“Natural selection will not remove ignorance from future generations.”

- Richard D.

Chapter 4

Selection Statements

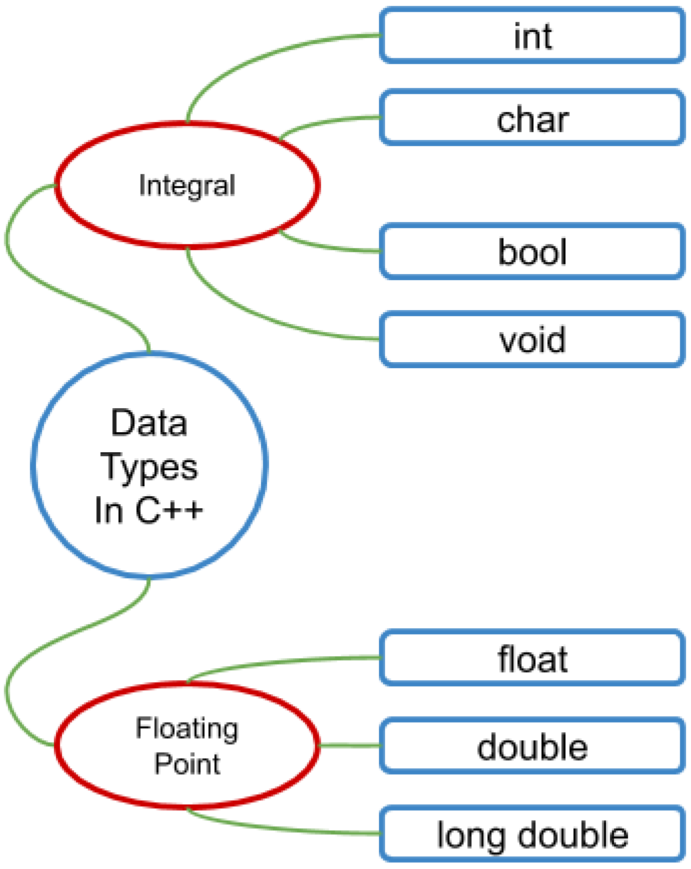
The Game of Rock-Paper-Scissor and Stopwatch

Summary

In this chapter we will discover how to use selection statements. We will learn different data types and operators used in C++. The main goal of the chapter is to integrate all the learnt tools and make two very important and fundamental small projects, one is **Rock-paper-Scissor** and the second is **Stopwatch**.

1. Data Types of the Variables

Data type, as cleared by its name, represents the name of the type of data. Before making any variable we have to specify the type of the memory they will be having, so that the compiler knows which type of data is going to store in that specific space. There are the following data types in C++.



1. Integral

Type	Size	Description
int	4 Bytes	It is used for storing whole numbers only like 1, 2, 23
bool	1 Byte	It is used for storing boolean or logical values, it only store true or false
char	1 Byte	It is used for storing a single character
void	1 Byte	Void mean value less it does not store any value

2. Floating point

Type	Size	Description
float	4 Bytes	It is used for storing single precision floating value
double	8 Bytes	It is used for storing double precision floating value

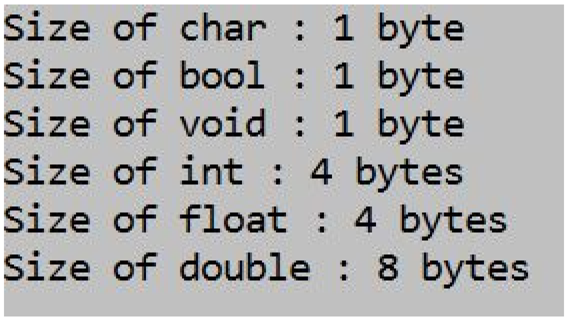
Remember
The sizes of these data types may be different from the table shown above. It depends on computer and compiler you are using.

Here is an example from which the size of these data types is checked.

Code

```
//program to check sizes of data types
#include<iostream>
using namespace std;

int main()
{
    cout << "Size of char : " << sizeof(char) << " byte" << endl;
    cout << "Size of bool : " << sizeof(bool) << " byte" << endl;
    cout << "Size of void : " << sizeof(void) << " byte" << endl;      // Error  As this doesn't have any specific size
    cout << "Size of int : " << sizeof(int) << " bytes" << endl;
    cout << "Size of float : " << sizeof(float) << " bytes" <<endl;
    cout << "Size of double : " << sizeof(double) << " bytes" << endl;

    return 0;
}
```

Output



```
Size of char : 1 byte
Size of bool : 1 byte
Size of void : 1 byte
Size of int : 4 bytes
Size of float : 4 bytes
Size of double : 8 bytes
```

Description

The above code is used to determine the size of various data types. As you see in code **sizeof** is used. It is a keyword in C++ to check size. The syntax to write **sizeof** is shown below

sizeof(data type);

1. Operators

In C++ operators are such symbolic representations of conceptual ideas which we can use to perform various mathematical operations like addition, subtraction, and so on (we will see its details in a while) and also logical operations like **and**, **or** and **not** (we will learn their functionality in a while).

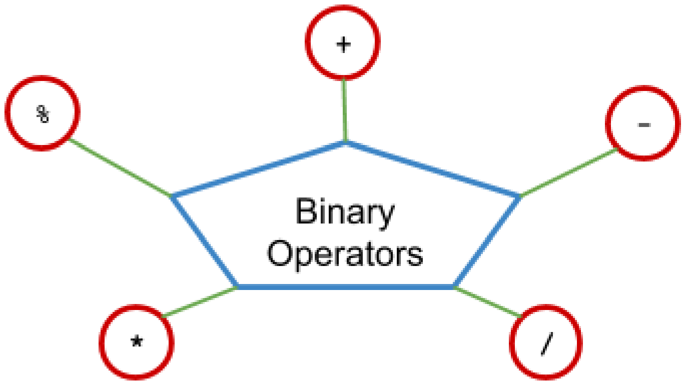
Basically There are 4 types of operators.

a. Arithmetic operators
    i. Unary operators
    ii. Binary operators
b. Logical operators
c. Comparison operators
d. Assignment operators

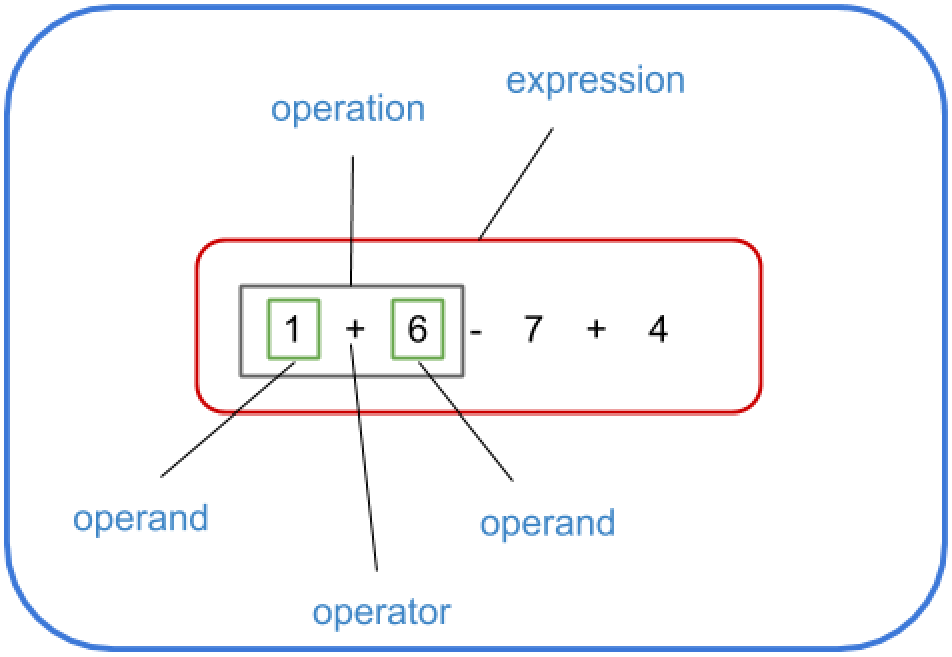
Let's start learning their functions one by one.

Arithmetic operators

a. Binary operator



Such symbols which are used to perform operations on numbers. Numbers can be of different types such as integers, floats or others. Actually the cause of calling it a binary operator is because this operator is used between two **operands**. In an expression one or more operations are performed and in a **operation** there are two operands and a operator like described in picture below



Hu

Following are symbols of arithmetic operators and their operation. Assume variable **A** holds **10** and **B** holds **20**.

