#6 Decision Control Instructions

By Saurabh Shukla | 2017©mysirg.com

Control Instructions

Program is a set of instructions. We know that each instruction of the program is executed by the processor. It executes instructions one by one. Whatever we write in our program is executed in the same order as they appear in the program. We can say processor executes instructions in a sequential manner.

At an instance, processor is executing some line of code; we say control of processor is on that line. Processor's control moves from one line to another. This movement of control is known as flow of the program.

Sometimes, it is required that the program's flow shouldn't be sequential. For example, we want to execute first line of our program, then second and third line but after executing third line, we may want to skip fourth line and jump to the fifth line. In such situations, we use control instructions.

Control instruction gives power to the programmer to decide the flow of the program.

There are four types of control instructions:

- Decision control instruction
- Iterative control instruction
- Switch case control instruction
- goto instruction

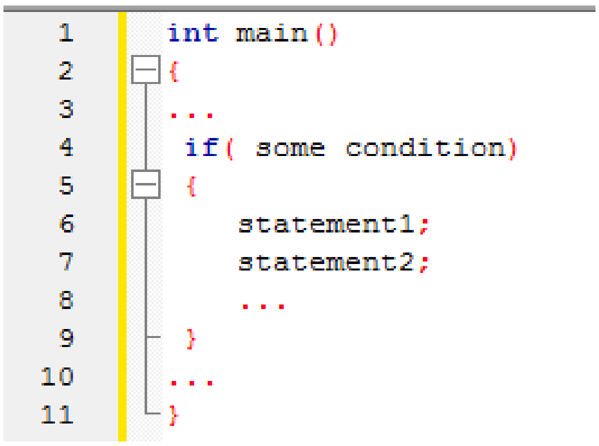
.

Decision Control Instruction

Decision control instruction is also known as selection control instruction. C language provides three ways to implement this instruction:

- if
- if-else
- Conditional Operator

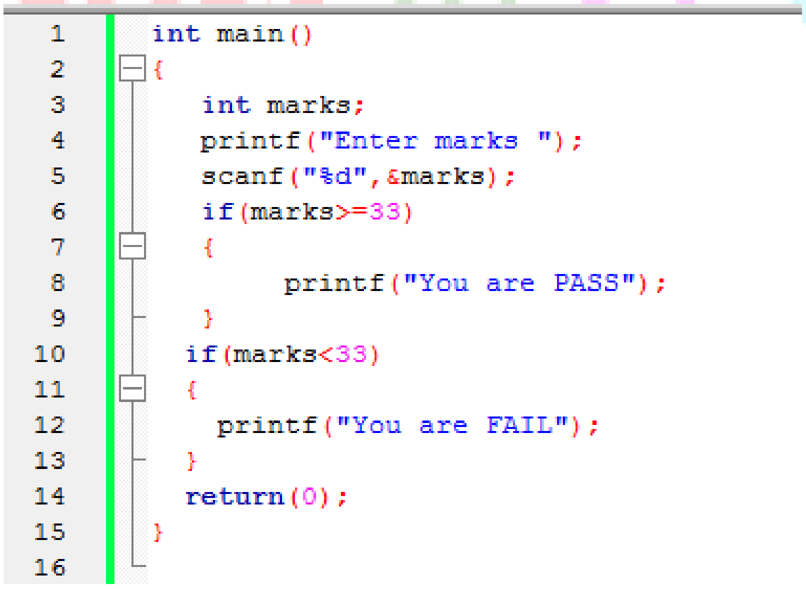
if statement



```c
int main()
{
...
 if( some condition)
 {
     statement1;
     statement2;
     ...
 }
...
}
```

- if is a keyword which let compiler know the decision control statement begins.
- Immediately after if keyword some condition is there in the parenthesis.
- Condition can be any valid expression in C language.
- Condition is always evaluated as true or false.
- If the result of expression is non-zero it is considered as TRUE otherwise FALSE.
- Immediately after this condition there is a block of code. This block is known as if block. Whatever we write in if block will be execute only when condition is TRUE.
- When condition is false control skip if block and execute statements written after if block.

