STATEMENTS

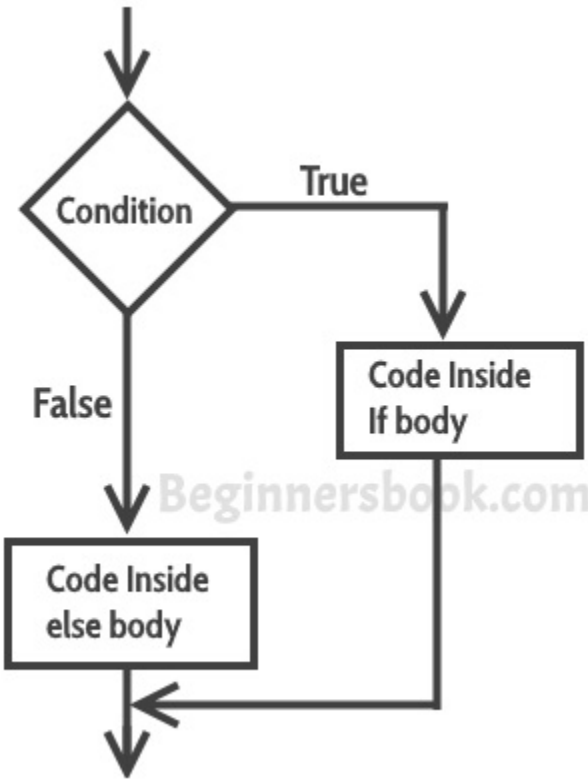
Syntax of if else statement:

If condition returns true then the statements inside the body of “if” are executed and the statements inside body of “else” are skipped.

If condition returns false then the statements inside the body of “if” are skipped and the statements in “else” are executed.

```
if(condition) {
  // Statements inside body of if
}
else {
  //Statements inside body of else
}
```

Flow diagram of if else statement



C Nested If..else statement

When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.

Syntax of Nested if else statement:

```
if(condition) {
   //Nested if else inside the body of "if"
   if(condition2) {
```

```
    //Statements inside the body of nested "if"
  }
  else {
    //Statements inside the body of nested "else"
  }
}
else {
  //Statements inside the body of "else"
}
```

C – else..if statement

The else..if statement is useful when you need to check multiple conditions within the program, nesting of if-else blocks can be avoided using else..if statement.

Syntax of else..if statement:

```
if (condition1)
{
  //These statements would execute if the condition1 is true
}
else if(condition2)
{
  //These statements would execute if the condition2 is true
}
else if (condition3)
{
  //These statements would execute if the condition3 is true
}
.
.
else
{
  //These statements would execute if all the conditions return false.
}
```

Nested If and Switch Statement with Example

It is possible to have a switch as a part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

Syntax

The syntax for a nested switch statement is as follows −

```
switch(ch1) {

case 'A':
printf("This A is part of outer switch" );

switch(ch2) {
case 'A':
printf("This A is part of inner switch" );
break;
case 'B': /* case code */
}

break;
case 'B': /* case code */
}
```

Example

Live Demo

```
#include <stdio.h>

int main () {

  /* local variable definition */
  int a = 100;
  int b = 200;

  switch(a) {

    case 100:
      printf("This is part of outer switch\n", a );

      switch(b) {
        case 200:
          printf("This is part of inner switch\n", a );
      }
  }

  printf("Exact value of a is : %d\n", a );
  printf("Exact value of b is : %d\n", b );
```

```
  return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
This is part of outer switch
This is part of inner switch
Exact value of a is : 100
Exact value of b is : 200
```

WRP to Find Given Number is Even or Odd

An even number is an integer that is exactly divisible by 2. For example: 0, 8, -24

An odd number is an integer that is not exactly divisible by 2. For example: 1, 7, -11, 15

Program to Check Even or Odd

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   // true if num is perfectly divisible by 2
   if(num % 2 == 0)
      printf("%d is even.", num);
   else
      printf("%d is odd.", num);

   return 0;
}
```

Run Code

Output

```
Enter an integer: -7
-7 is odd.
```

- In the program, the integer entered by the user is stored in the variable num.

Then, whether num is perfectly divisible by 2 or not is checked using the modulus % operator.

If the number is perfectly divisible by 2, test expression number%2 == 0 evaluates to 1 (true). This means the number is even.

However, if the test expression evaluates to 0 (false), the number is odd.

Program to Check Odd or Even Using the Ternary Operator

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   (num % 2 == 0) ? printf("%d is even.", num) : printf("%d is odd.", num);
   return 0;
}
```

Run Code

Output

```
Enter an integer: 33
33 is odd.
```

Program to Find Number of Days in a Month by Using Switch and Case Statement

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 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.

Syntax:

```
switch (n)
{
   case 1: // code to be executed if n = 1;
      break;
   case 2: // code to be executed if n = 2;
      break;
   default: // code to be executed if n doesn't match any cases
}
```

Some important keywords:

1) Break: This keyword is used to stop the execution inside a switch block. It helps to terminate the switch block and break out of it.
2) Default: This keyword is used to specify the set of statements to execute if there is no case match.
Important Points About Switch Case Statements:

1) The expression provided in the switch should result in a **constant value** otherwise it would not be valid. Some valid expressions for switch case will be,

```
// Constant expressions allowed

switch(1+2+23)

switch(1*2+3%4)


// Variable expression are allowed provided

// they are assigned with fixed values

switch(a*b+c*d)

switch(a+b+c)
```

2) Duplicate case values are not allowed.
3) The **default statement is optional**. Even if the switch case statement do not have a default statement,
it would run without any problem.
4) The **break statement is used inside the switch to terminate a statement** sequence. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
5) The **break statement is optional**. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.
6) Nesting of switch statements is allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes the program more complex and less readable.
7) Switch statements are **limited to integer values** only in the check condition.