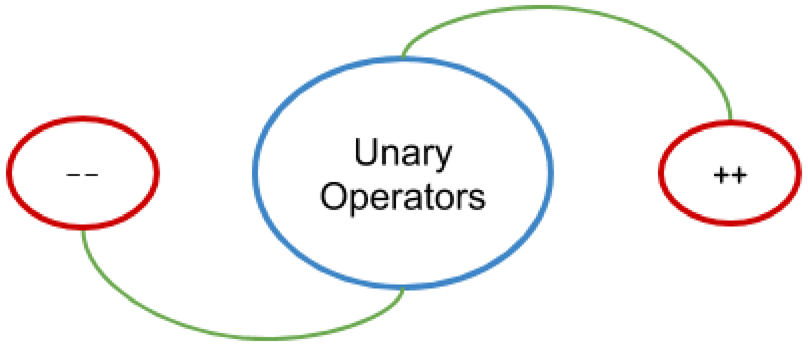
Symbol	**Operation**	**example**
+	addition	**A + B** replaced by **30**

-	subtraction	**B - A** replaced by **10**
*	multiplication	**A * B** replaced by **200**
/	division	**B / A** replaced by **2**
%	modulus(remainder)	**B % A** replaced by **0**

Some of these operators you have seen already seen in previous chapter addition(**+**), subtraction(**-**), multiplication(*) the rest are used in this chapter. In place of variables in above table there could be numbers or even further mathematical expressions or numbers.

Remember
Integer and floats are two different data types. An integer(int) is number without decimal point. A floating point(float) is number which has a decimal point. Modulus only works on integers because in mathematics remainder is only applicable on integer division.

b. Unary operators



Such symbols(operators) which uses only one operand. Like binary operators, unary operators also operate on integers and other data types except **`bool`**. Unary operator are of two types postfix (increment and decrement) and prefix (increment and decrement).
Like shown in the table below.

Assume variable x holds 3 and y holds 0

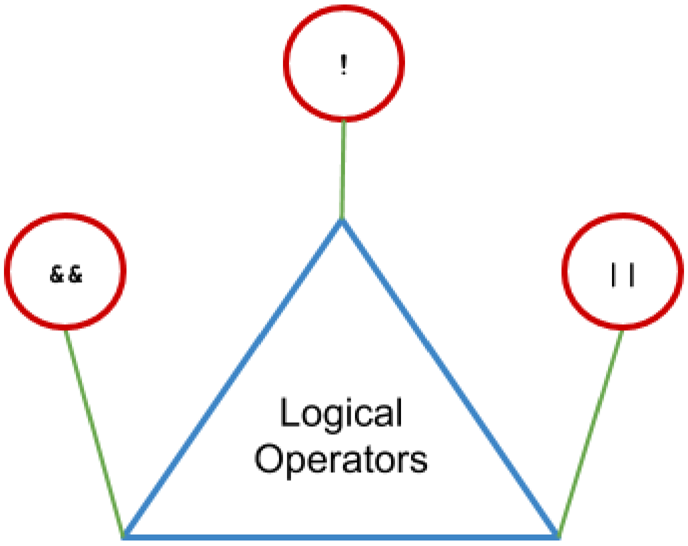
Symbol	**Meaning**	**Example**	**Example**
++	Increments (Increase value of the variable by one on which the operator is applied).	**y = ++x;** **// is equivalent to x = x+1 , y = x.** This is called pre-increment, which means first increment will happen and then that expression will be replaced by the new value. e.g. After this instruction's, **x** will contain	**y = x++;** **// is equivalent to x = y , x = x+1** This is called post-increment, which means first the value of **x** will be replaced and then after that **x** will be incremented. e.g. After this instruction execution, **x** will

		4, **y** will also contain **4**	contain **4**, **y** will contains **3**
--	Decrement (Decrease value of variable by one)	**y = --x; // x = x-1 , y = x** This is called pre-decrement, After the execution of this instruction **x** will contain **2**, **y** contain **2**	**y = x--; // y = x , x = x-1** This is called post-decrement After the execution of this instruction's **x** will contain **2**, **y** will contain **3**

Logical operators

Such operators which combines conditional statement, actually this operator is used to extend a logical condition. You can combine many conditions by logical operators to make a complex condition.
There are three types of logical operators:

1. And operation represented by the symbol **&&**
2. Or operation represented by the symbol **||**
3. Not operation represented by the symbol **!**



Assume variable **a** holds **1** and **b** holds **0**.

Symbol	**Meaning**	**Example of Syntax**
&&	**AND operator.** If both the values are nonzero then it is true otherwise false its format is the following: (condition1 && condition2)	**(a<b && b>a)** will evaluate to false/0 **(1<0 && 0>1)**will evaluate to false/0 **(a>b && b<a)**will evaluate to true/1 **(1>0 && 0<1)** will evaluate to true/1
||	**OR operator.** If any of the value is nonzero then it is true otherwise false	**(a<b || b>a)** will evaluate to false/0 **(1<0 || 0>1)** will evaluate to false/0 **(a>b || b>a)**will evaluate to true/1 **(1<0 || 0<1)** will evaluate to true/1
!	**NOT operator.** Use to reverses the logical state of its	**!(a<b && b>a)**will evaluate to true/1 **!(1<0 || 0>1)**will evaluate to true/1

	operand. If a condition is true, then Logical NOT operator will make it false. Similarly if the condition is false then the logical NOT will make it true.	**!(a>b && b<a)**will evaluate to false/0 **Similarly** **!(1<0 || 0<1)**will evaluate to false/0

And Operator

This operator combines multiple conditions and if all condition become true then the whole condition changes with a **true.** If only one condition is false then whole condition changes with **false.**

Or Operator

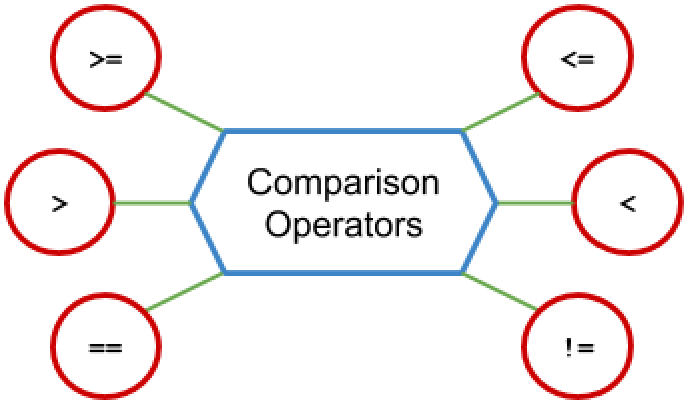
This operator also combines multiple conditions and if any condition become true then the whole condition changes with a **true**. In this case if all the condition are false then whole condition changes to **false** otherwise it's **true**

Not Operator

This operator in C++ inverts the result. If the condition is true it convert it into **false**, and if it's false then it converts to **true**

Remember
In C++ non-zero value in any condition is true and zero is considered as false.

Comparison operators



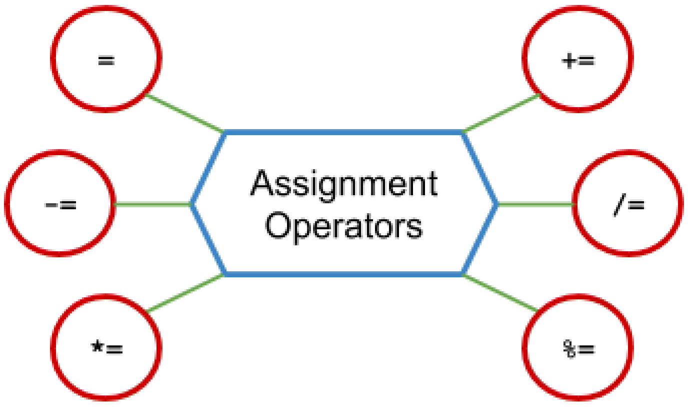
Assume that variable **x** holds **5** and **y** holds **6**.

Operators	**Meanings**	**Example**
<	true/1: if x is less than y else false/0	**x < y** (true/1) **y < x** (false/0)
>	true/1: if x is greater than y else false/0	**y > x** (true/1) **x > y** (false/0)
<=	true/1: if x is less than or equal to y else false/0	**x <= y** (true/1) **x <= x** (true/1)

		y <= x (false/0)
>=	true/1: if x is greater than or equal to y else false/0	**y >= x** (true/1) **y >= y** (true/1) **x >= y** (false/0)
==	true/1: if x is equal to y else false/0	**x == x** (true/1) **x == y** (false/0)
!=	true/1 if x is not equal to y else false/0	**x != y** (true/1) **x != x** (false/0)

The result of such operations is either **true** or **false**.

Assignment operators



Assume that variable **z** holds **5**.