Following is the program to check whether pass or fail on the basis of marks provided by user.



```c
int main()
{
   int marks;
   printf("Enter marks ");
   scanf("%d",&marks);
   if(marks>=33)
   {
        printf("You are PASS");
   }
  if(marks<33)
  {
    printf("You are FAIL");
  }
  return(0);
}
```

Output is

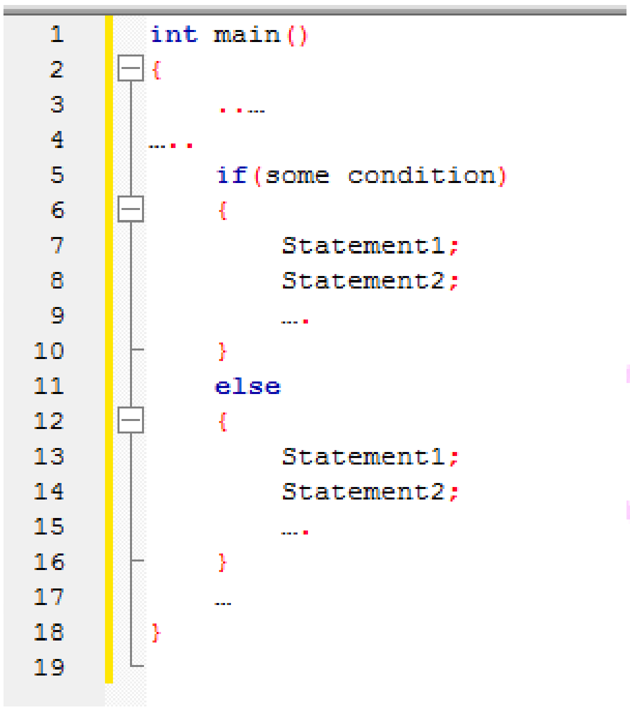
Enter marks 23

You are FAIL

Explanation:

- In this program output depends on the value given by user.
- Variable marks hold the value entered by user
- In the first if statement, the condition is *marks>=33*, thus if the marks is greater than or equal to 33 condition becomes TRUE and if block is executed, otherwise if block is skipped.
- Whatever may the result of first if condition, control has to reach second if statement.
- If marks is less than 33 condition is TRUE and execute if block, otherwise if block is skipped.

If else

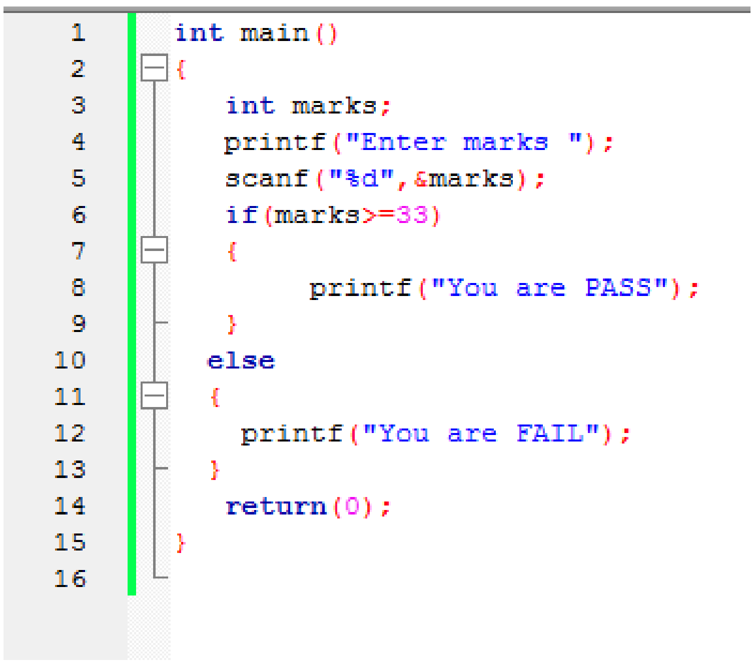


```
int main()
{
        ….
….
        if(some condition)
        {
                Statement1;
                Statement2;
                ….
        }
        else
        {
                Statement1;
                Statement2;
                ….
        }
        …
}

```

- This is similar to *if* but the *else* block is new add on.
- If the condition of *if* is TRUE *if* block is executed and if the condition of *if* is FALSE *else* block is executed.
- You can use *if* statement without *else* block but else must be in conjunction with *if* block.
- *else* block should appear immediately after *if* block otherwise an error occurred during compilation

Following is the program to check whether pass or fail on the basis of marks provided by user. This time we are going to use *if else*.



```
1   int main()
2   {
3       int marks;
4       printf("Enter marks ");
5       scanf("%d",&marks);
6       if(marks>=33)
7       {
8             printf("You are PASS");
9       }
10     else
11     {
12       printf("You are FAIL");
13     }
14      return(0);
15  }
16
```

