UNIT –II

DECISION MAKING AND BRANCHING

INTRODUCTION:

C program is a set of statements which are normally executed sequentially in the order.

C language possesses such decision – making capabilities by supporting the following statements:

1. If statement
2. Switch statement
3. Conditional operator statement
4. Goto statement

DECISION MAKING WITH IF STATEMENT :

The if statement is a powerful decision – making statement and is used to control the flow of execution of statements.

There are four types of If statement:

- Simple If statement
- The If….Else Statement
- The Nested If……Else Statement
- The Else If Ladder Statement
-

SIMPLE IF STATEMENT:

In simple if statement it check the test expression, if it is true than the statement block will be executed. Otherwise the statement – block will be skipped and the execution will jump to the next statement.

Syntax:

```
if ( test expression )
{
statement – block ;
}
 next statement;
```

Example:

```
if(a>b)
{
Printf(“A is greater”);
}
Printf(“B is greater”);
```

Program:

```
#include<stdio.h.
void main()
{
   int a,b;
   clrscr ();
   scanf(“%d %d”,&a, &b);
   if(a>b)
   {
        printf(“ A is greater”);
   }
   printf(“The  Condition is failed”);
   getch();
```

```
        }
```
Output :
```
        6  4
        A is greater
        3  8
   The Condition is failed
```

The If……Else Statement :

The if……else statement is the extension of the simple if statement . In this it will check the test expression, if it is true than it will execute the true block statement will execute otherwise false block statement will executed.

Syntax:

```
if( test expression )
{
True – block statements;
}
else
{
False – block statements;
}
```

Example:

```
if (a >b)
{
printf("A is greater");
}
else
{
Printf("B is greater");
}
```

Program :

```
#include<stdio.h>
void main()
 {
      int a,b;
      clrscr();
      scanf("%d %d",&a,&b);
       if(a>b)
       {
            printf(" A is greater");
       }
       else
       {
             printf("B is greater");
       }
       getch();
 }
```

Output:
5 6
B is greater
6 4
A is greater

The Nested If……Else Statement :
When a series of decisions are involved, we may have to use more than one if…..else statement in nested form.
Syntax:

```
if ( test condition 1)
{
if( test condition 2)
{
statement 1;
}
else
{
statement 2;
}
}
else
{
statement 3;
}
```

Example:

```
if (a>b)
{
if(a>c)
{
printf(“ A is greater”);
}
else
{
printf(“C is greater”);
}
}
else
{
if(b>c)
{
printf(“B is greater”);
}
else
```

```
{
printf("C is greater");
}
}
```

Program :

```
#include<stdio.h>
void main()
 {
     int a , b , c ;
     clrscr ();
     printf("Enter the value of  a, b and c");
     scanf("%d %d %d", &a ,&b , &c);
     if(a>b)
     {
          if(a>c)
          {
               printf( " A is greater");
           }
           else
           {
                Printf( " C is greater");
           }
     }
     else
     {
           if(b>c)
           {
                 printf(" B is greater ");
           }
           else
           {
                 printf(" C is greater");
           }
      }
      getch ();
  }
```

Output :

3 5 7

c is greater

The Else If Ladder:

This is the another way of putting if's together when multipath decision s are involved.

Syntax:

```
if ( condition 1)
    statement 1;
else if ( condition 2)
```

```
   statement 2;
            else if ( condition 3)
                  statement n;
            else
                  default statement ;
```

Example:

```
if (mark>60)
{
printf(“First class”);
               }
               else if (mark>50)
{
printf(“Second class”);
}
else if ( mark>40)
{
printf(“Third class”);
}
else
{
printf(“Fail”);
}
```

Program :

```
            #include<stdio.h>
            void main()
             {
                    int mark;
                    char grade[30];
                    clrscr();
                    scanf(“ %d “, &mark);
                    if ( mark >60)
                    {
                           strcpy ( grade, “ First Class”);
                           printf(“Grade is %s “, grade);
                     }
                     else if ( mark > 50)
                     {
                              strcpy ( grade,”Second Class”);
                              printf(“ Grade is %s “,grade);
                     }
                     else if( mark >40)
                     {
                               strcpy ( grade, “Third class”);
                               printf(“ Grade is %s”, grade);
                     }
                     else
```

```
        {
                strcpy ( grade,"Fail");
                printf(" Grade is %s ",grade);
         }
          getch();
    }
```

Output :

```
50
Second Class
40
Fail
```

THE SWITCH STATEMENT:

The switch statement test the value of a given variable against a list of case values and when a match is found, a block of statements associated with that case is executed .