Operators	**Meanings**	**Example**
=	To assign a value to left (store 5 in a), you can also assign a value of variable in another variable(e.g **a = b**)	**a = 5**
+=	Add right operand to left operand and store result in left operand	**z += 3** is equivalent to **z = z + 3**
-=	Subtract right operand to left operand and store result in left operand	**z -= 3** is equivalent to **z = z - 3**
***=**	Multiply right operand to left operand and store result in left operand	**z *= 3** is equivalent to **z = z * 3**
/=	Divide right operand to left operand and store result in left operand	**z /= 3** is equivalent to **z = z / 3**

%=	Divide right operand to left operand and store remainder in left operand	**z %= 3** is equivalent to **z = z % 3**

Type casting

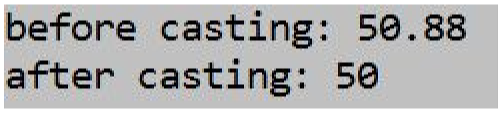
Type casting is the process of converting an expression of one type into another. You will understand it better when you see the example below

Example 1

Code

```
1  #include<iostream>
2  using namespace std;
3
4  //example 1 of type casting
5  int main()
6  {
7     cout << "before casting: ";
8     float a = 50.88;
9     cout << a << endl;
10
11    cout << "after casting: ";
12    int b = a;
13    cout << b << endl;
14
15    return 0;
16 }
```

Output



```
before casting: 50.88
after casting: 50
```

Description

On line 8 there is a variable of **float** data type named **a** and assigned a value 50.88 and now on line 12 value of **a** is assigning to integer **b** here variable of float type assigned to **a** integer, now what happened !!!!. The integer **b** stored only the whole part which means only 50 it just missed the part after point .

Another method of example 1

This example is just like above example but a little bit change but it will do the same.

Code

```
#include<iostream>
using namespace std;

//another variant of example 1 of type casting
int main()
{
    cout << "before casting: ";
    float a = 50.88;
    cout << a << endl;

    cout << "after casting: ";
    cout << int(a) << endl;    //you can also write it like (int)a

    return 0;
}
```

Output

```
before casting: 50.88
after casting: 50
```

Description

In this example we are doing the previous task by a new way. Here we did not make another variable we just make 1 variable of **float** type and forcefully show it as an integer while printing on console screen.

Example 2

Code

```
#include<iostream>
using namespace std;

//example 2 of type casting
int main()
{
    cout << "before casting: ";
    char symbol = 'A';
    cout << symbol << endl;

    cout << "after casting: ";
    cout << int(symbol) << endl;   //you can also write it like (int)symbol

    return 0;
}
```

Output

```
before casting: A
after casting: 65
```

Description

In this example we are just casting character into a integer. Previously we convert **float** to a **int** both were numbers now here is a different case we are converting a character into a integer. Here after casting character casted into its ASCII value.

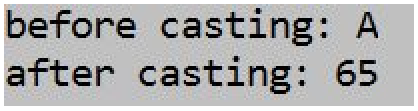
Another method of example 2

This example is just like above example but a little bit change but act same.

Code

```cpp
#include<iostream>
using namespace std;

//another variant of example 2 of type casting
int main()
{
    cout << "before casting: ";
    char symbol = 'A';
    cout << symbol << endl;

    cout << "after casting: ";
    int a = symbol;
    cout << a << endl;

    return 0;
}
```

Output



```
before casting: A
after casting: 65
```

Description

This example is another variant of previous one in this we make another integer variable and store the ASCII value in that variable.

Variable scope

In C++ variables is only accessible within their boundaries which is called scope of variables, there is no access of variables outside the boundaries. It is very important to know the scope of variable. The variable scope has two types

- Local variable
- Global variable

1. Local variable

Local variables are only accessible within curly brackets { } where it is declared, outside this curly brackets variables is not accessible because it is a part of that scope in which it is declared. Curly brackets are basically scope. Now look at the example below very carefully.

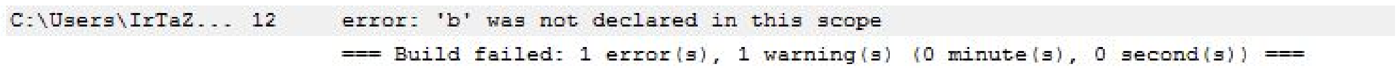
Example of local variable scope

Code

```
 1  // Local variable scope
 2  #include <iostream>
 3  using namespace std;
 4
 5  int main()
 6  {
 7     int a = 100;
 8     {
 9        int b = 200;
10     }
11     cout << a << endl;
12     cout << b << endl;
13     return 0;
14  }
```

Description

In this example there are two integer variables **a** and **b** is declared on line 7 and 9, variable **a** is declared in **main** scope so you can access it in whole **main** ,but variable **b** is inside curly brackets this is the scope of variable **b** outside this curly brackets variable **b** is not accessible as you see in code above there is error on line 12.



```
C:\Users\IrTaZ...  12     error: 'b' was not declared in this scope
                          === Build failed: 1 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===
```

Because variable **b** is not accessible outside its scope(curly brackets). We can only access variable **b** inside his scope as shown in the code below

```
 1  // Local variable scope
 2  #include <iostream>
 3  using namespace std;
 4
 5  int main()
 6  {
 7     int a = 100;
 8     {
 9        int b = 200;
10        cout << b << endl;
11     }
12     cout << a << endl;
13     return 0;
14  }
```

Output

```
200
100
```

Now you see the output because we access variable **b** inside his scope.

2. Global variable

Global variables are those variables which is accessible in whole program. Global variables must be declared outside the **int main().** You can initialize global variable anywhere in program. Global variable is automatically initialized with default value(if it is not initialized) which are as follows

Data Type	Initializer
int	0
char	‘\0’ (Null Character)
float	0
double	0

Example of global variable scope

Code

```
// global variable scope
#include <iostream>
using namespace std;

int a;
int main()
{
    cout << a << endl;
    a = 100;
    cout << a << endl;
    return 0;
}
```

Output

```
0
100
```

Description

In this example global variable is declared outside the **int main()** at line 5, at 8 we simply display value of variable which is automatically initialized with **0** because it is not initialized at line 5. At line 9 variable **a** is initialized with **100** and then we display that variable again but this time output is **100**.

Remember
Local variable do not initialize automatically like global variable, you have to initialize local variable otherwise it has garbage value(any random number given by computer).

Let's start our main flow of this chapter

Sometime in programming we are needed to run only selected statements on some conditions so for handling these type of situation this chapter is written. This chapter is all about such a flow in which we decide which set of statement will execute and which will not.

These type of decision flows are as follow:
- **if statement.**
- **if else statement.**
- **switch statement.**

The `if` statement

Variant 1

```
if(condition)
      Statement
```

```
if(condition)
{
      Statement
}
```

Variant 2

```
if(condition 1)
      Statement 1
if(condition 2)
      Statement 2
if(condition 3)
      Statement 3
```

```
if(condition 1)
{
      Statement 1
}
if(condition 2)
{
      Statement 2
}
if(condition 3)
{
      Statement 3
}
```

If statement is just like english language.
for example: if it's raining I will use umbrella so that I don't get wet.

In C++ **if** statement is same, if the condition is true then the statement inside the **if** executed otherwise not.

The if else statement

Variant 1

```
if(condition)
      Statement 1
else
      Statement 2
```

```
if(condition)
{
      Statement 1
}
else
{
      Statement 2
}
```

Variant 2

```
if(condition 1)
      Statement 1
else if(condition 2)
      Statement 2
else if(condition 3)
      Statement 3
else
      Statement
```

```
if(condition 1)
{
      Statement 1
}
else if(condition 2)
{
      Statement 2
}
else if(condition 3)
{
      Statement 3
}
else
{
      Statement
}
```

If else statement is also like if statement but with a slightly difference e.g. If there is a speed breaker then we apply breaks otherwise we move forward with same speed(here otherwise is used as **else**).
In C++ **if else** statement is same, if the condition is true then the portion of if statement will execute otherwise portion of **else** statement will execute.

The switch statement

```
switch(condition)
{
      case constant-expression:
            Statement
            break;
      case constant-expression:
            Statement
            break;
      case constant-expression:
            Statement
            break;
      .
      .
      .
      .
      default:
            statement
}
```

Switch statement is the extended form of **if else** statement but with a slightly change which is that in **if else** you use a condition but in **switch** you use only a statement.
Now let’s see the examples of these variants.
When no case is executed the **default** statement is executed. It is not compulsory to use this but it is good programming practice.

The if statement (Variant 1) example

Code

```
1   #include<iostream>
2   using namespace std;
3
4   //example of if statement(variant 1)
5   int main()
6   {
7       int a;
8       cout << "enter number: ";
9       cin >> a;
10
11      if(a < 30)
12      {
13          cout << "a is smaller than 30." << endl;
14      }
15      return 0;
16  }
```

Output

```
enter number: 10
a is smaller than 30.
```

Description

This example is of if statement. In this example we will find out either the entered number is smaller than thirty or not. If the number is smaller than thirty then the control of program will execute inside the if portion otherwise the control will skip the if part from execution.
On line 11 the if statement starts here we write its condition(**a < 30**) because of which it will execute the if’s part.and on the line 13 the internal part of if is written which is just a simple message. If the number is greater than 29 then there is no output on screen because the condition (**a < 30**) become false.