Output is

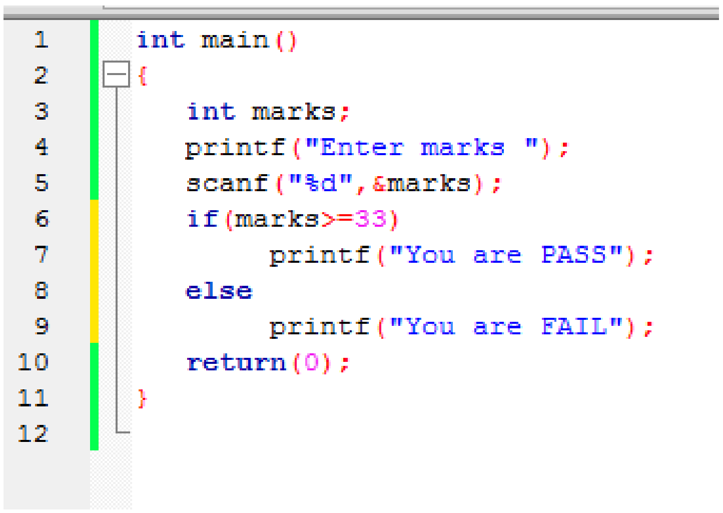
```
Enter marks 50

You are PASS
```

- In the above program, user entered a number, which we have taken in variable marks.
- At line number 6, if's condition is evaluated either as true or false, depending on the value stored in marks.
- If the value of marks is greater than or equal to 33, then the condition is true and “You are PASS” is printed on the screen.
- If the value of marks is lesser than 33, the condition becomes false, and control jumps to the else block, hence “You are FAIL” gets printed.

Remember:

- You can use if without else but cannot use else without if
- else block must be used just after the end of if block
- There is no separate condition for else block, rather it is executed when if's condition is evaluated false
- You can write any number of statements in if or else block
- When if or else block contains single statement, you can drop the curly braces (used to mention block). Following code is rewrite of previous program (not using curly braces for the body of if and else block):



```
int main()
{
    int marks;
    printf("Enter marks ");
    scanf("%d",&marks);
    if(marks>=33)
          printf("You are PASS");
    else
          printf("You are FAIL");
    return(0);
}
```

- Whenever there are more than one statement in if or else block, mentioning body using curly braces is mandatory.
- Do not put a semicolon at the end of if() or else, however this is not an error, but it is interpreted as if there is no statement in the block, also known as null statement.
- Do not write multiple conditions separated by comma. For example
  if(x>10, x<50)
  is not a valid way to write multiple conditions, rather you should use logical operators like
  if(x>10 && x<50)
- You can write another if or if else statement inside if or else block. This is called nesting.

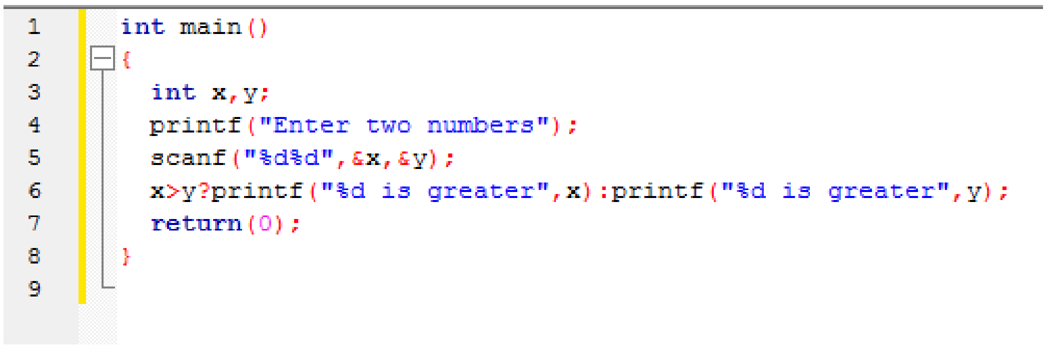
Conditional Operator

Conditional operator is the only operator in C language which requires three operands, hence known as ternary operator. Following is the syntax of conditional operator:

Expression1 ? expression2 : expression3;

Expression 1 is condition, evaluated as true or false. When expression 1 is true expression 2 is selected otherwise expression 3 is selected.

