

Chapter 3 - Conditional Instructions

Sometimes we want to watch comedy videos on YouTube if the day is Sunday.

Sometimes we order junk food if it is our friend's birthday in the hostel.

You might want to buy an Umbrella if it's raining and you have the money.

You order the meal if dal or your favorite bhindi is listed on the menu.

All these are decisions which depends on a condition being met.

In C language too, we must be able to execute instructions on a condition(s) being met.

Decision Making Instructions in C

→ If - else Statement

→ Switch Statement

If - else Statement

The syntax of an If - else Statement in C looks like :

```
if (condition to be checked) {  
    Statements - if - condition - true;  
}  
else {  
    Statements - if - condition - false;  
}
```

Code example:

```
int a = 23;
```

```
if (a > 18) {  
    printf("You can drive \n");  
}
```

Note that else block is not necessary but optional.

Relational Operators in C

Relational operators are used to evaluate conditions (true or false) inside the if statements.

Some examples of relational operators are :-

$=$, $>$, $<$, \neq

$=$ equals
 $>$ greater than
 $<$ less than
 \neq not equal to

→ Important note ÷ '=' is used for assignment whereas '==' is used for equality check.

→ The condition can be any valid expression. In C a non-zero value is considered to be true.

Logical Operators

22, 11 and ! are three logical operators in C. These are read as "AND", "OR" and "NOT". They are used to provide logic to our C programs.

Usage of Logical Operators:

(i) $\& \& \rightarrow$ AND \rightarrow is true when both the conditions are true

"1 and 0" is evaluated as false.

"0 and 0" is evaluated as false.

"1 and 1" is evaluated as true.

(ii) $\parallel \rightarrow$ OR \rightarrow is true when at least one of the conditions is true. $(1 \text{ or } 0 \rightarrow 1)(1 \text{ or } 1 \rightarrow 1)$

(iii) $! \rightarrow$ returns true if given false and false if given true

$!(3 == 3) \rightarrow$ evaluates to false

$!(3 > 30) \rightarrow$ evaluates to true.

As the number of conditions increases, the level of indentation increases. This reduces readability. Logical operators come to rescue in such cases.

else if clause

Instead of using multiple if statements, we can also use else if along with if thus forming an if-else-if-else ladder.

```
if {  
    // statements;  
}
```

```
else if {  
    ...  
}
```

```
else {  
    ...  
}
```

Using if - else if - else reduces indents
The last "else" is optional
Also there can be any number of "else if"

Last else is executed only if all conditions fail.

Operator precedence

Priority	Operator
1 st	!
2 nd	*, /, %
3 rd	+, -
4 th	<, >, <=, >=
5 th	==, !=
6 th	&&
7 th	
8 th	=

Conditional Operators

A short hand "if - else" can be written using the conditional or ternary operators

Condition ? expression-if-true : expression-if-false
↓
Ternary operators

Switch Case Control Instruction

Switch-Case is used when we have to make a choice between number of alternatives for a given variable.

```
Switch (integer-expression)
{
```

```
    Case C1:
```

```
        Code;
```

```
    Case C2:
```

```
        Code;
```

```
    Case C3:
```

```
        Code;
```

```
    default:
```

```
        Code;
```

```
}
```

$C_1, C_2 \text{ \& } C_3 \rightarrow \text{Constants}$

$\text{Code} \rightarrow \text{Any valid C Code.}$

The value of integer-expression is matched against C_1, C_2, C_3, \dots . If it matches any of these cases, that case along with all subsequent "case" and "default" statements are executed.

* Quick Quiz: Write a program to find grade of a student given his marks based on below:

$\rightarrow 90 - 100 \rightarrow A$ $\rightarrow < 70 \rightarrow F$.

$\rightarrow 80 - 90 \rightarrow B$

$\rightarrow 70 - 80 \rightarrow C$

$\rightarrow 60 - 70 \rightarrow D$

Important Notes

- 1> We can use switch-case statements even by writing cases in any order of our choice (not necessarily ascending)
- 2> char values are allowed as they can be easily evaluated to an integer
- 3> A switch can occur within another but in practice this is rarely done.