Procedural Programming – I

UNIT – II

Conditional Statements-

- Conditional statements are used to execute a set of statements on some conditions.
- It can used to provide the conditions in the program.
- The conditions can be given using operator.
- It checks the given condition and then executes it's sub-blocks.
- The decision statement decides the statement to be executed after the success or failure of a given condition.

- **Types**
  There are two types.
  1) if statement
  2) switch statement

*** If statements-**

C uses the keyword 'if' to implement the conditional statement (decision control instruction).

Syntax –

```
If(condition)
{
Execute statement;
}
```

Example-

```
// Demonstrate of if statement
#include<stdio.h>
#include<conoio.h>
void main()
{
int num;
printf ("enter a number less than 10")
scanf("%d",&num);
if(num<=10)
{
printf("what an obedient servent you are!");
}
getch();
}
```

If you type number less than or equal to 10, you get a massage on the screen through printf() as "what an obedient servent you are!". If you type some other number no output will be printed. Here the if statement will not have a ';' as the scope of the statement continue further.

Therefore The if statement has three forms-

1) Simple if
2) Multiple if
3) Nested if

*** The simple if(if else statement)-**

If else statement can be used when there is only one condition.

Syntax –

```
If(condition)
{
Group of statement1;
}
Else
{
Group of statement2;
}
```

If the condition given with if statement is true then statement1 will be executed otherwise else part will be executed.

Else part is optional if there are multiple statement given in if block then enlcosedin {}.

Example-

```
//Program to find greater number between two numbers entered.
#include<stdio.h>
#include<conio.h>
void main()
{
 int num1,num2;
 clrscr();
 printf("Enter first number:\n");
 scanf("%d",&num1);
 printf("enter second number:\n");
 scanf("%d",&num2);
 if(num1>num2)
 {
  printf("num1 is greater than num2");
 }
 else
 {
 printf("num2 is greater than num1");
 }
```

```
getch();
}
```

*** Multiple if (if else-if)-**

Multiple if statement can be used when we want to be perform multiple operation depending on different conditions.

Syntax-

```
if(condition1)
{
statement1;
}
else if(condition2)
{
statement2;
}
else if(condition3)
{
statement3;
}
:
:
else if(conditionN)
{
statementN;
}
else
statementX;
```

like this we can give number of conditions and we can execute different statement. Conditions will be checked one by one. If condition is satisfying statement will be executed and control will be transferred to the end of ifstatement, if it false then only condition2 will be checked. Among all the given conditions any one will be satisfied at a time. If all of the given conditions false then final else block will be executed.

Example-

```
// Program to print grade of students based on percentage
#include<stdio.h>
#include<conio.h>
void main()
{
 int S1,S2,S3,S4,S5,total;
float per;
 clrscr();
 printf("enter your marks");
 scanf("%d %d %d %d %d",&S1,&S2,&S3,&S4,&S5);
```

```
 total=S1+S2+S3+S4+S5;
printf("your total of marks is: %d/n",total);
per=(total/500)*100;
 printf("your petcentage is: %d\n",per);
 if(per<=100 && per>=80)
 {
   printf("your grade is O");
 }
 else if(per<80 && per>=70)

{
   printf("your grade is A");
 }
 else if(per<70 && per>=60)
 {
   printf("your grade is B");
 }
 else if(per<60 && per>=50)
 {
   printf("your grade is C");
 }
 else if(per<50 && per >=40)
 {
  printf("your grade is D");
 }
 else
 {
  printf("you are fail");
 }
getch();
}
```

In above example it will be accept the percentage and grade will be displayed by checking condition.

*** Nested – if-**

If is perfectly all right if we write an entire 'if-else' construct within anather if block or else block. This called 'nesting' of its.
If in if condition is called as nested.

Syntax-

```
if(condition)    //outer if
{
  if(condition)     //inner if
{
statement;
```

```
}
else
statement;
}
else
Statement;
```

In this case the second if statement is known as inner if and main if statement is known as outer if. The inner if condition will be checked only the outer if condition is satisfying.

