# Switch Statement in C

Switch case statement evaluates a given expression and based on the evaluated value(matching a certain condition), it executes the statements associated with it. Basically, it is used to perform different actions based on different conditions(cases).

* Switch case statements follow a selection-control mechanism and allow a value to change control of execution.
* They are a substitute for long [if statements](https://www.geeksforgeeks.org/decision-making-c-c-else-nested-else/) that compare a variable to several integral values.
* The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

In C, the switch case statement is used for executing one condition from multiple conditions. It is similar to an if-else-if ladder.

The switch statement consists of conditional-based cases and a default case.

## Syntax of switch Statement in C

**switch(expression)**

**{**

**case** value1**:** statement\_1;

**break;**

**case** value2**:** statement\_2;

**break;**

.

.

.

**case** value\_n**:** statement\_n;

**break;**

**default:** default\_statement;

}

## How to use switch case Statement in C?

Before using the switch case in our program, we need to know about some rules of the switch statement.

### Rules of the switch case statement

Following are some of the rules that we need to follow while using the switch statement:

1. In a switch statement, the “**case value**” must be of “**char**” and “**int**” type.
2. There can be one or N number of cases.
3. The values in the case must be **unique**.
4. Each statement of the case can have a break statement. It is optional.
5. The default Statement is also optional.

### Example

* C

|  |
| --- |
| // C program to Demonstrate returning of day based numeric  // value  #include <stdio.h>    int main()  {    // switch variable      int var = 1;      // switch statement      switch (var) {          case 1:              printf("Case 1 is Matched.");              break;            case 2:              printf("Case 2 is Matched.");              break;            case 3:              printf("Case 3 is Matched.");              break;            default:              printf("Default case is Matched.");              break;      }        return 0;  } |

**Output**

Case 1 is Matched.

**Difference Between Switch Case and Else If Ladder**

While programming, a number conditions come and a number of decisions need to be taken by a programmer. For decision making in [C programming](https://www.codewithc.com/c-programming-a-modern-approach-pdf-kn-king/), the statements such as if, if..else, else if, switch case etc. have defined in the standard C [library](https://www.codewithc.com/mini-project-in-c-library-management-system/). In case of multi-conditional processing, basically there come two statements in the choice of a programmer; they are switch case and else if ladder. In this post, I have presented the features and applications of the two statements, and the difference between switch case and else if ladder.

Though the function of switch case and else if ladder is same, there are a number of remarkable difference between switch case and else if ladder in many aspects such as memory consumption, [speed](https://www.codewithc.com/32-things-tools-that-will-make-software-development-speed-50-faster/) of processing, variable requirement, comfort in programming etc. Appropriate choice between switch case and if else ladder is essential for the sake of ease, comfort, accuracy and efficient programming.

## Else If Ladder:

else if statement can be defined as a control statement which controls the statement(s) to be executed on the basis of some conditions. Whenever the else if statement is used, the [compiler](https://www.codewithc.com/difference-between-compiler-interpreter/) or interpreter initially checks the condition whether it is true or false and if the condition is found to be true then, the corresponding statements are executed. If the condition is found to be false, it continues checking the next else if statement until the condition comes to be true or the control comes to the end of the else if ladder.

The **syntax of else if ladder** can be represented as:

if( condition-1)

statement-1;

else if (condition-2)

statement-2;

else if (condition-3)

statement-3;

else if (condition-4)

statement-4;

else if (condition-n)

statement-n;

else

default statement;

**goto Statement in C**

The **C goto statement** is a jump statement which is sometimes also referred to as an **unconditional jump** statement. The goto statement can be used to jump from anywhere to anywhere within a function.

**Syntax**:

Syntax1 | Syntax2

----------------------------

goto label; | label:

. | .

. | .

. | .

label: | goto label;

In the above syntax, the first line tells the compiler to go to or jump to the statement marked as a label. Here, the label is a user-defined identifier that indicates the target statement. The statement immediately followed after ‘label:’ is the destination statement. The ‘label:’ can also appear before the ‘goto label;’ statement in the above syntax.

**Examples:**

**Type 1**: In this case, we will see a situation similar to as shown in Syntax1 above. Suppose we need to write a program where we need to check if a number is even or not and print accordingly using the goto statement. The below program explains how to do this:

* C

|  |
| --- |
| // C program to check if a number is  // even or not using goto statement  #include <stdio.h>    // function to check even or not  void checkEvenOrNot(int num)  {      if (num % 2 == 0)          // jump to even          goto even;      else          // jump to odd          goto odd;    even:      printf("%d is even", num);      // return if even      return;  odd:      printf("%d is odd", num);  }    int main()  {      int num = 26;      checkEvenOrNot(num);      return 0;  } |

**Output:**

26 is even