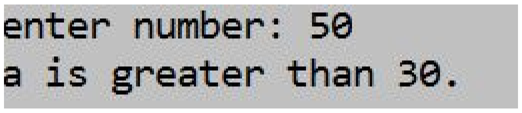
Link: http://codepad.org/Bhc3GX5o

The `if` statement (Variant 2) example

Code

```
 1  #include<iostream>
 2  using namespace std;
 3
 4  //example of if statement(variant 2)
 5  int main()
 6  {
 7      int a;
 8      cout << "enter number: ";
 9      cin >> a;
10
11      if(a < 30)
12      {
13          cout << "a is smaller than 30." << endl;
14      }
15      if(a > 30)
16      {
17          cout << "a is greater than 30." << endl;
18      }
19      if(a == 30)
20      {
21          cout << "a is equal to 30." << endl;
22      }
23      return 0;
24  }
```

Output



```
enter number: 50
a is greater than 30.
```

Description

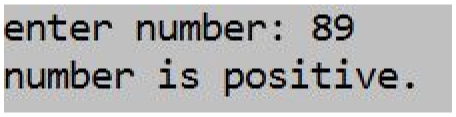
This example is extended version of the previous example. In this example multiple **if** statements are used with different conditions for different case. First on line 11 if is written there so the program check either the condition is true or false, if the condition is true then our program will execute the first **if** part otherwise in case of false statement it will skip and check the next **if** statement, the flow is similar for every **if** statement if one condition is true our program will not skip the other ones it will check every condition, execution of if statement is depends upon its respective condition.

Link: http://codepad.org/ZJZ3IOE0

The if else statement (Variant 1) example
Code

```
1   //example of if else statement(variant 1)
2   #include<iostream>
3   using namespace std;
4
5   int main()
6   {
7       int num;
8       cout << "enter number: ";
9       cin >> num;
10
11      if(num > 0)
12      {
13          cout << "number is positive." << endl;
14      }
15      else
16      {
17          cout << "number is negative" << endl;
18      }
19      return 0;
20  }
```

Output



```
enter number: 89
number is positive.
```

Description

In this example **if** statement is on line 11 with condition (**num > 0**) if the user entered number is greater than 0 than the body of **if** will execute otherwise body of else will execute in every other condition either it is not mentioned else is on line 16 to 18. In if else statement only one statement must have to execute either it can be if's part or it can the part of else.

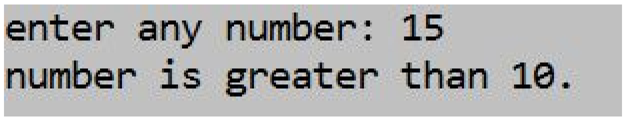
Link: http://codepad.org/XevmAEgb

The if else statement (Variant 2) example

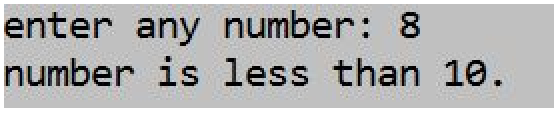
Code

```cpp
//example of if else statement(variant 2)
#include<iostream>
using namespace std;

int main()
{
   int num;
   cout << "enter any number: ";
   cin >> num;

   if(num > 10)
   {
      cout << "number is greater than 10." << endl;
   }
   else if(num < 10)
   {
      cout << "number is less than 10." << endl;
   }
   else
   {
      cout << "number is equal to 10." << endl;
   }
   return 0;
}
```

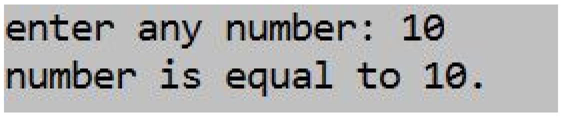
Output



```
enter any number: 15
number is greater than 10.
```



```
enter any number: 8
number is less than 10.
```



```
enter any number: 10
number is equal to 10.
```

Description

This example is slightly different from the previous one, here the **else** part is attached with an **if** which means the **else** part also have some condition. Not exactly as its own condition but to enter in **else** part you have to check either the condition is true or false. In **if else** statement if one **if's** statement become true then it will skip the other **if** statment which are attached with the **if else-if** part.

Link: http://codepad.org/bqxXd4y3

Program 1: Digits Summation (using operators +, %, /)

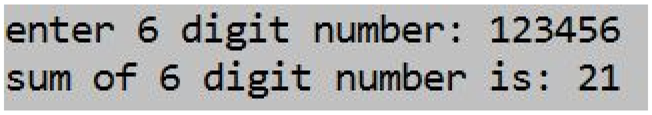
Write and run a program that reads a six-digit integer and prints the sum of its six digits

Sample Input: 153426
Sample output: 21

Code

```cpp
1  #include <iostream>
2  using namespace std;
3  // This program print the sum of 6 digit number
4  int main()
5  {
6     int number;
7     int sum = 0;
8     cout << "enter 6 digit number: ";
9     cin >> number;
10    if(number > 999999)
11    {
12       cout << "Invalid Input.";
13    }
14    else
15    {
16       sum = sum + number % 10;
17       number = number / 10;
18       sum = sum + number % 10;
19       number = number / 10;
20       sum = sum + number % 10;
21       number = number / 10;
22       sum = sum + number % 10;
23       number = number / 10;
24       sum = sum + number % 10;
25       number = number / 10;
26       sum = sum + number % 10;
27       cout << "sum of 6 digit number is: " << sum << endl;
28    }
29    return 0;
30 }
```

Output



```
enter 6 digit number: 123456
sum of 6 digit number is: 21
```

Description

In this program user enter six digit number or less digit number, then this code separate them one by one and add them together that will be the outcome (result) of this code. At line 10 **if** check is applied to check either the number is greater than 6 digit number or not, if the number is greater than 6 digit number then display “invalid input” otherwise proceed to **else** part At line 16 we will take modulus(**%**) with 10 so the last digit of the entered number can be obtained and add that number in **sum** variable remember when you take a modulus of number with 10 the last digit is separated. At line 17 division is applied with 10 so the last digit which is saved or added can be remove from the actual number, the same concept is applied but division cut the last number from 6 digit number and after division the total digits are 5 this whole process is repeated 5 times. After this whole process we will get a sum of 6 digit number which we input in variable **sum** as answer.

Link: http://codepad.org/pCAKYT7Z

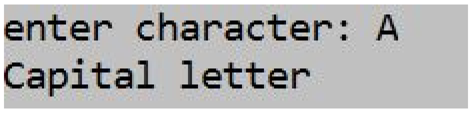
Program 3.1: Capital Letter or Small Letter (using operators >=, <=, &&)

Take a character input from the user and tell whether the character is a capital letter or a small letter.
Sample Input: A
Sample Output: Capital letter
Sample Input: g
Output: Small letter
Sample Input:)
Sample Output: Non

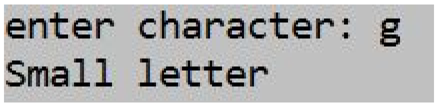
Code

```cpp
// This program tell either the alphabet is Capital letter or Small letter
#include <iostream>
using namespace std;

int main()
{
   char character;
   int symbol;
   cout << "enter character: ";
   cin >> character;
   symbol = int(character);

   if(symbol >= 65 && symbol <= 90)
   {
      cout << "Capital letter" << endl;
   }
   else if(symbol >= 97 && symbol <= 122)
   {
      cout << "Small letter" << endl;
   }
   else
   {
      cout << "Non" << endl;
   }
   return 0;
}
```

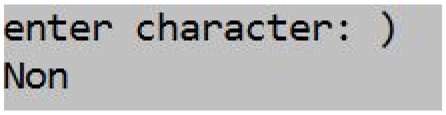
Output



```
enter character: A
Capital letter
```



```
enter character: g
Small letter
```



```
enter character: )
Non
```

Description

This program simply tells that user entered alphabet is capital letter or small letter. **char**(character) data type is used in this program to initialize the char following method is used

char letter = ‘A’;

Remember that **char** data type only hold single alphabet and it is always inside single quotes.

In this example when user enter an alphabet it is type casted in integer variable **symbol** on line 11 because it is syntax of converting any data type with the integer data type, and then on line 13 **if** statement’s condition will check that stored number in **symbol** match correspondingly to ASCII value that it is between 65 to 90 or not, if the value of **symbol** is not between 65 to 90 it check the **else if** on line 17 for small letter, and at the last on line 21 if both (**if** and **else if**) is not executed the **else** will be execute.