Syntax:

```
switch ( expression )
        {
Case value -1:
block -1;
 break;
case value -2:
block -2;
break;
.
.
default :
default – block;
 break;
}
```

Program :

```
#include<stdio.h>
void main()
{
      int  i , j;
      clrscr();
      scanf("%d",&i);
       j= i/10;
       switch(j)
       {
```

```
                case 1 :
                        printf(" The value of i is %d",j );
                        break;
                case 2 :
                        printf(" The value of i is %d",j);
                        break;
               case 3 :
                        printf("The value of i is %d",j);
                        break;
               default :
                        printf(" There is no value for j");
                        break;
             }
             getch();
}
```

Output:
30
The value of i is 3
40
There is no value of j

THE CONDITIONAL OPERATOR:
The conditional operator is a combination of ? and : and takes three operator is popularly known as the conditional operator.
Syntax:
Conditional expression? expression1: expression 2;
Example:
(i>8) ? 0 :1;

Program :

```
#include<stdio.h>
void main()
{
        int  i, k ;
        clrscr();
        scanf("%d",&i);
        k = (i<8) ? 0 : 1 ;
        printf(" The value of k is %d\n",k);
        getch();
 }
```

Output :

```
4
The value of k is 0
10
 The value of k is 1
```

THE GOTO STATEMENT:

C supports the goto statement to branch unconditionally from one point to another in the program.

In this two jump is there:

Backward jump

Forward jump

Example:

```
goto label;label:
  ……..
……..
label:goto label;
```

Forward jump Backward jump

Goto breaks the normal sequential execution of the program. If the label: is before the statement goto label; a loop will be formed and some statements will be executed repeatedly. Such a jump is known as a backward jump.

On the other hand , if the label : is placed after the goto label; some statements will be skipped and the jump is known as a forward jump.

Program : (Forward Jump)

```
#include<stdio.h>
void main()
{
   goto label;
   printf(" The loop is executed");
   label:
   printf("The loop is not executed");
   getch();
}
```

Output :

The loop is not executed

Program :

(Backward Jump)

```
#include<stdio.h>
void main()
{
      label:
      printf(" The loop is executed");
      goto loop;
      getch();
}
```

Output :

The loop is executed

DECISION MAKING AND LOOPING

INTRODUCTION:

In C if we want to initialize and increment a counter and test its value at an appropriate place in the program for the completion of the loop.

The C language provides for three constructs for performing loop operations. They are:

1. The while statement
2. The do statement
3. The for statement

THE WHILE STATEMENT :

The while is an entry – controlled loop statement. In while statement, first it check the condition if the condition satisfied than only the loop while be executed.

Syntax:

```
while (test condition )
{
Body of the loop;
}
```

Example:

```
while ( n <=10)
{
Printf("The value of i is %d",n) ;
n++;     }
```

Program :

```
#include<stdio.h>
void main()
{
    int n = 6;
    while(n <=10)
    {
        printf("The value of i is %d",n);
        n++;
    }
    getch();

 }
```

Output :

```
The value of i is 6
The value of i is 7
The value of i is 8
The value of i is 9
The value of i is 10
```

THE DO WHILE STATEMENT:

This do while statement first execute the loop than only it will check the condition. If the condition is false than also the loop will be executed atleast one time.

Syntax:

```
do
{
body  of the loop;
}
while ( test – condition);
```

Example:

```
do
{
printf("The value i is greater than 10");
}
while( i>10);
```

Program :

```
#include<stdio.h>
void main()
{
    int i;
    clrscr();
    scanf("%d",&i);
    do
    {
        printf("The loop is executed");
        printf("The value of i is %d\n");
     }
     while (i>5);
     getch();

}
```

Output :

```
4
The loop is executed
The value of i is 4
7
The loop is executed
The value of i is 7
The loop is executed
The value of i is 6
```

THE FOR STATEMENT:

The for loop is another entry – controlled loop that provides a more concise loop control structure.

Syntax:

```
for( initialization; test  condition; increment )
    {
```

Body of the loop;
}

Example:

```
for ( i=0;i<5 ;i++)
{
Printf("The value of i is  %d", i);
}
```

Program :

```
#include<stdio.h>
void main()
{
    int i;
    scanf("%d",&i);
    for(i=0;i<5;i++)
    {
        printf("The value of i is %d",i);
    }
    getch();
}
```

Output :

```
The value of i is 0
The value of i is 1
The value of i is 2
The value of i is 3
The value of i is 4
```

JUMPS IN LOOPS:

Loops perform a set of operations repeatedly until the control variable fails to satisfy the test – condition. The number of times a loop is repeated is decided inadvance and the test condition is written to achieve this. Sometimes, when executing a loop it becomes desirable to skip a part of the loop or to leave the loop as soon as certain condition occurs.

Jumping Out of a Loop:

We can use

i) Break and
ii) Continue statement

i)Break :

Break statement is encountered inside a loop , the loop is immediately exit and the program continues with the statement immediately following the loop. The break will exit only a single loop.

Syntax:

```
break;
```

Example:

```
for ( i=0 ; i>n ; i++)
```

```
{
printf (”The value of i is %d”,i)’
break ;
}
```

Program :

```
#include<stdio.h>
void main()
 {
        int i, n=5;
        for(i=0 ; i>n; i++)
        {
                Printf(“The value of i is %d”,i);
                Break;
        }
        getch();
 }
```

Output :

The value of i is 0