# Chapter 5. Input and output operators

## 5.1 Data input and output streams in C++

A stream is an abstract concept. It refers to any transfer of data from a source to a destination. In addition, a stream is a sequence of bytes that are independent of the specific device from which the data is read.

C++ input and output streams provide reliable handling of standard and user-defined data types. Streams have a uniform syntax. Exchange with the stream is carried out through a special area of computer memory – a buffer. Data transfer occurs on output (if the buffer is full) and on input (if the buffer is exhausted).

*Extraction* is the process of reading data from a stream.

*Putting (inclusion)* is the process of putting data into a stream.

To display a typed message on the screen or, conversely, enter text from the keyboard, a special library file iostream is used. The iostream library is part of the C++ standard library that handles basic input and output. To include this file you need to write #include at the very beginning of the program. Using the function of this library is necessary for entering data from the keyboard and outputting data to the console.

The iostream library defines two standard streams: cin and cout. Cin is the standard input stream. It communicates with the keyboard. The user writes data there. Cout is the standard output stream. It communicates with the screen. The data is output from the stream to the console. Since these streams are in the std namespace, it should be included. To do this, you need to write a separate line “using namespace std;”.

The iostream library provides two main classes for data input and output: istream and ostream. The istream class uses the stream extraction operation (the>>sign) for input. The ostream class uses the operation of placing onto a stream (the<< sign) for output. The operations of retrieving data from a stream and placing data into a stream are defined by the shift operator overload. And to go to the next line, you need to use the endl manipulator. It works with the standard cout stream. In addition, you can use the service character '\n'.

*Operator overloading* is changing the meaning of an operator. In this case, the simultaneous existence of several options for using the operator is possible. Stream input or output operators are already overloaded: when used with cin and cout objects, they fetch or insert data into the stream, and when used with int objects, they shift bits to the right or left.

## 5.2 Bitwise shift operator

C++ implements *bitwise shift operators* – the left shift operator (<<) and the right shift operator (>>). These operators cause the bits of the left operand to shift left or right by the number of positions specified in the second operand. Each integer in memory is represented as a certain number of digits. Shift operations apply only to integer operands [7].

Syntax: first write the name of the variable to which the shift is applied, then one of the two characters of the shift operator (<< or >>), and the number of bits by which the number is shifted (figure 5.1).