Remember

Computer only understand numbers so every character or special character have some numerical representation for example the numerical representation of letter “A” is 65 in ASCII. ASCII stands for American Standard Code for Information Interchange. It's a worldwide standard of codes for numerical representation.
65 to 90 is ASCII value of capital A to Z.
97 to 122 is ASCII value of small a to z.

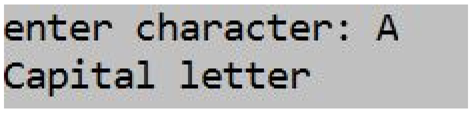
Link: http://codepad.org/kfQXuCvg

Program 3.2: Capital Letter or Small Letter(Variant 2) (using operators >=, <=, &&)

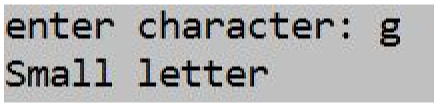
Code

```
 1  //This program tell either the alphabet is Capital alphabet or Small Letter
 2  #include <iostream>
 3  using namespace std;
 4  
 5  int main()
 6  {
 7     char character;
 8     cout << "enter character: ";
 9     cin >> character;
10  
11     if(character >= 'A' && character <= 'Z')
12     {
13        cout << "Capital letter" << endl;
14     }
15     else if(character >= 'a' && character <= 'z')
16     {
17        cout << "Small letter" << endl;
18     }
19     else
20     {
21        cout << "Non" << endl;
22     }
23     return 0;
24  }
```

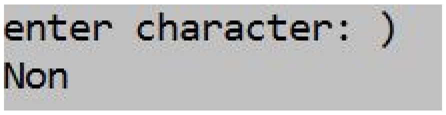
Output



```
enter character: A
Capital letter
```



```
enter character: g
Small letter
```



```
enter character: )
Non
```

Description

This program is slightly different from previous one, but it is a same program with a little bit of difference in code. In this program type casting is not used, when user enter any alphabet it is directly check with alphabet at line 11 that it is capital or not, at line 15 it check that letter is small or not. And if not capital or small then obviously entered character is not an alphabet. As you see in previous example we use ASCII value to check but in this example we directly check by alphabet at line 11 and 15. Character always written inside single quotes ‘’.

Link: http://codepad.org/RTz9RUyC

Program 4: Multiple of Each Other (using operators %, ==, if, else if, else)

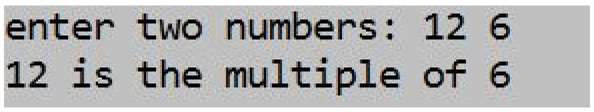
Write and run a program that reads two integers and then uses the conditional expression operator to print either “multiple” or “not” according to whether one of the integers is a multiple of the other.

Sample Input: 12 6
Output: 12 is the multiple of 6
Sample Input: 12 13
Output: NON

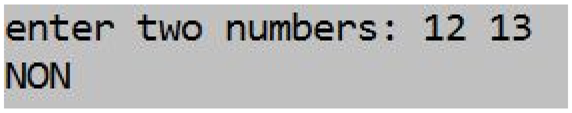
Code

```
1    // This program tell the numbers is multiple of each other or not
2    #include <iostream>
3    using namespace std;
4
5    int main()
6    {
7        int value1;
8        int value2;
9        cout << "enter two numbers: ";
10       cin >> value1 >> value2;
11
12       if(value1%value2 == 0)
13       {
14           cout << value1 << " is the multiple of " << value2 << endl;
15       }
16       else if(value2%value1 == 0)
17       {
18           cout << value2 << " is the multiple of " << value1 << endl;
19       }
20       else
21       {
22           cout << "NON" << endl;
23       }
24       return 0;
25   }
```

Output



```
enter two numbers: 12 6
12 is the multiple of 6
```



```
enter two numbers: 12 13
NON
```

Description

In this program user enter two numbers, at line 12 it check if the modulus of **value1** to **value2** is equal to zero(means if it is completely divisible and their remainder is zero) then statements of first **if** will execute, otherwise condition at line 16 is checked which is modulus of **value2** to **value1** either that is equal to zero. If both the conditions is false then **else** statement will execute which means non of two numbers are multiple of others as shown in output above. Modulus is actually a remainder when we divide 2 numbers..

Link: http://codepad.org/HVzfYjlq

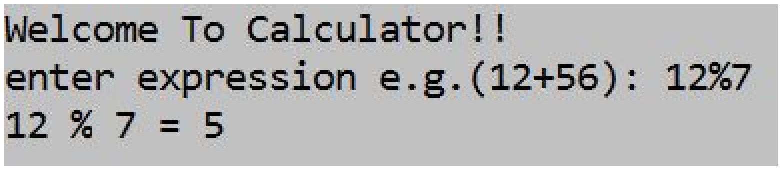
Program 5: Simple Calculator (using operators +, -, *, %, /, if, else if, else statements)

Write and run a program that simulates a simple calculator. It reads two integers and a character. If the character is a '+', the sum is printed; if it is a '-', the difference is printed; if it is a '*', the product is printed; if it is a '/', the quotient is printed; and if it is a '%', the remainder is printed.
Sample Input: 12%7
Sample Output: 5
Sample Input: 19*10
Sample Output: 190

Code

```
// This program do simple arithmetic operations on two given numbers
// Simple Calculator
#include <iostream>
using namespace std;

int main()
{
   int num1;
   int num2;
   char symbol;
   cout << "Welcome To Calculator!!" << endl;
   cout << "enter expression e.g.(12+56): ";
   cin >>  num1 >> symbol >> num2;

   switch(symbol)
   {
   case '+':
      cout << num1 << " + " << num2 << " = " << num1+num2 << endl;
      break;
   case '-':
      cout << num1 << " - " << num2 << " = " << num1-num2 << endl;
      break;
   case '/':
      cout << num1 << " / " << num2 << " = " << num1/(num2 * 1.00) << endl;
      break;
   case '%':
      cout << num1 << " % " << num2 << " = " << num1%num2 << endl;
      break;
   case '*':
      cout << num1 << " * " << num2 << " = " << num1*num2 << endl;
      break;
   default:
      cout << "invalid symbol." << endl;
   }
   return 0;
}
```

Output



```
Welcome To Calculator!!
enter expression e.g.(12+56): 12%7
12 % 7 = 5
```

Description

This program act as a simple arithmetic calculator in this program user enter first number, symbol(operation wants to perform) and second number on line 13, both numbers are stored in two integer variables(**num1, num2**) and symbol is stored in **char** variable(**symbol**) then on line 15 **switch** statement with condition **symbol** check the symbol and then case inside the body of switch is executed according to entered symbol. Case acts like **if** statement if anyone of the case is executed it **breaks** it and no further **case** is checked. In statement of **case** we just do simple arithmetic as we did in chapter 3.

Link: http://codepad.org/ZpSgFLmx

Program 6: Section of Highest Average(using operators >, if statement)

PF has 6 sections we are required to find out which sections average is higher. Write a program which takes each section's PF average and Output which section has won w.r.t average.

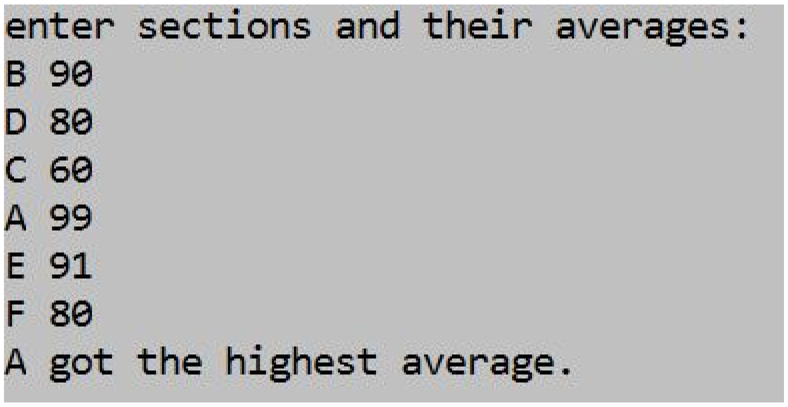
Sample Input:	B	90
	D	80
	C	60
	A	99
	E	91
	F	80

Output: A got the highest average.

