may be done inside the loop body in case of do-while loop.

Syntax:


while(c1)	do {
{	
	}while(c1);
}	

Break Statement

Break Statement is used to take the control out of a block.

```
for(i=0;i<=10;i++)  
executed 6 to 10)  
{  
    printf("%d",i);  
    if(i==5)  
        break;  
} //for
```


//i=0,1,2...5 (not
// 0 1 2 3 4 5



Continue Statement

Continue is used to bypass(skip) a set of statements (written below continue) for a particular iteration.

```
for(i=0;i<=10;i++)                                //i=0 1...5 6
{
    if(i!=5) //T                                     //i=5
    continue;
    printf("%d",i);                                  // 5
} //for
```



Break Vs Continue

1. Break takes the control out of a particular block, but continue skips rest of the statements for a particular condition.
2. Break exits the loop completely but in case of continue the loop keeps executing with next iteration.

Break statement works in loop as well as switch statement. but continue works with loops.

Example:

```
for(j=1;j<=3;j++)  
{  
printf("%d\t%d",i,j);  
if(j==2)
```

```
break ;
```

example of continue

```
} //for 1
```

// repeat same code but replace break with continue for

Goto Statement

Goto is used for unconditional/conditional branching.
It requires a label (identifier) where a branch is to be made.
It can have two forms.

Syntax: (Forward Jump)

Goto **label**;

.....

.....

Label:

Statements;



Go to statement

Syntax: (backward Jump)

Label:

Statements;

.....

.....

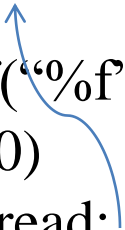
goto **label**;



Example: Goto

```
#include<math.h>
#include<stdio.h>
void main()
{
    float x,y;
    read:
    scanf("%f",&x);
    if(x<0)
    goto read;
    y= sqrt(x);
    printf("%f%f",x,y);

}
```



exit()

- exit() is a library function to jump out of a program.
- Integer value as argument
- Zero indicate normal termination
- Non-zero indicate abnormal termination
- Prototype in <stdlib.h>

```
.....  
.....  
if (test-condition) exit(0) ;  
.....  
.....
```


END