23ITT101 – PROGRAMMING IN C & DATA STRUCTURES

UNIT – II - DECISION STATEMENTS & FUNCTIONS

TOPIC – DECISION MAKING AND BRANCHING STATEMENTS

A program contains a sequence of instructions. When each information is executed only once in the order if appears, the flow of execution is said to be from the top to bottom of the program but in practice if may be necessary to make the sequence of this flow to be transferred from one part of the program to another part, this can be achieved with the help of control flow constructions, control flow in structures are also known as control structures or control constructs.

The different control constructs (Branching Statements)

a) if statement

b) if…else statement

c) nested if...else statement

d) if.. else.. if statement

e) switch statement

f) Jumping statements

The IF Statement

The condition used in the if statement should be specified within parenthesis. When a program with if statements is evaluated the condition specified after the if statements is executed first. If the condition is true then the executable x-statement(s) within the flower braces will be executed.

SYNTAX:

```
if (condition)

{

executable-x-statement(s);

}

executable-statement (s);
```

If the condition is false, the control will be directly transferred to the statement outside the flower braces without executing executable x-statement. The executable x-statement(s), referred as the body of if statements should be enclosed within flower braces.

Two or more conditions may be combined in an if statement using a logical 'and' operator or a logical 'or' operator. The if statement can compare any number of variables in a single if statements. The various expression statements that can be used as conditions inside an if statements.

Example

```
#include <stdio.h>

void main()

{

int x;

printf("We enter any number:");

scanf("%d", &x);

printf ("The number entered here is %d", x);

if (x>100)

printf ("Hi. You have entered a comparatively higher number.");

}
```

Output

Case #1:

If x = 300,

The output generated here will be:

Hi. You have entered a comparatively higher number.

Case #2:

If x = 50,

The compiler will skip the statement, "Hi. You have entered a comparatively higher number. "

The if–else Statement

The if–else statement is an extension of an ordinary if statement. The if-else statement takes care of true as well as false condition. It has two blocks. One block is for true condition and the other block is for false condition.

Syntax:

The syntax available for the *if-else* statement will be:

```
if (condition x)

{

// The statements present inside the body of if

}

else {

// The statements present inside the body of else

}
```

Example:

```
#include <stdio.h>

void main()

{

int year;

clrscr();

printf(“We will enter a year here:”);

scanf(“%d”, &yr);

if ((year%4 == 0) && ((year%100 !=0) || (year%400==0)))

printf(“Yes. The provided year is a leap year. Great job…!!!!”);

else

printf (“No. The provided year is not a leap year. Try again….!!!”);

}
```

Output:

Case #1:

year = 1893

The output generated here will be:

We enter a year: 1893

No. The provided year is not a leap year….!!!

Case #2:

year = 1564

The output generated here will be:

We enter a year: 1564

Yes. The provided year is a leap year. Great job…!!!!

Nested if-else Statements

When **one IF function inside of another**, allows you to test multiple criteria and increases the number of possible outcomes, then it is called as Nested if-else statement.

Syntax

```
if(condition1) {

  /* Executes when the Condition1 is true */
  if(Condition2) {
    /* Executes when the Condition2 is true */
  }
}
```

Example

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

  /* local variable definition */
  int a = 100;
  int b = 200;

  /* check the boolean condition */
  if( a == 100 ) {
```

```
    /* if condition is true then check the following */
    if( b == 200 ) {
      /* if condition is true then print the following */
      printf("Value of a is 100 and b is 200\n" );
    }
  }

  printf("Exact value of a is : %d\n", a );
  printf("Exact value of b is : %d\n", b );

  return 0;
}
```

Output

Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200

If.. else if Statements

When a series of decisions are involved we have to use more than one if-else statement called as if-else..if Statement.

Syntax:

The syntax available for the nested if statement will be:

```
{

if (condition x)

{ Statement x; }

// The C Statement for this condition

else_if(condition y)

// The C Statement for this condition

{ Statement y; }

else

Statement z;

default:
```

// The C Statement for this default condition

;

}

Example

```
#include <stdio.h>

int main()

{

int a=40,b=20;

if (a>b) {

printf(“The number a is greater than the number b”);

}

else if(a<n) {

printf(“The number a is less than the number b”);

}

else {

printf(“The number a is equal to the number b”);

}

}
```

The output generated here would be:

The number a is greater than the number b

The Switch() Statement

As if statement, the switch control statement allows us to make a decision from a number of choices. It is usually called as a switch-case statement.

Syntax

switch(expression)

{
case value1:
//code to be executed;
break; //optional
case value2:
//code to be executed;
break; //optional
.....
default:
code to be executed if all cases are not matched;
}

Rules for writing switch () statement:

- The expression in switch statement must be an integer value or a character constant.
- No real numbers are used in expression.
- Each case block and default blocks must end with break statements.
- The default is optional.
- The case keyword must terminate with colon (:).
- default may be placed anywhere in the switch
- No two case constants are identical.

While using a Switch Statement

- Multiple statements can be executed in each case without the use of pair of braces as in the case of if or if-else statement. Character values are also allowed in cases. You can also mix integer and character constants in different cases of switch.
- A switch may occur within another switch, but it is rarely done. Such statements are called as nested switch statements.
- A default is optional. If it is not present and if none of the other cases match no action takes place. The default may appear anywhere in the switch statement, but there should be only one default in a switch statement.
- The switch statement is very useful while writing menu driven programs.
- Only one variable can be tested with the available case statements and with the values stored in them
- Floating-point, double, and long type variables cannot be used as cases in switch statement.

Example

```
#include<stdio.h>

int main(){

int number=0;

printf("enter a number:");

scanf("%d",&number);
```

```
switch(number){

case 10:

printf("number is equals to 10");

break;

case 50:

printf("number is equal to 50");

break;

case 100:

printf("number is equal to 100");

break;

default:

printf("number is not equal to 10, 50 or 100");

}

return 0;   }
```

Output

```
enter a number:4
number is not equal to 10, 50 or 100


enter a number:50
number is equal to 50
```

Jumping Statements

The Break Statement

- The break statement is used to terminate or to exit from a switch statement. The general form or the syntax of the break statement is

  break;

- The break statement does not have any embedded expressions or arguments. The break statement can be also used within for, while or do-while loops. The break statement is usually used at the end of each case and before the start of the next case statement. The break statement causes the control to transfer out of the entire switch statement. The keyword break breaks the control only from the loop in which it is placed.

The Continue Statement

- The continue statement is used to transfer the control to the beginning of the loop. The loop does not terminate when a continue statement is encountered. The continue statement can be used within a while or a do-while or a for loop. The general form or the syntax of the continue statement is

  continue;

- The continue statement does not have any embedded expressions or arguments. The continue statement applies only to loops and not for switch. A continue statement may appear only within an iteration statement. It causes the control to pass to the loop-continuation portion in which the continue statement is enclosed. In while and do-while loop a continue statement causes the control to go directly to the test condition and then to continue the iteration process. In case of for loop, the increment section of the loop executed first and then the test condition is evaluated.

The goto statement

The goto statement is used to transfer the control in a loop or a function from one point to any other portion in that program. If misused the goto statement can make a program impossible to understand.

syntax of goto statement is

goto label ; Statement (s);

. . .

label: statement(s) ;