Code

```
// This program shows which section got highest average
#include <iostream>
using namespace std;

int main()
{
    char sec1, sec2, sec3, sec4, sec5, sec6;
    int avg1, avg2, avg3, avg4, avg5, avg6;
    cout << "enter sections and their averages: " << endl;
    cin >> sec1 >> avg1;
    cin >> sec2 >> avg2;
    cin >> sec3 >> avg3;
    cin >> sec4 >> avg4;
    cin >> sec5 >> avg5;
    cin >> sec6 >> avg6;

    int max_average;
    char max_section;
    max_average = avg1;
    max_section = sec1;
    if (avg2 > max_average)
    {
        max_average = avg2;
        max_section = sec2;
    }
    if (avg3 > max_average)
    {
        max_average = avg3;
        max_section = sec3;
    }
    if (avg4 > max_average)
    {
        max_average = avg4;
        max_section = sec4;
    }
    if (avg5 > max_average)
    {
        max_average = avg5;
        max_section = sec5;
    }
    if (avg6 > max_average)
    {
        max_average = avg6;
        max_section = sec6;
    }
    cout << max_section << " got the highest average." << endl;
    return 0;
}
```

Output



```
enter sections and their averages:
B 90
D 80
C 60
A 99
E 91
F 80
A got the highest average.
```

Description

This program check which section got highest average score. In this program six **char** variable and six **int** variable is declared on line 7 and 8. In this program user enter six sections(which is alphabets) and their averages. We make two more variables one is **char** and other is **int** on line 19 and 20 and we assign them first section and it's score so we can compare with other's, after that we simply check next section average from previous one in five if's on line 21, 26, 31, 36 and 41. If the next section score is greater than previous one then it is assigned to **max_average** variable and its section to **max_section** variable, at the end on line 46 we simply display the highest section which got highest average score.

Link: http://codepad.org/xmBPqhnU

EXERCISE

- *Will the above program will work if we use if then else instead of all ifs?*

Program 7: Roll Number of Highest Aggregate(using operators >, if statement)

Write a program which takes marks of 5 courses as input, of 5 students and output the students who got the highest aggregate.

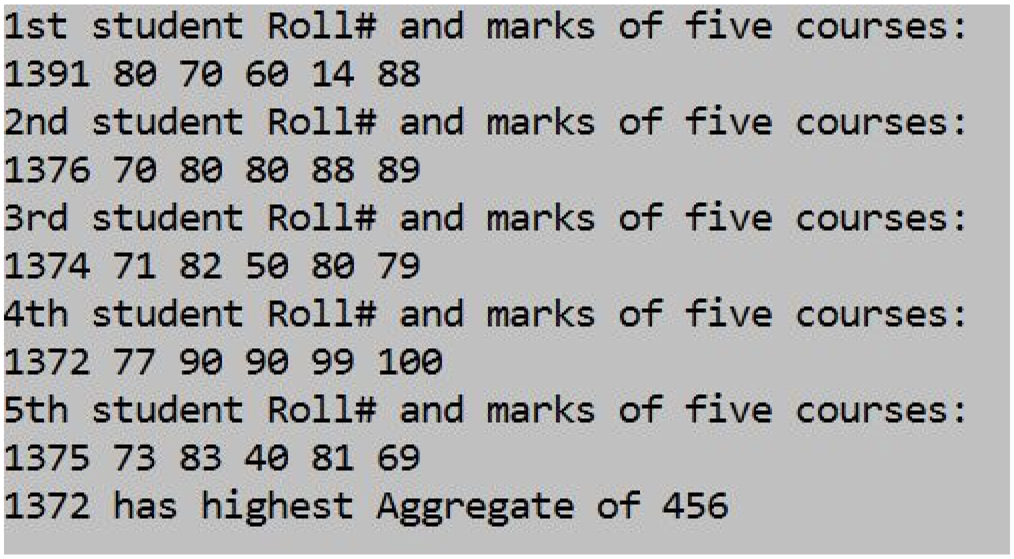
	Roll#	C1	C2	C3	C4	C5
Sample Input:						
	1391	80	70	60	14	88
	1376	70	80	80	88	89
	1374	71	82	50	80	79
	1372	77	90	90	99	100
	1375	73	83	40	81	69

Sample Output: 1372 has highest Aggregate of 456

Code

```
// This program shows which roll number got highest aggregate
#include <iostream>
using namespace std;

int main()
{
   int r_no, max_r_no, max_total = 0, sum;
   int c1, c2, c3, c4, c5;

   cout << "1st student Roll# and marks of five courses: " << endl;
   cin >> r_no >> c1 >> c2 >> c3 >> c4 >> c5;
   sum = c1+c2+c3+c4+c5;
   max_total = sum;
   max_r_no = r_no;
   cout << "2nd student Roll# and marks of five courses: " << endl;
   cin >> r_no >> c1 >> c2 >> c3 >> c4 >> c5;
   sum = c1+c2+c3+c4+c5;
   if(sum > max_total)
   {
      max_total = sum;
      max_r_no = r_no;
   }
   cout << "3rd student Roll# and marks of five courses: " << endl;
   cin >> r_no >> c1 >> c2 >> c3 >> c4 >> c5;
   sum = c1+c2+c3+c4+c5;
   if(sum > max_total)
   {
      max_total = sum;
      max_r_no = r_no;
   }
   cout << "4th student Roll# and marks of five courses: " << endl;
   cin >> r_no >> c1 >> c2 >> c3 >> c4 >> c5;
   sum = c1+c2+c3+c4+c5;
   if(sum > max_total)
   {
      max_total = sum;
      max_r_no = r_no;
   }
   cout << "5th student Roll# and marks of five courses: " << endl;
   cin >> r_no >> c1 >> c2 >> c3 >> c4 >> c5;
   sum = c1+c2+c3+c4+c5;
   if(sum > max_total)
   {
      max_total = sum;
      max_r_no = r_no;
   }
   cout << max_r_no << " has highest Aggregate of " << max_total << endl;

   return 0;
}
```

Output



```
1st student Roll# and marks of five courses:
1391 80 70 60 14 88
2nd student Roll# and marks of five courses:
1376 70 80 80 88 89
3rd student Roll# and marks of five courses:
1374 71 82 50 80 79
4th student Roll# and marks of five courses:
1372 77 90 90 99 100
5th student Roll# and marks of five courses:
1375 73 83 40 81 69
1372 has highest Aggregate of 456
```

Description

This program is slightly different from previous in this program user enter roll number and marks of five course five time, first time roll number assigned to **max_r_no** and sum of five courses assigned to **max_total** for the rest of four it check if it is greater than previous aggregate then it assigned to **max_total** as you see in lines 18, 26, 34 and 42 the **if** condition check if **sum** is greater than **max_total** assign the sum to **max_total** and roll number to **max_r_no,** and at the end the highest aggregate is displayed.

Link: http://codepad.org/nZshacLy

Program 8: Digit to Word(using switch statement)

Write a program which will take a digit number and output digit in word.

Sample Input: 7

Output: Seven

Code

```
// This program print digit into word
#include <iostream>
using namespace std;

int main()
{
    int num;
    cout << "enter one digit number: ";
    cin >> num;

    switch(num)
    {
    case 1:
        {
            cout<<" One ";
            break;
        }
    case 2:
        {
            cout<<" Two ";
            break;
        }
    case 3:
        {
            cout<<" Three ";
            break;
        }
    case  4:
        {
            cout<<" Four ";
            break;
        }
    case 5:
        {
            cout<<" Five ";
            break;
        }
    case 6:
        {
            cout<<" Six ";
            break;
        }
    case 7:
        {
            cout<<" Seven ";
            break;
        }
    case 8:
        {
            cout<<" Eight ";
            break;
        }
    case 9:
        {
```

```
55          cout<<" Nine ";
56          break;
57        }
58      case 0:
59        {
60          cout<<" Zero ";
61        }
62    }
63    return 0;
64  }
```

Output

```
enter one digit number: 7
 Seven
```

Description

In this program user just enters single digit number and program tells that digit into word as you show in output above. In this program **switch** statement is used on line 11 and its cases is checked one by one, when the number is matched with **case** constant-expression the program display its statement and **break**.

Link: http://codepad.org/GhodbPE4

Exercise (do it yourself)

Write a program which will take at max: a six digit number and output each of its digit in words, (Bonus) if the number is less than 6 digits it should not out print initial zeros. If the number is greater than 6 digits then it should output wrong input.

Sample Input: 651432
Output: Six Five One Four Three two
(Bonus)Sample Input: 1432
Sample Output: One Four Three two.

Output of 6 digit number(exercise question)

```
enter six digit number: 651432
 Six  Five  One  Four  Three  Two
```

Hint: use example 1

Program 9: Shape Recognizer using Coordinates(using operators +, *, ==, &&, !=)

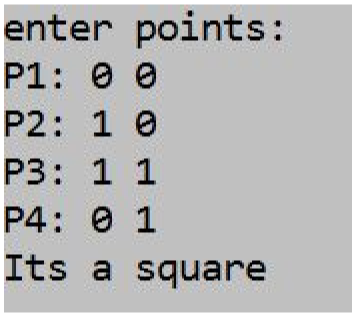
Write a program which takes as input 4 points and tell whether these points are the coordinates of Rectangle, Square or Quadrilateral.

