

1 Week 2

1.1 Cleaning up your code

Since when you write a program you are giving a sequence of instructions, the syntax is kind of like with the English language where we have sentences and paragraphs with indentation and periods to mark particular points. Each programming language will have syntax that it is particular about. The important syntax in C++ is curly braces and semicolons.

Consider our basic template from last week. You could take your code and write it

```
#include <iostream>
using namespace std;
int
main()
{
    cout
    << "Hello, World!\n"
    ;
    return
    0
    ;
}
```

or

```
#include <iostream>
using namespace std;
int main(){cout<<"Hello, World!\n";return 0;}
```

and the compiler will accept it. But it's rather ugly and hard to read. Here is an example of very readable code.

```
#include <iostream>

using namespace std;

/* This is my first code
   which will print 'Hello, World!' */
int main() {
    // cout means console out.
    cout << "Hello, World!\n";
    return 0;
}
```

Notice the following things:

- The indentation is consistent. Items 'nested' within curly braces have one more indentation.
- There's single-line comments for brief comments.

- There's multi-line comments for more detailed comments.
- Comments are short and to the point.
- The comments are consistently formatted. Comments in this example are always above the pertinent line.

Some of making code readable will be left up to you. For example, you could indent with a tab or indent with 4 spaces. The compiler doesn't require us to have these standards of readability. Making your code readable will mean you can share your ideas better. Having readable code will also mean you, future you, your TA, and instructor will be able follow your work.

This week you'll learn about variables. You can manipulate and change variables according to the rules of C++. The computer interprets such code as a bunch instructions to move information around or reserve places for it to live. You can name variables almost anything.

Variable names ...

1. must start with a letter or an underscore (`_`).
2. can contain letters, digits, and underscores.
3. cannot contain spaces or special characters other than underscores.
4. are case-sensitive.
5. cannot be the same as keywords (e.g. `int` is not a valid variable name).
6. should be descriptive and easy to understand.

The rules 1-5 satisfy the compiler. The last one is your choice to make your code more understandable. You might want to write a program that holds information like the age of the user. In your program, if you are going to store that, you could store it as `i`. But later on you might forget `i` was assigned an important value, try to use it, and then cause an error in your code. So you might want a more unique and descriptive variable name like `ageofuser`. Since you can't put spaces, programmers use two conventions to have their variable names be more readable: `camelCase` and `snake_case`. Here's an example of two variables that are each storing the age of a user.

```
int main() {
    int ageOfUser = 0;
    int age_of_user = 0;
}
```

We will go over more standards for readability when we get to writing and commenting functions and `hpp` / `cpp` arrangements.

1.2 Reading documentation

This week you'll begin to learn C++ syntax from new header files such as `cmath` and `string`. Part of learning new code will be reading the documentation provided by the programmers who created it. These websites include cplusplus.com and cppreference.com. The latter offers very complete information but cplusplus.com is more friendly for new users; although some formatting issues were introduced recently, unfortunately.

Let's take a look at the two pages for `cmath`:

<https://cplusplus.com/reference/cmath/>

- This shows the most commonly used functions (trig functions at the top, followed by hyperbolic functions, then exponential, etc.). It mentions data types but only at the very bottom.
- Once you can scroll past the ads, you can find the `abs` function near the bottom under ‘Other functions’. It says it computes the absolute value.
- One deficiency is that it only knows up to C++11, not C++14, C++17, or C++20.

<https://en.cppreference.com/w/cpp/header/cmath>

- This shows first the Types, then Macros (ignore this), then finally Functions. It includes all the functions from different versions of the same function.
- For example, the absolute value function `abs` can be found next to `fabs`, `fabsf`, and `fabsl`. These are other ways to take the absolute value but for more specific data types.
- In general, this one is more complete but has too much information for a beginner to sort through.
- There is one advertisement.

In summary, `cplusplus.com` has less information upfront, which makes it better for beginners to find what they need. The `cppreference.com` site is better organized, more complete, has less ads, and will work well in the future if you continue to want to master C++.

Let’s look at the `abs` function from `cplusplus.com`. Click on the link ‘abs’. You’ll find a page with this information:

```
double abs (double x);
float abs (float x);
long double abs (long double x);
```

This gives us three ways to use `cmath`’s function `abs`. (This may seem a bit strange. You’ll learn about overloading later!) You can see an example of `abs` working further down on the webpage. Copy the example into your own IDE and run it. Notice `abs` accepts a double and returns a double. Same with a float and long double. So `cmath`’s `abs` handles float-type inputs, but what if you give it an int?

