Tutorial Link https://codequotient.com/tutorials/if statement/5a229ec8c66cfe38f2962228

TUTORIAL

if statement

Chapter

A statement is a part of program that can be executed. Each statement specifies an action to do. Statements in C can be divided in following categories: - Expression, Selection, Iteration, Jump, Label. Expression is any statement which can be composed of any valid constructs in C, for example, identifiers, operators etc.

Selection Statements

By default the instructions in a program are executed sequentially. However, in serious programming situations, seldom do we want the instructions to be executed sequentially. Many a times, we want a set of instructions to be executed in one situation, and an entirely different set of instructions to be executed in another situation. C supports two selection statements if and switch and one conditional operator i.e. '?:'. These are also called conditional statements. The simple example of this situation is: -

```
if Ram scored 40 or above marks in Paper1
then he must be declared Pass
otherwise he must be declared Fail
```

So in above example, Ram must be either declared Pass or Fail, but not both. So we have to write all the statements in a program, but as per the situations happen, some of them will execute while some or not. Also may be if we change the situations, the other set of statements will execute instead of first one. So, if Ram scored 55 then he is Pass, if Ram scored 34 he is Fail. In C, we can write this type of constructs using 'if' statements.

A decision can be taken in C with following three constructs: -

- The if statement
- The if-else statement
- The switch statement

The if statement

The simplest form of selection is if statement. The general form of if is as below: -

```
if (condition)
{
   execute this set of statements.
}
```

if keyword is a hint to compiler about condition. Now if the condition in parentheses evaluates to true, the set of statement following the if statement will be executed, otherwise not. The condition is an expression which evaluates to true or false. Remember, In C everything other than 0 is true, and 0 is false. So for example,

```c
#include <stdio.h>

int main()
{
  if(23)
  {
    printf("23 is evaluated as true in C language, hence this line will be printed.\n");
  }
  if(0)
  {
    printf("0 is evaluated as false in C, so this line will not be executed.\n");
  }
  return 0;
}

```

The first if statement will execute in above program as its condition evaluates to true. Whereas in second if the condition is false due to 0.

We will not pass the values to if statement like this, so here the relational and logical operators come into picture. We use the relational and logical operators generally to form the conditions of if statement. For example, following are some conditions using relational operators:

```c
#include <stdio.h>

int main()
{
```

```
5    int x = 23, y = 24;
6
7    if(x < y)
8    {
9      printf("X is less than y.\n");
10   }
11
12   if(x > y)
13   {
14     printf("X is greater than y.\n");
15   }
16
17   if(x == y)
18   {
19     printf("X is equal to y.\n");
20   }
21
22   if(x != y)
23   {
24     printf("X is not equal to y.\n");
25   }
26
27   return 0;
28 }
29
```

If the if block has only one statement to execute, we can omit the curly braces around it. For example, we can all if statements above as: -

```
if(x < y)
   printf("X is less than y.\n");
```

Although it is a good practice to put the braces, as it reduce the errors in future, if you need to modify the code for addition of lines.

Also, while using relational operators, be careful for = and == operators. Single (=) is an assignment operator, whereas double (==) is an relational operator to check for equality. Assignment operator will assign a value to some variable and return the assigned value, so if you accidentally use the = operator in if condition, there are no errors instead that is a valid program. For example,

```c
#include <stdio.h>

int main()
{
  int x = 23, y = 10;

  if(x = 45)
  {
    y = 50;
  }
  printf("x = %d  y = %d\n",x,y);    // 45 and 50 not 23 and 10

  return 0;
}

```

The above program will not check x with 45, instead it will assign 45 to x, and then return 45, which is treated as true in C, hence y will assigned 50. whereas if you replace the if statement with,

```
if(x == 45)
```

The program will print 23 and 10, instead of 45 and 50. So use the == operator with cautions. So all following are valid conditions in C

language:

```
if( 4 * 5)   // 20 is true
if( a = 10)   // 10 is returned, which is true
if(-6)      // -6 is also true in C
if(0)      // false because of 0
if( 4 ==5)   // false as 4 is not equal to 5
```

codequotient.com

Tutorial by codequotient.com | All rights reserved, CodeQuotient 2023