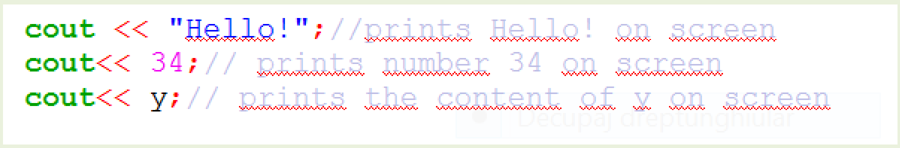
BASIC INPUT/OUTPUT

Using the standard input and output library, we will be able to interact with the user by printing messages on the screen and getting the user's input from the keyboard.

The standard C++ library includes the header file *iostream*, where the standard input and output stream objects are declared.

STANDARD OUTPUT (*cout*)

The standard output of a program is the screen, and C++ stream defined to acces it is *cout.*

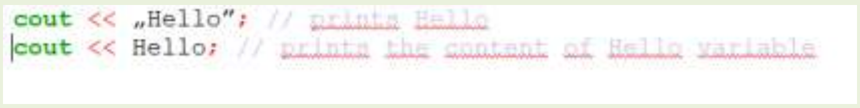


```
cout << "Hello!";//prints Hello! on screen
cout<< 34;// prints number 34 on screen
cout<< y;// prints the content of y on screen
```

The << operator insert the data that follows it into the stream preceding it.
In the examples above it inserted the constant string *Hello!* , the numerical constant 34 and variable y into the standard output stream <u>cout.</u>

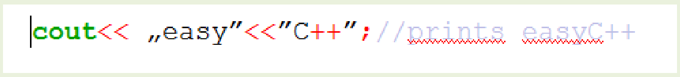
<u>Notice</u>: The sentence in the first instruction is enclosed between double quotes (“) because it is a constant string of caracters. Whenever we want to use constant strings of caracters we must enclose them between double quotes (”) so that thez can be clearly distinguished from variable names.

For example, these two sentences have verz different results:



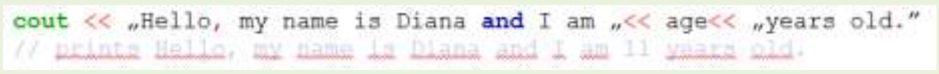
```
cout << „Hello”; // prints Hello
cout << Hello; // prints the content of Hello variable
```

The insert operator (<<) may be used more than once in a single statement:



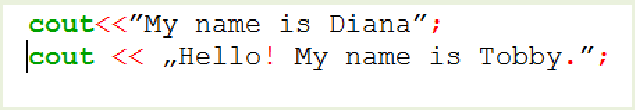
```
cout<< „easy”<<”C++”;//prints easyC++
```

The utility of repeating the insertion operator (<<) is demonstrated when we want to print out a comination of variables and constants of more than one variable; if we assume the age variable to contain the value 11:



```
cout << „Hello, my name is Diana and I am „<< age<< „years old.”
// prints Hello, my name is Diana and I am 11 years old.
```

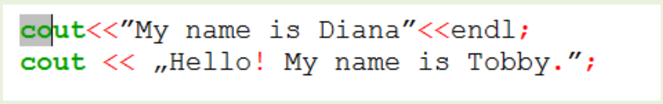
It is important to notice that cout does not add a line break after its output unless we explicity indicate it.



```
cout<<"My name is Diana";
cout << „Hello! My name is Tobby.";
```

will be shown on the screen: *My name is Diana. Hello! My name is Tobby.*

In order to perform a line break on the output we must explicity insert a new-line character into cout. In C++ a new-line character can be specified as *\n* or *endl*:



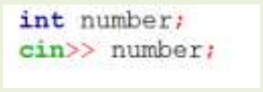
```
cout<<"My name is Diana"<<endl;
cout << „Hello! My name is Tobby.";
```

this produces the following output:
My name is Diana
Hello! My name is Tobby.

STANDARD INPUT (*cin*)

The standard input device is usually the keyboard. Handling the standard input in C++ is done by applying the overloaded operator of extraction (>>) on the cin stream. The operator must be followed by the variable that will store the data that is going to be extracted from the stream.

For example:

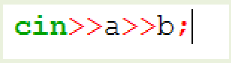


```
int number;
cin>> number;
```

The first statement declares a variable of type *int* called *number*, and the second one waits for an input from *cin* (the keyboard) in order to store it in this integer variable.

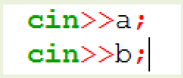
cin can only process the input from the keyboard once the *RETURN* key has been pressed.

You can also use *cin* to request more than one datum input from the user:



```
cin>>a>>b;
```

is equivalent to:



```
cin>>a;
cin>>b;
```

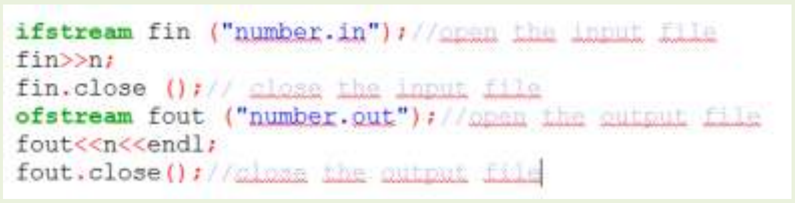
In both cases the user must give two data, one for variable *a* and another one for variable *b* that may be separated by any valid blank separator: a space, a tab character or a newline.

INPUT/OUTPUT WITH FILES

C++ provides the following classes to perform output and input of characters to/from files:

- **ofstream** : stream class to write on files;
- **ifstream** : stream class to read from files;
- **fstream** : stream class to both read and write from/to files.

These classes are derivated directly or indirectly from the classes *istream* and *ostream*.



```
ifstream fin ("number.in");//open the input file
fin>>n;
fin.close ();// close the input file
ofstream fout ("number.out");//open the output file
fout<<n<<endl;
fout.close();//close the output file
```