Following is the program to find greater between two numbers:



```
int main()
{
  int x,y;
  printf("Enter two numbers");
  scanf("%d%d",&x,&y);
  x>y?printf("%d is greater",x):printf("%d is greater",y);
  return(0);
}
```

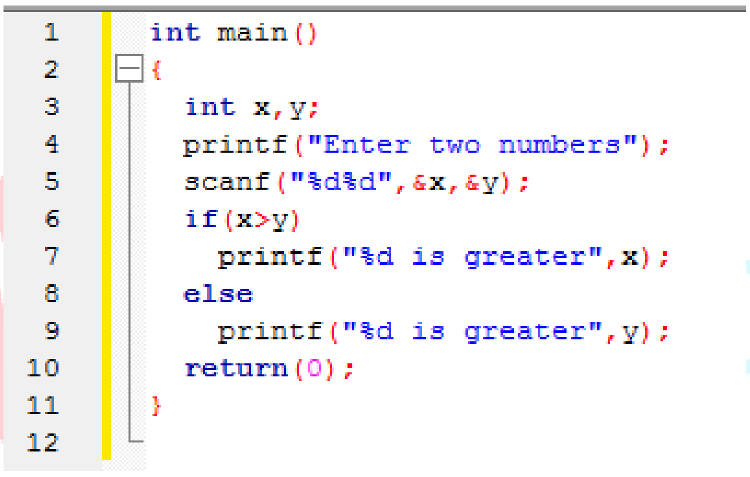
Output:

```
Enter two numbers 45

33

45 is greater
```

- Using conditional operator you can perform the same job which you could do with the help of if-else statement.
- You can easily transform if-else statement into conditional operator.
    - if's condition is expression 1 in conditional operator
    - if block's code is expression 2 in conditional operator
    - else block's code is expression 3 in conditional operator
- Above program can be assumed as a transformation of following program



```c
int main()
{
    int x,y;
    printf("Enter two numbers");
    scanf("%d%d",&x,&y);
    if(x>y)
      printf("%d is greater",x);
    else
      printf("%d is greater",y);
    return(0);
}
```

- Occasionally, you want to write more than one statement in expression 2 or expression 3, in that case use comma to separate statements. For example

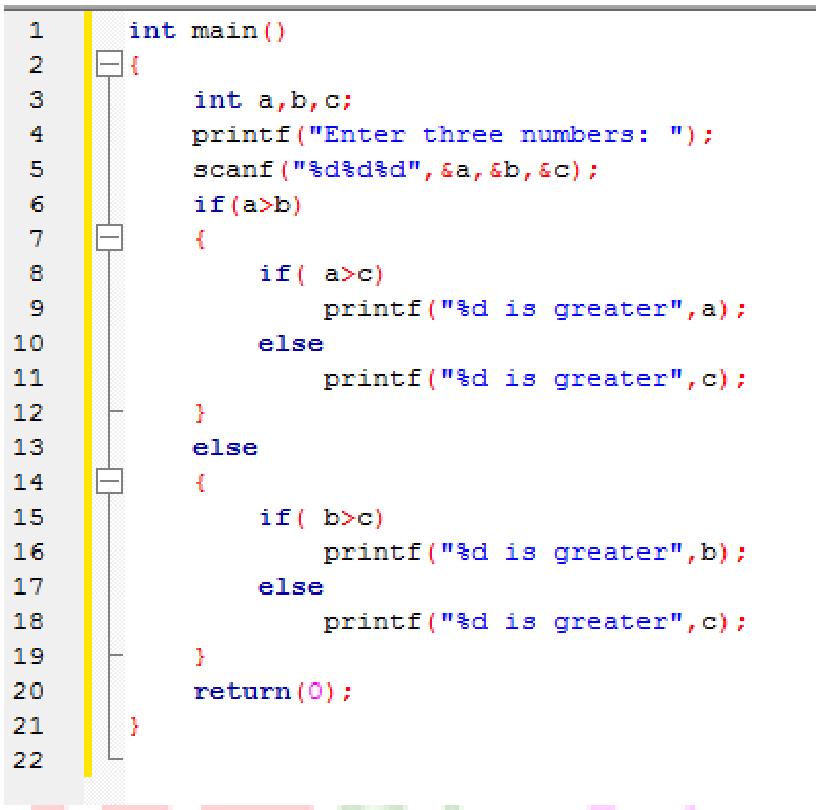
```c
x>y?printf("A"),printf("B"):printf("C"),printf("D");
```

- Conditional operator provides an ease of writing selective assignment. For example you want to assign max value between two data, to a variable:

```c
max=x>y ? x : y;
```

- You can use another conditional operator in a conditional operator as an expression2 or expression3. This is again called nesting.

