

Intro. to Computer Science/OOP [CSC-120] - Review Guide

[GitHub Repository](#)

Process of Writing Code

- Source Code: The code you're writing
- Pre-Processor: Including "codes" in your source code (i.e. `#include <iostream>`)
- Process: Source Code -> Pre-Processor -> Compiler -> Binary Code -> Linker -> Executable File

Elements in the Program

- Keywords/Reserved Words (i.e. int, double, float, char, string, bool) -> Comes with the language
- Programmer-Defined Identifiers (i.e. Variable Names)
- Operators (i.e. `<<`)
- Punctuations (i.e. comma, semi-colon)
- Syntax
- Main Function (Entry Point/Driver Function)

Errors

- Pre-Processor Error: Maybe file doesn't exist?
- Syntax Error: Compile Error
- Run-Time Error: Only happening when the code is running (i.e. dividing by 0)
- Logical Error: Code isn't running as expected

Input/Output

Cin & Cout

- Stands for "**Character Input/Output**"

```
1  #include <iostream>
2  using namespace std;
3
```

```
4  int main() {
5      int num1, num2;
6      cin >> num1 >> num2;
7      cout << num1 + num2 << endl;
8  }
```

Escape Character

IS A CHARACTER

EX:

- \n: New Line
- \t: Tab
- \b: Blank

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      cout << "Hello World!\n";
6      cout << "Hello World!" << endl;
7  }
```

Both will do the same thing, which is switching to a new line.

Reading & Writing Files

Use `#include <fstream>` or file access

- Input: ifstream
- Output: ofstream

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      //Declaring variables.
7      ifstream inFile;
8      ofstream outFile;
9  }
```

Opening Files:

```
1  #include <iostream>
```

```

2   using namespace std;
3
4   int main() {
5
6       ifstream inFile;
7       inFile.open("text.txt"); //File name can include the whole path.
8
9       //See if the file exists.
10      if (!inFile) {
11          cout << "File doesn't exist!" << endl;
12      }
13  }

```

Data Types

Integer (int)

- Between **about** -2B ~ 2B
- 4 Bytes (32 bits)
- By using **unsigned** -> all positive number -> double the "space"

Float (float)

- 4 Bytes (32 bits)
- Holds decimal places (not usually used)

Double (double)

- 8 Bytes (64 bits)
- Also holds decimal places like float, more commonly used

Character (char)

- 1 Byte (8 bits)
- Total Character Number: $2^7 = 128$
- ASCII Code: Each character has a matching number

Boolean (bool)

- 1 Byte (8 bits)
- Can only be **true** or **false**
- Things other than 0 will be false; otherwise, true

Operations

Modulus (%)

- Find the remainder of the division
- Cannot do with decimals

Division

```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5      cout << 13/5 << endl; //Prints 2
6
7      cout << 13/5.0 << endl; //Prints 2.6
8  }
```

Assignments & Initialization

Initialization

First time giving a value.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x = 10;
6  }
```

Assignment

Second+ time giving a value again.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x = 10; //Initialization
6
7      x = 20; //Assignment
8  }
```

Scope

- Main is a scope, where variables declared inside cannot be accessed outside the scope
- Scope of a variable: the part of the program in which the variable can be accessed

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int x; // Can only be recognized in main().
6  }

```

Condition

If-Else Statement

Asking questions

EX: If a variable is equaled to a certain value

The result of a condition is always true or false

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int income = 1000;
6
7      //Is x equaled to 100?
8
9      if (income > 1000) {
10         cout << "You are rich." << endl;
11     }
12     else {
13         cout << "You are poor." << endl;
14     }
15
16     //Going to print "You are poor." because income isn't greater than
17     1000.
18 }

```

Only If

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int income = 1000;
6
7      if (income > 1000) {
8          cout << "You are rich." << endl;
9      }

```

```

10
11      //Nothing will be printed because the if is passed.
12  }

```

Nested-If

If inside of an if statement (or more)

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int score = 100;
7
8      if (score >= 94) {
9          cout << "You get an A!\n";
10
11          if (score == 100) {
12              cout << "You also get a full!\n";
13          }
14      }
15  }

```

Relational Operators

Operators	Meaning
>	Greater Than
<	Less Than
>=	Greater Than & Equal To
<=	Less Than & Equal To
==	Equal To
!=	Not Equal To

Logical Operators

Operator	English	Meaning
&&	AND	New Relational Expressions will be true if both conditions are true. Otherwise, it's false.
	OR	New Relational Expressions will be true if either of the conditions is true. If both are false, it's false.