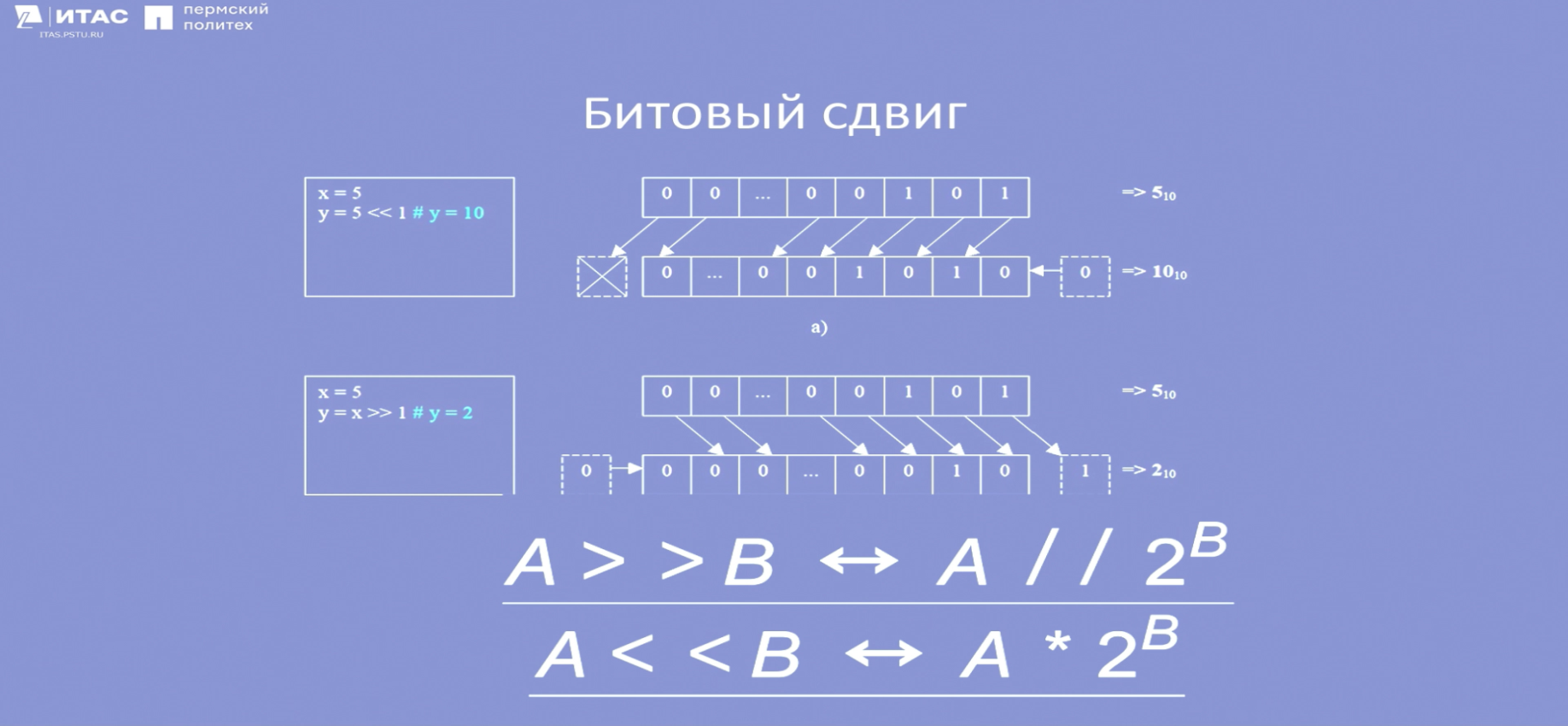


Figure 5.1 – Binary shift operation

An example of a bit shift operation is shown in figure 5.2. When the number 12 with the bit representation of 1100 is shifted to the left by 4 bits, the number with the bit representation 11000000 is obtained, which is equal to 192 in decimal notation. When a bit shift of the number 15, represented as 1111, to the right by 1, the result is a number with a bit representation of 111 – this is 7.

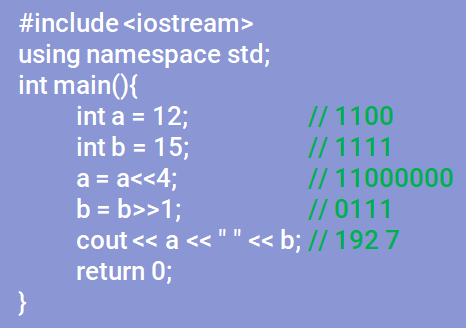


Figure 5.2 – An example of a binary shift

## 5.3 Practical work with the stream

Working with a stream is presented in more detail in figure 5.3. So, when you put the numbers 24 and 48 into the cout stream, they are output together in the console. To output numbers, strings, or information in general in different lines, you need to place control characters in the stream – '\n' or the word "endl".

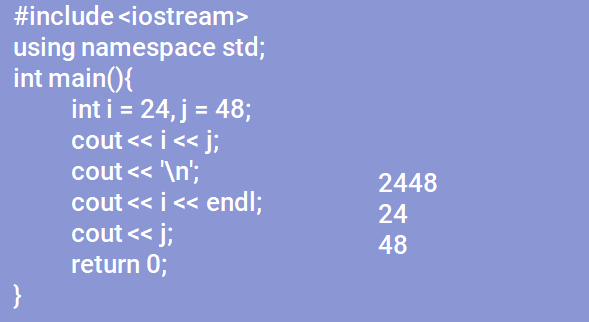


Figure 5.3 – Example of working with the cout output stream

Cin is the stream where the user writes data. In figure 5.4, data is retrieved from the cin stream and placed into the cout stream. So, the user typed the number 6, it was extracted, the variable is now equal to 6, and a line with a message and the value of the variable was written to the output stream [16].

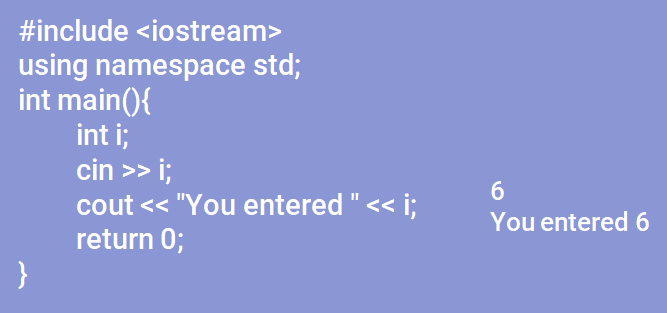


Figure 5.4 – An example of how data input and output streams work

In C++, it is possible to extract not just one variable from one stream, but several at once (figure 5.5). This requires the user to enter variables either by space or by entering, separating each variable by pressing the enter key. The type of entry in the console will depend on the selected input method (figure 5.6) [19]. Otherwise, these methods are similar, and therefore it does not matter how the user enters the data.

