C++ Programming Notes

*Based on w3schools*

**What is C++? (cpp)**

* Cross-platform language used to create high-performance applications. Developed by Bjarne Stroustrup as an extension to C.
* Gives high level of control over system resources and memory.

**Why use C++?**

* Found in today’s OS, GUI, and embedded systems.
* OOP language
* Adaptable to multiple platforms
* Similar to C, C#, and Java
* MAIN DIFFERENCE WITH C = Cpp supports classes and objects

**C++ Syntax**

To understand the structure of a Cpp program, we can use the Hello World example

|  |
| --- |
| #include <iostream> using namespace std;  int main() {   cout << "Hello World!";   return 0; } |

Breakdown of the code:

* Line 1: header file library, lets us work with I/O, add functionality to programs.
* Line 2: allows us to use names from the standard library.
* Line 4: main function of the program.
* Line 5: object used to print text.
* Line 6: ends the function.

Note: always end wih “;”

We can omit namespace by adding “**std::”** to the first line of the main function.

**C++ Output**

*cout* object with << operator is used to print text. Note that this doesn’t insert a new line at the end of the output.

* To insert a new line, use \n character – we can use “endl” the same way
* This is also called an **escape sequence**

A screenshot of a phone

Description automatically generated

**C++ Comments**

Used to explain the code and make it more readable. It can be used to prevent execution. Can be single lined (uses “//”, ignores text between // and end of line) or multi-lined (starts with “/\*” and ends with “\*/”, ignores text between start and end).

**C++ Variables**

Variables are containers for storing data values. There are different types of variables.

* int = stores integers (whole numbers), ex 123
* double = stores floating numbers (decimals), ex 19.9
* char = stores single characters, surrounded by single quotes, ex ‘a’
* string = stores text, surrounded by double quotes, ex “Hello World”
* bool = stores true or false

DECLARE VARIABLE

|  |
| --- |
| type variableName = value;  // example  int myNum = 15; |

* type = one of Cpp types
* variableName = name of variable
* equal sign = assign values to variable

It’s possible to declare a variable without assigning the value (assign later). You can overwrite the value to an existing variable as well.

|  |
| --- |
| int myNum; myNum = 15;  int myNum = 15;  // myNum is 15  myNum = 10;  // Now myNum is 10 |

Use cout to display variables.

|  |
| --- |
| cout << "I am " << myAge << " years old."; |

To declare many variables of the **same type**, use a comma-separated list. You can also assign the **same value** to multiple variables.

|  |
| --- |
| int x = 5, y = 6, z = 50;  int x, y, z;  x = y = z = 50; |

IDENTIFIERS

Variables must be identified with unique names or identifiers. It is recommended to use descriptive names to create understandable and maintainable code.

|  |
| --- |
| int minutesPerHour = 60; |

Rule of Thumb to naming variables:

* May contain letters, digits, and underscores.
* MUST begin with letters or underscoe.
* Names are case-sensitive.
* May NOT contain whitespace or special characters.
* Reserved Cpp words can’t be used as names.

CONSTANTS

To disallow the changing of variables, use *const* – which means unchangeable and read-only variable. Use for common, well-known values like pi.

|  |
| --- |
| **const** int myNum = 15;  // myNum will always be 15 |

* **Must** be assigned with a value

**C++ User Input**

Use *cin* to get input. This reads data from the keyboard with >> operator. Simple case:

|  |
| --- |
| int x;  cout << "Type a number: "; // Type a number and press enter  cin >> x; // Get user input from the keyboard  cout << "Your number is: " << x; // Display the input value |

**C++ Data Types**

In Cpp there are the following data types:

|  |
| --- |
| int myNum = 5;               // Integer (whole number)  float myFloatNum = 5.99;     // Floating point number  double myDoubleNum = 9.98;   // Floating point number  char myLetter = 'D';         // Character  bool myBoolean = true;       // Boolean  string myText = "Hello";     // String |

The data type specifies the size and type of information.

A screenshot of a computer

Description automatically generated

NUMERIC TYPES

Use *int* to store a whole number and *float* or *double* when you need decimals.