Following is a program to find greater among three numbers. Notice the nested if else used in the program:

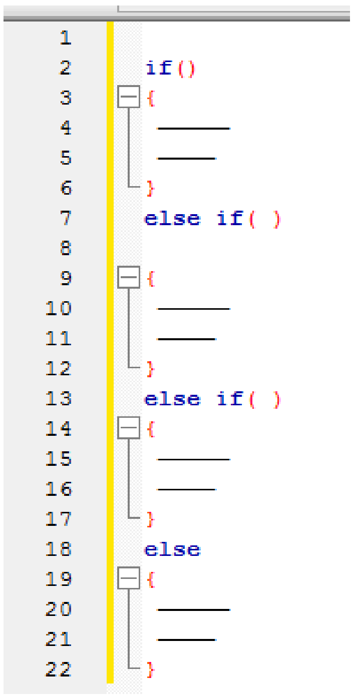


```
int main()
{
    int a,b,c;
    printf("Enter three numbers: ");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
    {
        if( a>c)
            printf("%d is greater",a);
        else
            printf("%d is greater",c);
    }
    else
    {
        if( b>c)
            printf("%d is greater",b);
        else
            printf("%d is greater",c);
    }
    return(0);
}
```

- Since, if-else statement is considered as a single action statement, thus you can remove curly braces used to mention the body of if block and else block.
- First condition a>b is evaluated, if it is true then control moves inside the if block and check for another condition a>c
- Similarly if the condition a>b is evaluated as false, then control jumps to the else block and check for another condition b>c

<u>else if ladder</u>

else if ladder is nothing but a special case of nesting where nesting only appears in else block. You can rearrange the structure as:



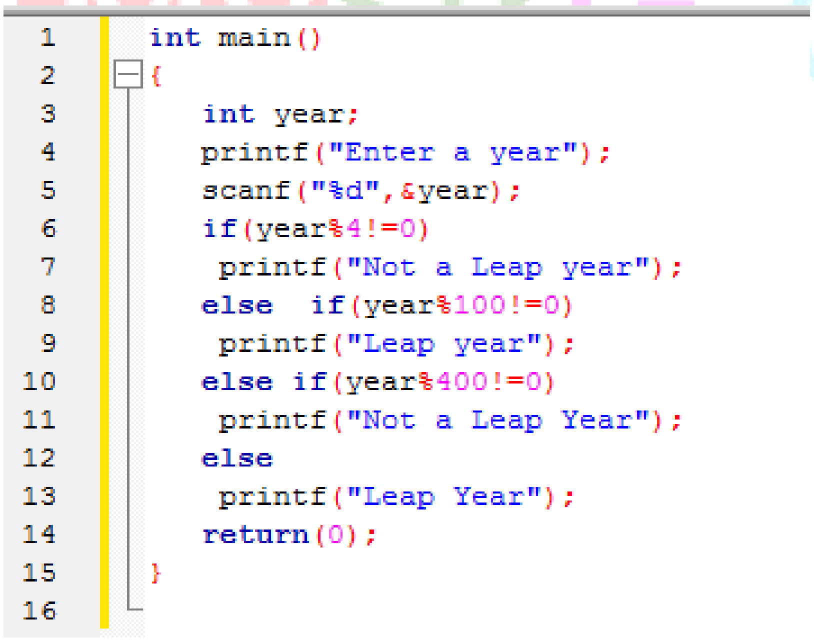
```

if()
{
 ______
 ____
}
else if( )

{
 ______
 ____
}
else if( )
{
 ______
 ____
}
else
{
 ______
 ____
}
```

Only one block is selected for execution.

Following is the program to check whether an year is leap year or not:



```
int main()
{
    int year;
    printf("Enter a year");
    scanf("%d",&year);
    if(year%4!=0)
     printf("Not a Leap year");
    else  if(year%100!=0)
     printf("Leap year");
    else if(year%400!=0)
     printf("Not a Leap Year");
    else
     printf("Leap Year");
    return(0);
}

```

References

YouTube video links

- Lecture 6 Decision Control in C part-1
  - https://www.youtube.com/watch?v=I2LRuehVLNA&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW
- Lecture 6 Decision Control in C part-2
  - https://www.youtube.com/watch?v=TNHkZKLXkyw&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW
- Lecture 6 Decision Control in C part-3
  - https://www.youtube.com/watch?v=HAwWchAM7UI&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW

Exercise

1. Write a program to check whether user inputted number is positive or non positive
2. Write a program to check whether user inputted number is divisible by 5 or not
3. Write a program to check whether user inputted number is even or odd
4. Write a program to check whether user inputted number positive, negative or zero
5. Write a program to find greater between two numbers. You have to print greater number, if both are equal then print any number.
6. Write a program to find greater between two numbers. Your result should be the number which is greater or if both are equal your output should be “equal”.
7. Write a program to print greater among three numbers. Print only the greater number.
8. Write a program to check whether a number is odd or even without using modulus (%) operator
9. Write a program to find roots of a given quadratic equation
10. Write a program to input marks of five subjects. You have to check whether the candidate is pass or fail. If the candidate is passed then print percentage marks and division.