INTRODUCTION TO CONTROL STATEMENTS IN PROGRAMMING LANGUAGE--

C

PREPARED BY
SONIA MAHINDRU
ASSISTANT PROFESSOR IN COMPUTER SCIENCE
HANSRAJ MAHILA MAHA VIDYALAYA JALANDHAR

CONTROL STATEMENTS

It is a statement whose execution results in a choice being made as to which of two or more paths should be followed.

Types of statements

1. Branching: These are used when a condition arises in a statement.
2. Looping: When a single statement or a group of statements will be executed again and again.
3. Jumping: These are used to jump from one statement to another and make the execution of the programming procedure fast.

BRANCHING STATEMENTS

IF STATEMENT

This is used when only one condition occurs in a statement.

Syntax:

```
if(condition)
{
true statement block;
}
Statement X;
```

Program to check if a number input by user is even

```
main()
{
int n;
Printf(“Enter number:”);
Scanf(“%d”,&n);
If(n%2==0)
{
Printf(“Its even”);
}
}
```

BRANCHING STATEMENTS

IF ELSE STATEMENT

This statement has a single condition with two different blocks. One is true block and other is false block.

Syntax:

```
if(condition)
{
true statement block;
}
else
{
False statement block;
}
Statement X;
```

Program to check whether a number is positive or negative

```
main()
{
int n;
clrscr();
Printf(“Enter number:”);
Scanf(“%d”,&n);
If(n>0)
{
Printf(“Positive number”);
}
else
{
Printf(“Negative number”);
}
getch();
}
```

NESTED IF STATEMENT

This is used when an if statement occurs within another if statement

Syntax:

```
if(condition1)
{
        if(condition2)
        {
        true statement;
        }
        else
        {
        false statement;
        }
}
else
{
        if(condition3)
        {
        true statement;
        }
        else
        {
        false statement;
        }
}
Statement X;
```

Program to check whether a number is zero, positive or negative

```
main()
{
int n;
clrscr();
Printf(“Enter number:”);
Scanf(“%d”,&n);
If(n==0)
{
Printf(“number is zero”);
}
else
{
        if(n<0)
        {
        Printf(“Negative number”);
        }
        else
        {
         Printf(“Positive number”);
        }
}
getch();
}
```

LADDER IF OR ELSE IF STATEMENT

This is used when a series of many conditions have to be checked.

In this statement first condition will be checked, if it is true then action will be taken, otherwise further next condition will be checked and this process will continue till the end of the condition.

Syntax:

```
if(condition1)
{
        st1;
}
else if(condition2)
{
        st2;
}
_______
_______
_______
else if(condition-n)
{
        stn;
}
else
{
default statement;
}
statement-X;
```

SWITCH STATEMENT

When number of condition occurs in a problem and it is very difficult to solve such type of complex problem with the help of ladder if statement then there is need of such type of statement which should have different alternatives or different cases to solve the problem in simple and easy way.

Decision is made by switch statement based on its argument that can be only of int or char variable but not any floating point variable. It does not allow use of && and ||operators.

Switch statement is faster to execute than else-if ladder.

There is no strict rule about ordering the cases, various cases can be written in any order.

```
switch(e or v)
{
        case value1:
                block 1;
                break;
        case value2:
                block 2;
                break;
        ----------------
        ----------------
        case value n:
                block n;
                break;
        default:
                block n+1;
}
Statement X;
```

Program to print the color according to the code

```
main()
{
int code;
Printf(“\n MAIN MENU”);
Printf(“\n1. For color Red”);
Printf(“\n2. For color Green”);
Printf(“\n3. For color White”);
Printf(“\n1. For color Yellow”);
Printf(“\nEnter the color code”);
Scanf(“%d”,&code);
Switch(code)
{
case1:
        printf(“\nColor is Red”);
        break;
case2:
        printf(“\nColor is Green”);
        break;
case3:
        printf(“\nColor is White);
        break;
Case4:
        printf(“\nColor is Yellow”);
        break;
default:
         printf(“\nColor inot found”);
}
getch();
}
```

LOOPING STATEMENTS

When a single statement or a group of statements will be executed again and again in a program(in an iterative way),then such type processing is called loop. Loop is divided into two parts :

a) Body of the loop
b) Control of loop

Body mainly control of loop is further subdivided into two parts:

i. Entry Control Loop
ii. Exit Control Loop

Entry controlled and Exit controlled loops are purely opposite to each other.

WHILE STATEMENT

It is an entry control loop. In this first of all condition is checked and if it is true, then group of statements or body of loop is executed. It will execute again and again till condition becomes false.

Syntax:

```
while(test condition)
{
block of statements;
}
statementX;
```

DO STATEMENT

In this statement first body of the loop is executed and then the condition is checked. If condition is true then the body of the loop is executed. When condition becomes false then it will exit from the loop.

Syntax:

```
do
{
block of statements;
}
while(condition);
statement X;
```

Program to find average of n positive numbers

```
main()
{
int i,sum,n;
Float av;
clrscr();
Printf(“Enter number:”);
Scanf(“%d”,&n);
Sum=0;
i=1;
While(i<=n)
{
sum = sum + i;
i = i + 1;
}
av = (float)sum/n;
Printf(“\n Sum=%d”,sum);
Printf(“\n Average=%f”,av);
getch();
}
```

FOR STATEMENT

It is a looping statement which repeat again and again till it satisfies the defined condition. It is a one step loop, which initialize, check the condition and increment/decrement the step in the loop in a single statement.

Syntax:

```
for(initial value; test condition; increment/decrement)
{
Body of loop;
}
statement X;
```

NESTED FOR STATEMENT

When a for statement is executed within another for statement then it is called nesting

Syntax:

```
for(initial value1; test condition1; increment1/decrement1)
{
for(initial value2; test condition2; increment2/decrement2)
        {
        inner Body of loop;
        }
outer body of loop;
}
statement X;
```

Program showing nested loop

```
main()
{
int n=2, m=3, i ,j;
clrscr();
Printf(“Output is:”);
for(i=1;i<=n;i++)
{
        for(j=1;j<=m;j++)
        {
        printf(“\n Hello”);
        }
 printf(“\n OK”);
}
getch();
}
```

JUMPING STATEMENT

goto statement

This statement moves the control on a specified address called label or label name. The jump can be either in forward or backward direction.

Forward goto: The control moves forward at a specified label either according to a condition or without condition.

Syntax

```
S1;
S2;
goto label;
S3;
S4;
label:
        S5;
        S6;
```

JUMPING STATEMENT

Backward goto: It moves the control back to the specified addresses and so creates a loop. In case of conditional backward statement it creates finite looping. But in case of unconditional backward go to it creates infinite looping

Syntax

```
S1;
 label:
S2;
S3;
 goto label;
S4;
```

```
S1;
label:
S2;
S3;
if(condition)
goto label;
S4;
```

BREAK STATEMENT

This statement will quit from the loop when the condition is true. It is always used with decision making statement like if and switch statements. Break statement discontinues the execution of the loop.

Syntax:

```
break;
```

CONTINUE STATEMENT

This statement will skip some part of iteration and comes to the next looping step i.e it will increment/decrement the loop value when continue occurs. This statement is also used within any loop statement like do loop, while loop and for statement.

Continue statement continues the loop with next iteration.

Syntax:

```
continue;
```