Operator	English	Meaning
!	NOT	Opposite of the expression.

Relational Expressions

Result of condition can be assigned as variable (boolean)

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      bool result;
6
7      int x = 10, y = 20;
8
9      result = x == y;
10
11     /*
12     x == y is a conditional statement, which will return either true
13     or false.
14     */
15
16     //In this case, result will be false since x != y.
17 }
```

Conditional Operators

Shortens the if-else statements

Format: expression ? expression : expression;

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int x = 10, y, z;
7
8      //If x > 0, y = 10. Else, z = 10.
9      x > 0 ? y = 10 : z = 10;
10
11     //Same code.
12
13     if (x > 0) {
14         y = 10;
15     }
16     else z = 10;
```

```
17
18     return 0;
19 }
```

Flags

Usually implemented as "bool"

- 0 = false
- non-0 = true

Loops

While Loop

- Executes "same code" for many times
- Not sure how many times the loop will run
- Statement will return true or false

For Loop

Knows **exactly** how many times to execute code

3 Expressions in For Loop

- Initialization --> Start with a number (counter)
- How many times? (Condition)
- Plus 1 (or other statements)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      //Prints "My name is Sean!" 10 times.
6      for (int i = 0; i<10; i++) {
7          cout << "My name is Sean!" << endl;
8      }
9
10     //Initialization: int i = 0;
11     //How many times? (Condition): i<10;
12     //Plus 1 (or other statements): i++
13 }
```

Do-While

Basically the same as while but **guarantee** at least 1 execution

Nested Loops

A nested loop is a loop inside the body of another loop

Inner (inside), outer (outside) loops:

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      //This loop will be executed 9 times.
7      for (int i = 0; i<3; i++) {
8          for (int j = 0; j<3; j++) {
9              cout << "HI\n";
10         }
11     }
12 }
```

Range Loop

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      string text = "text";
6
7      cout << "Characters in the string are: \n";
8      for (char ch : text) {
9          cout << ch << '\n';
10     }
11 }
```

Functions

EX:

```
1  void add() {
2      int a = 1, b = 2;
3      cout << a + b << endl;
4  }
```

- Functions that don't return a value are declared as "void"
- Functions that return values are declared whatever value they return
- Function parameters are considered local

Parts of Functions

1. Prototype (declaration of the function)
2. Definition: What does the function do?
3. Call of function

Array

List/Collection of data

Reserving the numbers of bytes needed for the array in RAM **continuously**

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      //5 grades --> 20 bytes reserved in memory
7      int grades[5];
8
9      //Index starts from 0 --> 0 Index
10     grades[0] = 100;
11     grades[1] = 90;
12     grades[2] = 80;
13     grades[3] = 70;
14     grades[4] = 60;
15
16     return 0;
17 }
```

- Global Array -> All elements initialized to 0
- Local Array -> Uninitialized
- Static Array -> Doesn't change size (in C++).

Vector

- Dynamic array -> Not static
- `#include <vector>` -> Doesn't come with C++, like string
- Using an array behind the scene

```
1  #include <iostream>
2  #include <vector> //Include the file
3  using namespace std;
4
5  int main() {
6      //Put whatever that's storing inside the vector inside < >.
```

```
7     vector<int> nums;
8
9     //To "add" elements into vectors, use "push_back()".
10    nums.push_back(1);
11
12    return 0;
13 }
```

Behind the Scene - Vector

- Initially creates an array of 10 elements
- Has a variable called capacity -> starts with 10
- Using the capacity variable, creates an array
- Has a size variable -> Keep track of how many elements inside vector -> Starts with 0
- Increases size by 1 with each push_back
- Once capacity is full -> Doubles capacity -> Creates new array with new capacity -> Copies elements from old array to new array