The Basics

Terms

Camel case
Comment
Constant
Directive
Expression
Hungarian notation
Mathematical expression
Operand
Operator
Pascal case
Snake case
Standard input stream
Standard output stream
Stream extraction operator
Stream insertion operator
Variable

Summary

- We use *variables* to temporarily store data in the computer's memory.
- To declare a variable, we should specify its type and give it a meaningful name.
- We should initialize variables before using them. Using an uninitialized variable can lead to unexpected behavior in our programs since these variables hold garbage values.
- Unlike variables, the value of *constants* don't change.
- The common naming conventions used in C++ applications are: **PascalCase**, **camelCase**, and **snake_case**.
- An *expression* is a piece of code that produces a value. A mathematical (arithmetic) expression consists of an operator (+, -, *, /, %) and two operands.
- Multiplication and division operators have a higher priority than addition and subtraction operators. So, they're applied first.

- We can use parentheses to change the order of operators.
- We use **cout** (pronounced sea-out) to write characters to the *Standard Output Stream* which represents the *terminal* or *console* window.
- We use **cin** (pronounced sea-in) to read data from the *Standard Input Stream* which represents the keyboard.
- We use the *Stream Insertion Operator* (<<) to write data to a stream.
- We use the *Stream Extraction operator* (>>) to read data from a stream.
- The *Standard Template Library* (STL) consists of several files each containing functions for different purposes.
- To use functions in the Standard Library, we should include the corresponding files using the #**include** *directive*.
- Using *comments* we can explain what cannot be expressed in code. This includes why's, how's, and any assumptions we made while writing code.

```
// Declaring a variable
int number = 1;

// Declaring a constant
const double pi = 3.14;

// Mathematical expressions
int x = 10 + 3;

// Writing to the console
cout << "x = " << x;

// Reading from the console
cin >> number;
```