**Standard input.**

(cin)

In most program environments, the standard input by default is the keyboard, and the C++ stream object defined to access it is *cin*.

For formatted input operations, *cin* is used together with the extraction operator, which is written as >> (i.e., two "greater than" signs). This operator is then followed by the variable where the extracted data is stored. For example:

|  |  |  |
| --- | --- | --- |
| 1 2 | int age;  cin >> age; |  |

The first statement declares a variable of type int called *age*, and the second extracts from *cin* a value to be stored in it. This operation makes the program wait for input from *cin*; generally, this means that the program will wait for the user to enter some sequence with the keyboard. In this case, note that the characters introduced using the keyboard are only transmitted to the program when the ENTER (or RETURN) key is pressed. Once the statement with the extraction operation on *cin* is reached, the program will wait for as long as needed until some input is introduced.

The extraction operation on *cin* uses the type of the variable after the *>>* operator to determine how it interprets the characters read from the input; if it is an integer, the format expected is a series of digits, if a string a sequence of characters, etc.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | // i/o example  #include <iostream>  using namespace std;  int main ()  {  int i;  cout << "Please enter an integer value: ";  cin >> i;  cout << "The value you entered is " << i;  cout << " and its double is " << i\*2 << ".\n";  return 0;  } | Please enter an integer value: 702  The value you entered is 702 and its double is 1404. | [Edit & Run](https://cplusplus.com/doc/tutorial/basic_io/) |

As you can see, extracting from *cin* seems to make the task of getting input from the standard input pretty simple and straightforward. But this method also has a big drawback. What happens in the example above if the user enters something else that cannot be interpreted as an integer? Well, in this case, the extraction operation fails. And this, by default, lets the program continue without setting a value for variable *i*, producing undetermined results if the value of *i* is used later.

This is very poor program behavior. Most programs are expected to behave in an expected manner no matter what the user types, handling invalid values appropriately. Only very simple programs should rely on values extracted directly from *cin* without further checking. A little later we will see how *stringstreams* can be used to have better control over user input.

Extractions on *cin* can also be chained to request more than one datum in a single statement:

|  |  |  |
| --- | --- | --- |
|  | cin >> a >> b; |  |

This is equivalent to:

|  |  |  |
| --- | --- | --- |
| 1 2 | cin >> a;  cin >> b; |  |

In both cases, the user is expected to introduce two values, one for variable a, and another for variable b. Any kind of space is used to separate two consecutive input operations; this may either be a space, a tab, or a new-line character.