Example-

```
#include<stdio.h>
#include<conio.h>
void main()
{
int marks;
printf("enter marks of student:");
scanf("%d",&marks);
if(marks>=250)
{
  if(attendance>=50)
  {
    printf("Pass");
    }
else
printf("your attendance os poor");
}
else
printf("Fail");
getch();
}
```

*** Switch statement-**

In real life we are often face with situation where we require to make a choice from a number of alternative rather than one or two. The control statement , which allows us to make a decision from the number of choices, is called 'switch' or more correctly a switch-case-default, since thse three keywords go together to make up the control statement.

Syntax-

```
Switch(Expression)
{
Case1: statement1;
        break;
Case2: statement2;
        break
```

```
:
:
Case N: statement N;
        break;
default: default_statement;
}
```

- The expression following switch keyword is any ‘c’ expression that yields an integer value.(a) It could be an integer constant like 1,2,3-----
  (b) A variables that evaluates to an integer or char.
- An integer or a character constant follows the keyword case in the condition
- Each constant in each case must be different from all the others.
- The group of statements in the above fromof switch represents any valid ‘c’ statements.
- The integer expression, following the keyword switch is evaluated
- The value it gives is then checked one against the constant value that follows the case statement.
- When a match is found the program executes the statement following that case condition.
- The break statement used in a switch takes the control outside the switch once the condition is satisfied.
- If no match is found with any of the case statements, then only the default statement will be executed.
- There are multiple statement to be executed in each “case” there is no need to enclose them within a pair of braces. Unlike if and else.
- The swatch statement is very useful while writing menu derive programs.

Example-

```
//program to display message using switch…case statement.
#include<stdio.h>
#include<conio.h>
main()
{
Int i=2;
switch(i)
{
Case 1: printf (“\n I am in case 1”);
        break;
Case 2: printf (“\n I am in case 2”);
        break;
Case 3: printf (“\n I am in case 3”);
        break;
default: printf(“I am in default”);
}
getch();
}
```

In above program value of ii 2, I is passed to the switch block, case condition will be checked one by one, case 1 is false so it will check next case 2 is true and it will print message as: 'I am in case 2'. The immediate break statement will terminate the switch block.

*** Loop Control Structure**

The computer ability is to perform a set of instructions repeatedly.
This involves repeating some portion of the program either a specified number of times or until a particular condition is being satisfied.
This repetitive operation is done using loop control structure.
There are three methods by way of which we can repeat a part of a program.

1) using for loop
2) using while loop
3) using d0-while loop

*** For Loop-**

The for loop allow us to specify three things in a single line. Initializing a loop counter, testing the loop counter to determine whether its value has reached the number of repetitions desired. Increasing the value of loop counter each time the program segment within the loop has been executed.

Syntax-

for(initialization;condition;increment/decrement);
{
Statements;
}

Initialization part will be executed only once i,e at the starting of loop. Then condition will be checked, if it is true then statements will be executed and then increment or decrement part be executed. Again condition will be checked and same procedure will be performed. All the three part must be separated by a ';'. For loop can be used when there is fixed number of iterations.

Operators -
i=i+1
i++ : it is a increment operation which increments the value by 1.
i+=1 : it is a compound assignment operator. It is increment the value by 1.

The above three statements are same, they will give the same result i.e will be incremented by 1. You can use any of them.

Example-

```
#include<stdio.h>
#include<conio.h>
main()
{
int i;
clrscr();
for(i=1;i<=5;i++)
{
printf("\n %d",i);
}
getch();
}
```

The above program will print 1 to 5 numbers. Variable i is initialized to 1, condition will be checked , it is true and value of i will be printed as 1. I will be incremented by 1 and again control will be transferred to condition. Same procedure will be performed till the condition is satisfying. Once condition is false loop will be terminated.
In for loop we can initialize more than one variables also.

*** While loop-**

The statement within the while loop would kept on getting executed till the condition being tested remains true.
When the condition become false, the control passes to the first statement that follows the body of the first statement that fallows the body of the while loop.
While loop is called entry verified loop.
The condition being tested may use relational or logical operators.
The statement within the loop may be a single line or a block of statements should be enclosed within pair of braces({}).