* Double vs. Float

The precision of a floating point indicates how many digits the value can have after the decimal point.

* + Float = 6-7 decimal digits
  + Double = ~ 15 digits (more commonly used)

SCIENTIFIC NUMBERS

Float can be a scientific number with *e* (exponent) to indicate powers of 10, ex:

|  |
| --- |
| float f1 = 35e3; |

BOOLEAN TYPES

Declared with *bool* and can only be true or false. When returned, true = 1 and false = 0.

* Used for conditional testing.

CHARACTER TYPES

Used to store a **single** character, must be surrounded by single quotes.

* You can use ASCII to display certain characters. Refer to [here](https://www.w3schools.com/charsets/ref_html_ascii.asp).

STRING TYPES

Used to store sequence of characters (text), must be surrounded by double quotes.

* You need to include an additional header file:

|  |
| --- |
| // Include the string library  #include <string> |

**C++ Operators**

Used to perform operations on variables and values.

ARITHMETIC OPERATORS

Used to perform common mathematical operations. Here’s a ist of operators:

A white and black text on a white background

Description automatically generated

ASSIGNMENT OPERATORS

Used to assign values to variables. Here’s a list of operators:

A table of math equations

Description automatically generated

COMPARISON OPERATORS

Used to compare 2 values or variables. Helps find answers and make decisions. The return value is either 1 or 0. Here’s a list of operators:

A screenshot of a phone

Description automatically generated

LOGICAL OPERATORS

You can test for true or false with logical operators. Used to determine the logic between variables and values.

A screenshot of a phone

Description automatically generated

**C++ Strings**

Used for storing text.

STRING CONCATENATION

The *+* operator can be used between strings to add them together to make a new string (**concatenation**).

|  |
| --- |
| string firstName = "John ";  string lastName = "Doe";  string fullName = firstName + lastName; |

* Add space using quotes (“ “ or ‘ ‘).

APPEND

A string is an object, which can contain functions. We can concatenate with the *append()* function.

|  |
| --- |
| string firstName = "John ";  string lastName = "Doe";  string fullName = firstName.append(lastName); |

ADDING NUMBERS AND STRINGS

* If you add 2 numbers, the result is a number.
* If you add 2 strings, the result is a concatenated string.
* If you add number and string, **an error occurs**.

STRING LENGTH

Use the *length()* function – you can also use *size()*.

|  |
| --- |
| string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  cout << "The length of the txt string is: " << txt.length();  cout << "The length of the txt string is: " << txt.size(); |

ACCESS STRINGS

Refer to its index number inside square brackets [ ].

* Index starts at 0, NOT 1.

|  |
| --- |
| string myString = "Hello";  cout << myString[0]; // Outputs H |

CHANGE STRING CHARACTERS

Access the string through index numbers and you can change the characters at that position.

|  |
| --- |
| string myString = "Hello";  myString[0] = 'J';  cout << myString; // Outputs Jello instead of Hello |

SPECIAL CHARACTERS

Use the **backslash** **escape** ( \ ) character to write quotes within a string.

A screenshot of a test results

Description automatically generated

|  |
| --- |
| string txt = "We are the so-called \"Vikings\" from the north."; |

USER INPUT STRINGS

It’s possible to use >> on *cin* to store a string entered by a user.

|  |
| --- |
| string firstName;  cout << "Type your first name: ";  cin >> firstName; // get user input from the keyboard  cout << "Your name is: " << firstName;  // Type your first name: John  // Your name is: John |

NOTE: *cin* considers a space (whitespace, tabs, etc) as a terminating character, so you can only input single words.

* To overcome this, we use *getline()* to read a line of text – *cin* is the first parameter, the string is the second parameter.

|  |
| --- |
| string fullName;  cout << "Type your full name: ";  getline (cin, fullName);  cout << "Your name is: " << fullName;  // Type your full name: John Doe  // Your name is: John Doe |

OMITTING NAMESPACE

The “using namespace std” line can be omitted and replaced with the *std* keyword followed by :: operator.

|  |
| --- |
| #include <iostream>  #include <string>  int main() {  **std::**string greeting = "Hello";  **std::**cout << greeting;    return 0;  } |