Sample Input: P1 0 0
P2 1 0
P3 1 1
P4 0 1
Output: Its a square

Code

```cpp
// This program determines the shape by its coordinates
#include <iostream>
using namespace std;

int main()
{
    int p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y;
    int d1,d2,d3,d4,dg1,dg2;
    cout << "enter points: " << endl;
    cout << "P1: ";
    cin >> p1x >> p1y;
    cout << "P2: ";
    cin >> p2x >> p2y;
    cout << "P3: ";
    cin >> p3x >> p3y;
    cout << "P4: ";
    cin >> p4x >> p4y;

    d1=((p2x-p1x)*(p2x-p1x))+((p2y-p1y)*(p2y-p1y));
    d2=((p3x-p2x)*(p3x-p2x))+((p3y-p2y)*(p3y-p2y));
    d3=((p4x-p3x)*(p4x-p3x))+((p4y-p3y)*(p4y-p3y));
    d4=((p1x-p4x)*(p1x-p4x))+((p1y-p4y)*(p1y-p4y));
    dg1=((p3x-p1x)*(p3x-p1x))+((p3y-p1y)*(p3y-p1y));
    dg2=((p4x-p2x)*(p4x-p2x))+((p4y-p2y)*(p4y-p2y));

    if (d1==d2 && d1==d3 && d1==d4)
    {
        if (dg1==dg2)
        {
            cout << "Its a square" << endl;
        }
        else
        {
            cout << "Its a rhombus" << endl;
        }
    }
    else if (d1==d3 && d2==d4 && d1!=d2)
    {
        if (dg1==dg2)
        {
            cout << "Its a rectangle" << endl;
        }
        else
        {
            cout << "Its a parallelogram" << endl;
        }
    }
    else
    {
        cout << "Its a quadrilateral" << endl;
    }
    return 0;
}
```

Output



```
enter points:
P1: 0 0
P2: 1 0
P3: 1 1
P4: 0 1
Its a square
```

Description

This example is just for identifying either the entered points are forming a quadrilateral, square or rectangle. In this program user enters four points with their x and y axis (total 8). Then we just calculate each side of quadrilateral (4 sides) and it's both diagonals.
For calculating the sides and diagonals distance formula is used which is used as D=(X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1).
Actually in original formula there is a square root of whole RHS of formula but we are not complicating it. We can do our task without taking square root because here we are just comparing so if a number is greater than other than their square root is also greater than other number. After calculating the sides (**d1,d2,d3,d4**) and diagonals (**dg1,dg2**) on line 19,20,21,23 and 24 we just compare them with another. If all the sides are of same length which is checked on line 26 then it can be a square or a rhombus and then if the diagonals are also of same length which is checked on line 28 then it is definitely a square and if sides are same but diagonal are not than it will be a rhombus. Now if the opposite sides are equal in length which is checked on line 37 then it can be a rectangle or a parallelogram now we compare diagonals if diagonals are equal which is checked on line 39 then it is a rectangle but if diagonals are not same then it is definitely a parallelogram. Now if any side is not equal to anyone then it is a simple any quadrilateral.

Link: http://codepad.org/zHi5U3cK

Program 10: Triangle Shape Recognizer(using operators +, -, *, ==, &&, !=, ||)

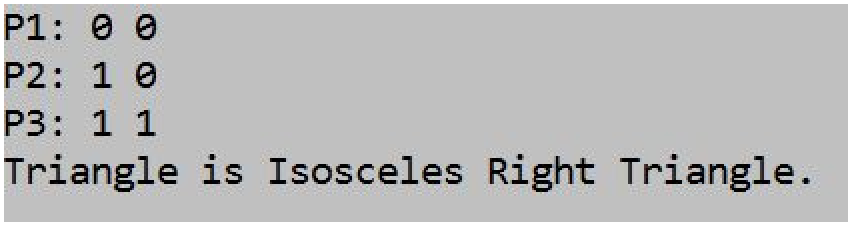
Write a program which takes as input 3 points and tell whether these points are the coordinates of isosceles or equilateral or right angled or scalene triangle.

Sample Input: P1 0 0
P2 1 0
P3 1 1
Sample Output: Right Angle Triangle

Code

```cpp
// This program determines the shape of triangle by its coordinates
#include <iostream>
using namespace std;

int main()
{
    int p1x, p1y, p2x, p2y, p3x, p3y, s1, s2, s3 ;
    cout << "P1: ";
    cin >> p1x >> p1y;
    cout << "P2: ";
    cin >> p2x >> p2y;
    cout << "P3: ";
    cin >> p3x >> p3y ;
    s1 = ((p2x-p1x)*(p2x-p1x)) + ((p2y-p1y)*(p2y-p1y));
    s2 = ((p3x-p2x)*(p3x-p2x)) + ((p3y-p2y)*(p3y-p2y));
    s3 = ((p3x-p1x)*(p3x-p1x)) + ((p3y-p1y)*(p3y-p1y));

    if (s1==s2 && s2==s3)
    {
        cout<<"Triangle is Equilateral Triangle.";
    }
    else if ( s1==s2 || s1==s3 || s2==s3 )
    {
        if ( s1==s2+s3 || s2==s1+s3 || s3==s1+s2 )
        {
            cout<<"Triangle is Isosceles Right Triangle." << endl;
        }
        else cout<<" Triangle is Isosceles Triangle." << endl;
    }
    else if ( s1==s2+s3 || s2==s1+s3 || s3==s1+s2 )
    {
        cout<<" Triangle is Right Angled Triangle." << endl;
    }
    else
    {
       cout<<" Triangle is Scalene Triangle." << endl;
    }
    return 0;
}
```

Output



```
P1: 0 0
P2: 1 0
P3: 1 1
Triangle is Isosceles Right Triangle.
```

Description

Here we have a program which will take six integers as input and tell that the triangle form by the points/integers enter is a equilateral, scalene ,isosceles or right angle triangle. First of all we enter all the inputs as the points then for the calculation to figure out what triangle is depends on the distance between the given points. So first of all we calculate the distance using distance formula. Then we put checks on the distance first check is on line number 18 that if all the distances are equal to each other then the output will be that triangle is equilateral. Then on the line number 22 there is the second check which is for isosceles triangle or isosceles right triangle. Firstly we check that if the two distances are same then it comes in the condition and check if it is a right angle triangle by using the pythagoras theorem If the check is right so the output will be Isosceles right triangle and if the check is false then then output will be Isosceles triangle. Then on line number 30 there is a check using pythagoras theorem that whether the triangle is right angle triangle or not. At last on line number 34 there is an else condition in which there is only an output after all checks that triangle is scalene.

Link: http://codepad.org/3rXzMsg2

Program 11: Point Check(using operators >, <, &&, || and if else statement)

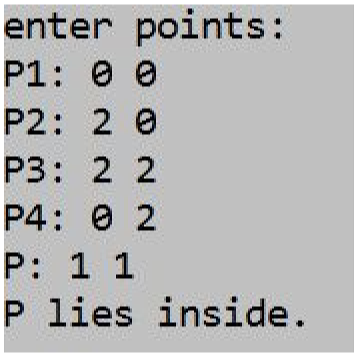
Take 4 coordinates of the Rectangle and a point P. Your program should be able to tell whether P lies inside the Rectangle or Not.

```
Sample Input: P1    0 0
              P2    2 0
              P3    2 2
              P4    0 2
              P     1 1
Output: P lies inside Square
```

Code

```
//This program determines the point lies within the square or not
#include<iostream>
using namespace std;

int main()
{
    char choice;
    int p1x,p1y,p2x,p2y,p3x,p3y,p4x,p4y,px,py;
    cout << "enter points:" << endl;
    cout << "P1: ";
    cin >> p1x >> p1y;
    cout << "P2: ";
    cin >> p2x >> p2y;
    cout << "P3: ";
    cin >> p3x >> p3y;
    cout << "P4: ";
    cin >> p4x >> p4y;
    cout << "P: ";
    cin >> px >> py;

    if(((px<p1x) || (px<p2x) || (px>p3x) || (px>p4x)) &&
       ((py<p1y) || (py<p2y) || (py>p3y) || (py>p4y)))
    {
        cout << "P lies outside." << endl;
    }
    if(((py>p1y) || (py<p2y) || (py<p3y) || (py>p4y)) &&
       ((px>p1x) || (px>p2x) || (px<p3x) || (px<p4x)))
    {
        cout << "P lies inside." << endl;
    }
    else
    {
        cout << "P lies on line." << endl;
    }
    return 0;
}
```

Output



```
enter points:
P1: 0 0
P2: 2 0
P3: 2 2
P4: 0 2
P: 1 1
P lies inside.
```

Description

This program takes 10 inputs from user, 8 inputs are of rectangle’s coordinates and rest 2 are point for which we have to find whether they are inside, outside or on the line.

