Ohlone College
CS 124 Programming with Data Structures: C++

Coding Style Guidelines

Some Style Issues in C++

• variable names	Use descriptive variable names. Do not use a single char.
• spacing	Use whitespace (blank lines and spaces) to create logical spacing. (e.g. Use a blank line after declarations; use a space around operators.)
• indentation	**Use 3 spaces for indentation. Do not use TABS.**
• documentation	Document your programs. Include a program header description as well as inline comments.
• printer friendliness	**Keep statement width < 80 characters.**

- **variable names**

The name of a variable should mean something. It should, at a glance, tell the reader for what the variable is being used. For example, if you are writing a function which measures the number of miles between two cities, you might use a variable named:

quality	**variable name**	**reasoning**
bad	**d**	Nondescriptive. *Maybe* the programmer will remember that **d** means "distance", but no one else will know.
poor	**dist**	Only a little better.
fair	**distance**	A number attached to this could be millimeters or parsecs.
good	**distance_in_miles**	Gives meaning to the number. Multiple words separated by "_" is a common practice.
good	**distanceInMiles**	Gives meaning to the number. Multiple words, each word but the first Capitalized (*Camel Case*) is a common practice.
quality	**Function/Method name**	**reasoning**
good	**setTime(); set_time(); isValidNumber()**	Void function names should describe a single **action** or **state-of being.** Other function names may **describe** the return value.

An exception to this might be variable names like **i**, **j**, or **k**, if these are being used as counters in a ***for*** loop. These are used so often this way that these names are immediately recognizable as loop counters.

Constants are usually written as all capital letters with '_' between words:

```
const float TAX_RATE = 0.095f;
```

Constants should be named for ***usage***, not ***value***:

(***i.e.*** **NOT** const float NINE_POINT_FIVE_PERCENT = 0.095f;)

Non-constant variables should **only** be declared within the scope of a function. "**Global**" non-const variable should be completely avoided.

As a separate matter, all primitive variables should be initialized.

- **spacing**

White space should be used wherever it makes the code more readable. Adding blank lines after the declarations and between logical "*chunks*" of code are often useful;

A single space on both sides of a **binary** operator is customary:

```
change_in_dollars = cost_in_dollars * dollars_tendered;
```

Exceptions to this are the **"dot" (.)** and **"target" (->)** operators, which customarily do not have surrounding spaces:

```
complexNumber.print();
complexNumberPointer->print();
```

Spaces should also be included after *commas (,)*

```
number = pow(baseNumber, exponent);
```

Indent statements within a logical code block:

```
{
    int myAge = 25;

}
```

Indent within a logical code block using a consistent width.

```
{
    int myAge = 25;
    int yourAge = 30;

    combinedAges = myAge + yourAge
}
```

- **indentation**

All code within a block should be indented and aligned. The most common practice is to place the "*curly braces*" on separate lines, aligned with each other. The same indentation rules hold if there are no "*curly braces.*" If a line of code is too long and needs to be continued on the next line, the continued line should also be indented.

A sample might be:

```
//-----------------------------------------------------------
// File: squares.cpp
//
// Functions:
//    main()
//-----------------------------------------------------------
#include <iostream>
#include <cstdlib>
using namespace std;
const int MAX_NUMBER = 6;
const int MAX_IN_LINE = 3;
//-----------------------------------------------------------
// Functions: main()
//
// Title: Square Writer
//
// Description:
// calculates and displays the squares of 1 - 6
//        as a double
//
// Programmer: Student Name
//
// Date: 2/2/2021
//
// Version: 1.0
//
// Environment: // Hardware: Intel Xeon PC
// Software: Windows 7
// Compiles under Microsoft Visual Studio 2010
//
// Output: squares of numbers 1 through 6 to console
//
// Parameters: void
//
// Returns: EXIT_SUCCESS
//
// History Log:
// 2/2/21 <student initials> completed version 1.0
//-----------------------------------------------------------
```

```
int main(void)
{
   int sum_of_squares = 0;

   for (int i = 0; i < MAX_NUMBER; ++i)
   {
      sum_of_squares += i * i;
      cout << "The sum of the squares up through"
           << i << " is "<< sum_of_squares << endl;
      if(i % MAX_IN_LINE == MAX_IN_LINE - 1)
         cout << endl;
   }

   return EXIT_SUCCESS;
}
```

- **documentation**

For our assignment submissions in this course, refer to the lab specifications for the style of headers.

Other comments should be sparse. Variable names should be descriptive enough that they need no further explanation.

Comments should only be added if they are needed to explain code that is obscure or unclear.

- **printer friendliness**

Code should be written so that individual lines of code should be short enough that when they are printed out, lines do not automatically "wrap", or are not cut off at the end. Lines that are too long should be broken at a logical place *(not inside of strings)* and made into multiple lines *(indenting the continued lines)*.