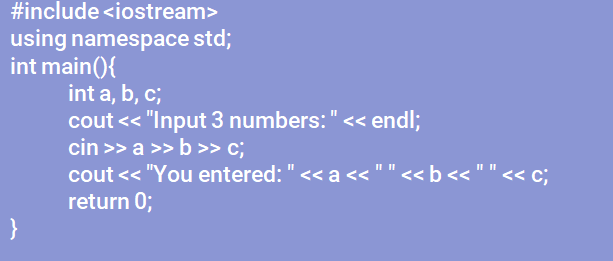


Figure 5.5 – Basic methods of entering data from the keyboard

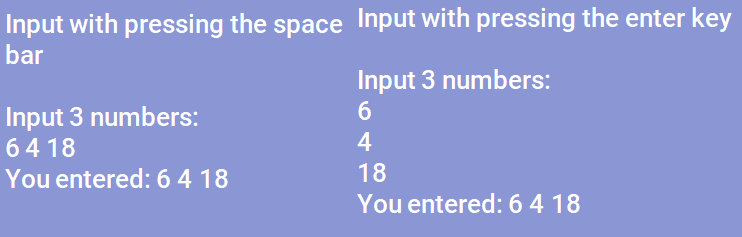


Figure 5.6 – Displaying data entered in different ways

In C++, operators have different priorities. If an expression uses multiple operators, high priority operations are executed before low priority operations. The put-to-stream operator has a lower priority than the addition operator, but the put-to-stream operator has a higher priority than the comparison and bit shift operators (figure 5.7).

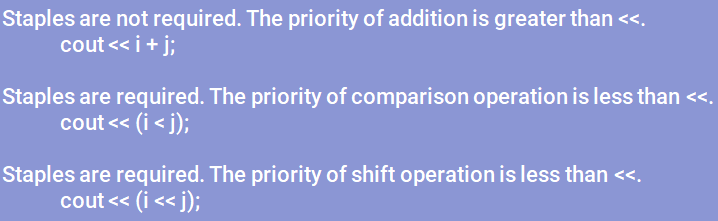


Figure 5.7 – Operator priorities

## 5.4 Localization of the program to the Cyrillic alphabet

If a program uses Cyrillic when outputting to the console, then the user may encounter a situation where other characters will be displayed instead of Cyrillic characters. The solution is to use the setlocale() function, inherited from the C library. To use this function, it is necessary that the Cyrillic alphabet be installed in the settings of languages and standards in the Windows OS. Or you can open a file in windows1251 encoding by writing the line: system(“chcp1251>NULL”); and connect the header file to the program. An example of entering the Cyrillic alphabet is shown in Figure 5.8.

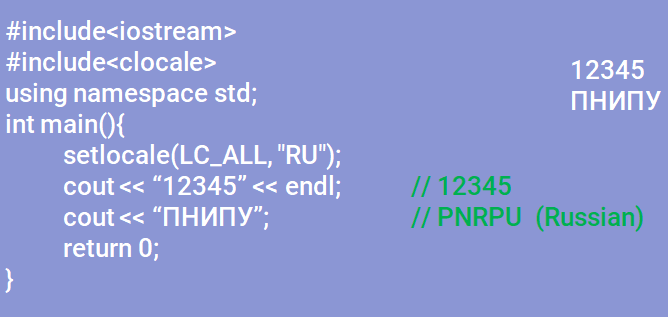


Figure 5.8 – Program localization

In the presented program code, localization was carried out through the setlocale() function, so the letters of the Russian alphabet were displayed on the screen.

To consolidate the material on input and output operators, you need to complete two practical tasks presented below. These tasks involve user input and output of information and variable values to the console.

1. Write a program to multiply two numbers and display the result. The numbers are specified by the user;

2. Write a fragment of a program in which five characters are entered, and then they are displayed on the screen in reverse order.