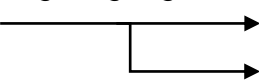
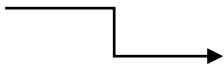
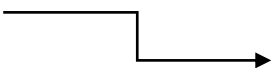


# C++ PROGRAMMING LANGUAGE

## DATA TYPES & THE OPERATORS

In most programming languages there are three types of data are included:

- 1- Numbers: 
  - a- Integer
  - b- Float
- 2- Characters: 
  - 1 character
- 3- Strings: 
  - Many of characters (word or statement)

While writing program in any language, you need to use various variables to store various information.

**Variables** are reserved locations in the memory to store values.

This means that when you create a variable you reserve some space in memory.

A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

### 1- Integer numbers:

Type	Typical Bit Width	Typical Range
int	4bytes	-2,147,483,648 to 2,147,483,647
Short	2bytes	-32768 to 32767
long	4bytes	-2,147,483,648 to 2,147,483,647

### Example of declaration

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    short b;
```

```
long c;  
a=5;  
b=6;  
c=7;  
cout <<"a="<<a<< endl;  
cout <<"b="<<b<< endl;  
cout <<"c="<<c<< endl;  
return 0;  
}
```

Output:

```
a=5  
b=6  
c=7
```

## 2- Floating numbers:

Type	Typical Bit Width	Typical Range
float	4bytes	+/- 3.4e +/- 38
double	8bytes	+/- 1.7e +/- 308
Long double	10bytes	+/- 3.4e +/- 4932

### Example of declaration

```
#include <iostream>  
using namespace std;  
int main()  
{  
    float a;  
    double b;  
    a=3.;  
    b=5.1;  
    cout <<"a="<<a<< endl;  
    cout <<"b="<<b<< endl;  
    return 0;  
}
```

Output

```
a=3  
b=5.1
```

### 3- Characters:

Type	Typical Bit Width	Typical Range
char	1bytes	-127 to 127 or 0 to 255

#### Example of declaration

```
#include <iostream>
using namespace std;
int main()
{
    char ch;
    ch='a';
    cout <<"ch="<<ch<< endl;
    return 0;
}
```

#### Output

ch=a

### 4- Boolean:

Type	Typical Bit Width	Typical Range
bool	1bytes	False or True (0 or 1)


#### Example of declaration

```
#include <iostream>
using namespace std;
int main()
{
    bool a,b;
    a=5>1;
    b=5<1;
    cout <<"a="<<a<< endl;
    cout <<"b="<<b<< endl;
    return 0;
}
```

#### Output

a=1  
b=0

**Note:** 1- sizeof operator returns the size of a variable. For example, sizeof(a), where 'a' is integer, and will return 4.

2- We can the floating number ( $a \cdot 10^n$ )   $ae+n$

### **Variable Names**

The names given to variables (and other program features) are called identifiers.

What are the rules for writing identifiers?

- 1- You can use only 1 character to naming a variable.
- 2- Identifiers can be as long as you like, but most compilers will only recognize the first few hundred characters.
- 3- You can use upper- and lowercase letters, and the digits from 1 to 9. You can also use the underscore (\_).
- 4- The first character must be a letter or underscore.
- 5- The compiler distinguishes between upper- and lowercase letters, so Var is not the same as var or VAR.
- 6- You can't use a C++ keyword as a variable name.

**A keyword** is a predefined word with a special meaning. int, class, if, and while are examples of keywords

### **THE OPERATORS**

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provide the following types of operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Increment and Decrement Operators.

#### **1- Arithmetic Operators**

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
*	Multiplies both operands
/	Divides numerator by denominator
%	Modulus Operator and remainder of after an integer division

#### **2- Relational Operators**

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to

<=	Less than or equal to
==	equal to
!=	Not equal to

### 3- Logical Operators

Operator	Description
&&	AND
	OR
!	NOT

### 4- Assignment Operators

Operator	Description
+=	A+=5 mean A=A+5
-=	A-=5 mean A=A-5
*=	A*=5 mean A=A*5
/=	A/=5 mean A=A/5
%=	A%=5 mean A=A%5

### 5- Increment and Decrement Operators.

Operator	Description
++	A++ mean A=A+1 postfix
	++A mean A=A+1 prefix
--	A-- mean A=A-1 postfix
	--A mean A=A-1 prefix

This example illustrates the difference between postfix and prefix:

```
#include <iostream>
using namespace std;
int main()
{
    int a=5,b;
    cout <<"a="<<a<< endl;
    ++a;
    cout <<"a="<<a<< endl;
    b=++a;
    cout <<"a="<<a<< endl;
    cout <<"b="<<b<< endl;
    return 0;
}
```

Output:

```
a=5
a=6
a=7
b=7
```

```
#include <iostream>
using namespace std;
int main()
{
    int a=5,b;
    cout <<"a="<<a<< endl;
    a++;
    cout <<"a="<<a<< endl;
    b=a++;
    cout <<"a="<<a<< endl;
    cout <<"b="<<b<< endl;
    return 0;
}
```

Output:

```
a=5
a=6
a=7
b=6
```

## Operators Precedence in C++

operator	Precedence
( )	highest
* / %	
+ -	lowest

**Note:** if the precedence is equal (such as \*, /), we must perform which locate on the left before the other.

**Examples** Find the result for following:

- 1-  $10/5*(2+4) = 10/5*6 = 2*6 = 12$
- 2-  $30/3/5 = 10/5 = 2$
- 3-  $2+(5 * (6+7) +3) = 2+(5*13+3) = 2+(65+3) = 2+68 = 70$
- 4-  $15/4=3$
- 5-  $3/6+(18+20) = 3/6+38 = 0+38 = 38$
- 6-  $3.0/6+(18+20) = 3.0/6+38 = 0.5+38 = 38.5$
- 7-  $27\%5+2 = 2+2 = 4$
- 8-  $15-14\%3 = 15-2=13$
- 9-  $15-14\%(-3) = 15-2=13$

**Examples** Write the following expression in c++ language:

- 1-  $\frac{x+y}{a+b} = (x+y)/(a+b)$
- 2-  $x + \frac{y}{a+b} = x+y/(a+b)$
- 3-  $a - f \frac{a-b}{x-y} = a-f*(a-b)/(x-y)$
- 4-  $\frac{a+b}{c(d+e)} = (a+b)/(c*(d+e))$
- 5-  $2 + 5(6 + \frac{x-y}{a-b}) = 2+5*(6+(x-y)/(a-b))$

**Example** Write a program to calculate the number of hours and minutes and seconds in a time=10000 seconds:

```
#include <iostream>
using namespace std;
int main()
{
    int hrs,mins,secs=10000;
    mins=secs/60; // 1 minutes = 60 seconds.
    secs=secs%60; // To find the rest of the number of seconds
    hrs=mins/60; // 1 hours = 60 minutes.
    mins=mins%60; // To find the rest of the number of minutes
    cout<<hrs<<":"<<mins<<":"<<secs<<endl;
    return 0;
}
```

**References:**

- Object-Oriented Programming in C++, Fourth Edition
- Tutorials Point <https://www.tutorialspoint.com/cplusplus/>