Syntax-

```
Initialize loop counter
While (condition)
{
Group of statements;
Increment loop counter;
}
```

Example-

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

```
int i;                  //initialization
clrscr();
while(i<=1)             //condition
{
printf("Deogiri ");
i++;                    //increment
}
getch();
}
```

*** Do…while loop-**

Do…while loop is same as the while loop.

Syntax-

```
do
{
statement;
loop counter variable;
}while(condition)
}
```

The difference is in do…while loop statements will be executed first & condition will be checked at the end of loop as it is given with the while statement i.e in do…while statement will be executed at least once even if the condition is false.

Example-

```
#include<stdio.h>
#include<conio.h>
main()
{
int n;                  //initialization
clrscr();
do
{
printf("/n %d",n);
n++;                    //increment
}
while(n<=10);           //condition
getch();
}
```

The above program will display 1 to 10 numbers. First number(n) will be printed and then condition will be checked. Every time n will be incremented by one. If initially the condition is false still number will be printed as 1.

*** Nested Loop-**

Loop within loop is called nested loop
It includes inner loop & outer loop.
Useful when displaying multidimensional data. (eg- matrix, array, etc)
Nested loop can be used to print patterns that require precise control over rows and columns. also used in data processing.

Syntax-

```
for(initialization;final condition;increment/decrement)
{
for(initialization;final condition;increment/decrement)
{
Statements;
}  //end of loop
}
```

Example-

// To print a numbers in following format

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include<stdio.h>
#include<conio.h>
main()
{
 int i,j;
for(i=1;i<=5;i++)
{
  for(j=1;j<=i:j++)
  {
     printf("%d \t",j);
  }
printf("\n");
}
getch();
}
```

Here inner loop will be executed for 5 times till outer if condition is satisfying. First single '1' will be printed and inner loop will be closed then cursor will be set to next line using "\n". i will be incremented and again inner loop will be executed, now it will print two numbers'1 2'. Outer loop value remains constant for complete inner loop.

*** Jump statements-(break, continue, goto, exit)**

1) Break statement-

The break statement in C is used in two different context.
In switch case, break is placed as the last statement of each case block.
When we used inside a loop, break causes the loop to be terminated.

Example-

```
#include<stdio.h>
#include<conio.h>
main()
{
int i;
for(i=1;i<=5;i++)
{
(i==3)
break;
printf("%d",i);
}
getch();
}
```

In above program output will be printed till only 3 because we used break statement there.

2) Continue statement-

Used to skip the execution of the rest of the statement within the loop in the current iteration & transfer it to the next loop iteration.

Example-

```
#include<stdio.h>
#include<conio.h>
main()
{
int i;
for(i=1;i<=5;i++)
{
(i==3)      //no. 3 is skipped
continue;
printf("%d",i);
}
```

Output will be-1
2
3
4

3) Goto statement-

Used to transfer program control to a predefined label

Goto statement is preferable when we need to break multiple loop using single statement at the same time.
Goto statement is very confusable so that ignore to use this.
We can repeat the program using goto.
In a difficult programming situation it seems so easy to use a goto where you want to.

Syntax-
goto labelname

Example 1-
```
#include<stdio.h>
#include<conio.h>
main()
{
Label:
printf("hello…world");
goto label;
getch();
}
```

Output will be printed as hello…world till infinite.

Example 2-
```
#include<stdio.h>
#include<conio.h>
main()
{
Top:
printf("hello…world");
char c;
printf("do you want to repeat this program? Yes/No ");
scanf("%d",&c);
if(c==yes)
{
goto top;
}
getch();
}
```

If we input yes then execution will be repeated, if we input no then control goes to program.

4) Exit statement-
Terminated entire program execution by using exit();.
The exit() is used to end a program.
It is standard library function of c, control by the operating system to exit the program.
It can be written any where in program body.

Syntax- *exit();*
Its like return 0;