Link: http://codepad.org/ycJ9hwF8

Program 12: Find Second Maximum Number(using operators &&, >, <)

Write a program which takes 3 inputs integers and tell the 2nd maximum.
Sample Input: 90 5 60
Sample Output: 60

Code

```
1    // This program display the 2nd maximum number from 3 numbers
2    #include <iostream>
3    using namespace std;
4
5    int main()
6    {
7       int num1, num2, num3;
8       cout << "enter 3 numbers: ";
9       cin >> num1 >> num2 >> num3;
10
11      if ((num1>num2 && num1<num3)||(num1<num2 && num1>num3))
12      {
13          cout << num1 << endl;
14      }
15      else if ((num2>num1 && num2<num3)||(num2<num1 && num2>num3))
16      {
17          cout << num2 << endl;
18      }
19      else
20      {
21          cout << num3 << endl;
22      }
23      return 0;
24   }
```

Output

```
enter 3 numbers: 90 5 60
60
```

Description
This program takes three input from user and tell the second maximum from three numbers. At line 11 **if** is used with condition(**(num1>num2 && num1<num3)||(num1<num2 && num1>num3)**), this condition is used if first entered number is second maximum. At line 15 **else if** is used with condition(**(num1>num2 && num1<num3)||(num1<num2 && num1>num3)**), this condition applies on second entered if it is second maximum. If both the conditions is false(means if number one or number two is not second maximum) then definetly else is executed display 3rd number.

Link: http://codepad.org/nrnEsaFL

Program 13: Find Ceiling Integer(using operators > ,+, &&, != and if else statement)

Write a program which takes as input a floating number and prints its ceiling Integer.

Sample Input: 5.5 Output: 6	Sample Input: -5.5 Output: -5	Sample Input: 5 Output: 5

Code

```
1   // This program display ceiling integer from floating number
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       float num;
8       int ceil;
9       cout << "Enter Number: ";
10      cin >> num;
11      ceil = num;
12
13      if (num>0 && ceil!=num)
14      {
15          cout << ceil+1;
16      }
17      else
18      {
19          cout << ceil;
20      }
21      return 0;
22  }
```

Output

```
Enter Number: 5.5
6
```

```
Enter Number: -5.5
-5
```

```
Enter Number: 5
5
```

Description

This program take input(number) from user and output its ceiling value. When the user enter number it is stored in **num** and that **num** is also assigned to **ceil** which is of integer type. At line 13 we just check if the **num** is greater than zero and **ceil** not equal to **num** then display **ceil** value by adding 1 in it other wise just display **ceil** value.

Link: http://codepad.org/sNt16ksm

Program 14: Find Floor Integer(using operators > ,+, &&, != and if else statement)

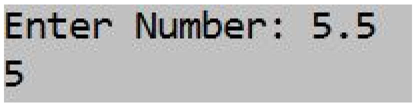
Write a program which takes as input a floating number and prints its floor value.

Sample Input: 5.5 Output: 5	Sample Input: -5.5 Output: -6	Sample Input: 5 Output: 5

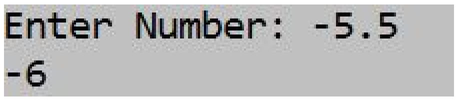
Code

```
1   // This program convert and display floating value to floor value
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       float num;
8       int floor;
9       cout << "Enter Number: ";
10      cin >> num;
11      floor = num;
12
13      if (num<0 && floor!=num)
14      {
15          cout << floor-1;
16      }
17      else
18      {
19          cout << floor;
20      }
21      return 0;
22  }
```

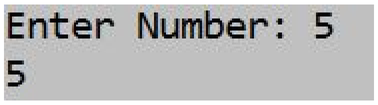
Output



```
Enter Number: 5.5
5
```



```
Enter Number: -5.5
-6
```



```
Enter Number: 5
5
```

Description

This program take input(number) from user and output its floor value. When the user enter number it is stored in **num** and that **num** is also assigned to **floor** which is of integer type. At line 13 we just check if the **num** is smaller than zero and **floor** not equal to **num** then display **floor** value by subtracting 1 in it other wise just display **floor** value.

Link: http://codepad.org/48wHs1FT

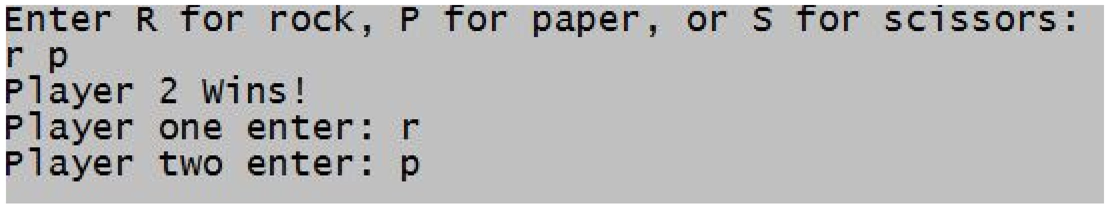
Mini Projects

Rock, Paper and Scissor game (using operators ==, &&, ||, if, else if and else statement)

Write and run a program that plays the game of “Rock, paper, scissors.” In this game, two players simultaneously say (or display a hand symbol representing) either “rock,” “paper,” or “scissors.” The winner is the one whose choice dominates the other. The rules are: paper dominates (wraps) rock, rock dominates (breaks) scissors, and scissors dominate (cut) paper. You can use r = rock, p = paper, s = scissors

```
#include <iostream>
using namespace std;
// Rock, Paper and Scissor game
int main()
{
   char player1_symbol;
   char player2_symbol;

   cout << "Enter R for rock, P for paper, or S for scissors:" << endl;
   cin >> player1_symbol >> player2_symbol;

   if ((player1_symbol == 's' && player2_symbol == 'p') ||
      (player1_symbol == 'r' && player2_symbol == 's') ||
      (player1_symbol == 'p' && player2_symbol == 'r'))
   {
       cout << "Player 1 Wins!" << endl;
   }
   else if ((player1_symbol == 'p' && player2_symbol == 's')||
           (player1_symbol == 's' && player2_symbol == 'r')||
          (player1_symbol == 'r' && player2_symbol == 'p'))
   {
      cout << "Player 2 Wins!" << endl;
   }
   else if ((player1_symbol == 'r' && player2_symbol == 'r')||
           (player1_symbol == 'p' && player2_symbol == 'p')||
          (player1_symbol == 's' && player2_symbol == 's'))
   {
      cout << "Tie!" << endl;
   }
   else
   {
      cout << "Invalid Input." << endl;
   }
   cout << "Player one enter: " << player1_symbol << endl;
   cout << "Player two enter: " << player2_symbol << endl;
   return 0;
}
```

Output



```
Enter R for rock, P for paper, or S for scissors:
r p
Player 2 Wins!
Player one enter: r
Player two enter: p
```

Description

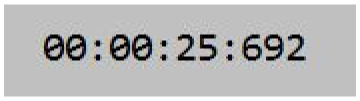
Link: http://codepad.org/4HioBgjF

StopWatch(using operators ++ , ==)

Code

```
1   //Stop-Watch
2   #include <iostream>
3   #include <iomanip>
4
5   using namespace std;
6
7   int main()
8   {
9      int hours=0,minutes=0,seconds=0,milliseconds=0;
10     cout <<"\n\n\n\n";
11
12     while(true)
13     {
14        cout << "\r\t\t\t\t"
15        << setw(2)<<setfill('0')<<hours<<":"
16        << setw(2)<<setfill('0')<<minutes<<":"
17        << setw(2)<<setfill('0')<<seconds<<":"
18        << setw(3)<<setfill('0')<<milliseconds;
19        milliseconds++;
20
21        if (milliseconds==1000)
22        {
23           milliseconds=0;
24           seconds++;
25        }
26        if (seconds==60)
27        {
28           seconds=0;
29           minutes++;
30        }
31        if (minutes==60)
32        {
33           minutes=0;
34           hours++;
35        }
36     }
37     return 0;
38  }
```

Output



```
00:00:25:692
```

Description

This program is stopwatch. As you see in code we are simply using **while** which is infinite. Four integer variables are declared and initialized with `0` at line 9. At line 3 another library named **<iomanip>** is included, in this library **setw** and **setfill** is defined. **Setw** is used to set the width of output operation as you see on line 15 **setw** with width 2 is used for hours because hours is not greater than 24, 24 are 2 digits that's why field set to 2. **Setfill** is used to initialize that output operation as you see in code it is initialized with zero mean stopwatch start from zero.

Syntax for **setw** and **setfill** is below

setw(width);
setfill('number for initialization');

You also see **\n \r \t** in code. **\n** means newline it is short form of **endl** but **\n** is written inside double quotes. **\t** is used for horizontal tab means several spaces is applied on output. **\r** is carriage return it tells the compiler to move the cursor at the start of the line.

Link: http://codepad.org/EBD6aq0n