C PROGRAMMING HANDBOOK BY SAICHANDAN GORLI

CHAPTER 3 : CONDITIONAL INSTRUCTIONS

Sometims we order junk kfood if mom is not at home.

In C language , we use be able to excute intructions on a condition

QUICK LINKS

DECISION MAKING INSTRUCITONS IN C

- if-else statememnt
- switch statement

IF-ELSE STATEMENT

Theh syntax of an if-else statement in C looks like:

```
if(condition){
    // Statement if condition is true
}
else{
// Statement if condition is false
}
```

CODE EXMAPLE:

```
int age = 18;
if(age > 18){
    printf("your age is %d",age);
}
```

RELATIONAL OPERATORS IN C

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

Some examples of relational operators are:

- < (less than)
- > (greater than)
- <= (less than equal to)
- >= (greater than equal to)
- == (Equality operator)
- != (not equal to)

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

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

LOGICAL INSTRUCTIONS IN C

- && (Logical AND)
- || (Logical OR)
- ! (Logical NOT)

Three logical operators in C. These are read as 'AND','OR' and 'NOT'.

They are used to provide logic to our C program.

USE OF LOGICAL OPERATORS:

1. && (AND) → is true whne both the conditions are true.
    a. "1 and 0" is evaluated as false.
    b. "1 and 1" is evaluated as true.
    c. "0 and 0" is evaluated as false.
2. || (OR) → is true when at least one of the condition is true
    a. "1 and 0" is evaluated as true.
    b. "1 and 1" is evaluated as true.
    c. "0 and 0" is evaluated as false.

3.! (NOT) → returns true if given false and false if gives true

```
a. !(1 == 1) is evaluated as false.

b. !(5 > 50) is evaluated as true.
```

As the number og condition sincreases, the level of the indentation increases. This reduces he readibitly. LOgical operators come to rescue in such cases.

ELSE-IF LADDER

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

CODE EXAMPLE

A typical if-else if-else ladder looks like:

```
if (/* condition */)
{
    /* code */
}
else if (/* condition */)
{
    /* code */
}
else
{
    /* code */
}
```

IMPORTANT NOTE

1. Using the is-else if-else reduces indents.
2. The last "else" is optional.
3. Also there can be any number of "else-if".
4. Last else is executed only if conditions fail.

OPERATOR PRECEDENCE

The following table lists the operator priority in C

Priority	Operators
1st	!
2nd	* / %
3rd	+ -
4th	<>,<=,>=
5th	==,!=
6th	&&
7th	11
8th	=

TERNARY OPERATORS

A shorth and "if-else" can be written using conditional or ternary operator.

```
condition ? expression-if-true : expression-if-false

// #include <stdio.h>
// int main()
// {
//     int min, a = 5, b = 6;
//     min = (a < b) ? a : b;
//     printf("minimum : %d", min);
//     return 0;
// }
```

SWITCH CASE CONTROL INSTRUCTION

switch-case is used when we have to make a choice between number of alternatives for given variables

```
switch (expression) {
    case constant1:
        // Code to execute if expression equals constant1
        break;

    case constant2:
        // Code to execute if expression equals constant2
        break;

    // You can have any number of case statements

    default:
        // Code to execute if expression doesn't match any case
}
```

The value of integer-expression is matched against constant1,constant2... If it matches any of these cases ,that case along with all subsequent "case" and "default" statements are execute.

Key Points:

- **Expression**: The value that you are evaluating (usually an integer or character).

- **case**: Each case must have a constant expression (like an integer or character literal).
- **break**: This statement exits the switch block. If omitted, execution will continue into the next case (fall-through).
- **default**: This case runs if none of the specified cases match the expression.

Example :

```
#include <stdio.h>

int main() {
    int day;
    printf("enter the day no to know the day : ");
    scanf("%d",&day);

    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:

            printf("Wednesday\n");
            break;
        case 4:
            printf("Thursday\n");
            break;
        case 5:
            printf("Friday\n");
            break;
        default:
            printf("Weekend\n");
            break;
    }

    return 0;
}
```

_Some important note : _

- We can use Switch-case statements even by writing cases in any order out od our choice.
- char values are allowed as they can be easily evaluated to an integer.

- A switch can occur within another but in practice this is rarely done.