Type `abs` in the website’s search and press Enter. You’ll be directed to another `abs` which lives in the header file `cstdlib`. This stands for C General Utility Library and is a something we don’t usually think about because its already included by XCode and Visual Studio when we add the header file `iostream`. When we do `abs` with an integer input, the compiler knows to use `cstdlib`’s function instead of `cmath`’s function. We will talk more about overloading functions in week 5.

The `cmath` header file has many useful external functions. Let’s look at another example of documentation of the `string` header file: <https://cplusplus.com/reference/string/string/>

One difference between this page and the page for `cmath` is we see there are ‘Member functions’. We’ll talk more about exactly what those are in week 6. For now think of these as tools we can use with our `string` variables.

1.3 Exercises

1.3.1 Practice coding exercises

1. Someone who is not your good friend shares with you their code with you.

```
#include <iostream>
using namespace std;int main(){double num1,num2;cout<<"Enter two doubles. "
<<endl;cout<<"I'll try to divide them. "<<endl;cin>>num2;cin>>num1;cout<<
num2<<"/"<<num1<<" = "<<num2/num1<<endl;return 0;}
```

It is messy but the code does function correctly. Tidy up their code by

- (a) Adding whitespace and linebreaks correctly.
 - (b) Naming variables sensibly.
 - (c) Add concise comments a user can follow.
2. The following code takes the input from the user as a string. Your task is to add code that will change the first and last vowels to uppercase.

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string str;
    //    cin >> str; // why not this line?
    getline(cin, str);

    // Add you're own code here.

    cout << str << "\n";

    return 0;
}
```

Think of the following cases:

- (a) If there is no vowel.
- (b) If there is just one vowel.

Test your code with a variety of test cases.

3. Given a dividend and divisor, find the quotient and remainder. Our goal is to make sure the remainder is positive.

- (a) Declare 4 ints called `dividend`, `divisor`, `quotient` and `remainder`. Supposing that these are appropriately named variables, write down an equation involving `==`, `*`, and `+` that should be true. Directly assign 13 to `dividend` and 4 to `divisor`, 3 to `quotient` and 1 to `remainder`. Check that `cout`ing the expression you wrote down prints true.
- (b) Instead of assigning 3 to `quotient` and 1 to `remainder`, assign appropriate expressions involving `dividend` and `divisor`. Check that your code still prints true and `cout` the value of the `remainder` (it should be 1).
- (c) Change 13 to -13. Depending on what you did in part (b), you might obtain a negative remainder. We'd prefer a positive remainder.
- (d) If necessary, add control flow between the formulae in step 2 and the `cout` statements to update `quotient` and `remainder` so that whatever values you assign to `dividend` and `divisor`, true is printed and the remainder is positive. To help with this step you might ask yourself, "when is the quotient and remainder incorrect?" and "in what way is it incorrect?".

Review: string functions, user input

Preview: logic, if-else

4. Write a program to check if an integer called `temperature` is higher than 70. If it is, it is too hot outside. If it's not, then it is nice outside. Have the program print appropriate statements for different `temperature` choice. Test your code. Write code to `cin` a user's entry.
5. Write a program that prompts a user for a username. After the user enters the username, it will take the username and capitalize the first letter if it's not capitalized.
6. Using control flow, write code to compute the minimum and maximum of two integers. Here's how it will start:

```
#include <iostream>
using namespace std;

int main() {
    int x, y;
    cout << "Please enter two numbers: ";
    cin >> x;
    cin >> y;

    return 0;
}
```

7. Write a program that takes in a numerical grade from a user (stored as a double) and then prints out the letter grade. Have it give the response "Invalid input" if the numerical grade is outside of 0-100.

1.3.2 Predict the output exercises

Predict the output for the follow code (it runs error-free).

8.

```
#include <cmath>
#include <iostream>

using namespace std;

/* If you're unsure about the behavior of a function.
   Check the documentation of cmath. */
int main() {
    cout << pow(2, 8) << endl;
    cout << sqrt(9) << endl;
    cout << round(2.5) << endl;
    cout << cos(0) << endl;

    return 0;
}
```
9.

```
#include <cmath>
#include <iostream>
using namespace std;

int main() {
    cout << static_cast<int>>(-3.4) << " " << static_cast<int>>(3.4) << "\n";
    cout << floor(-3.4) << " " << floor(3.4) << "\n";
    cout << ceil(-3.4) << " " << ceil(3.4) << "\n";
    cout << round(3.4) << " " << round(3.6) << "\n";
    cout << trunc(3.4) << " " << trunc(3.6) << "\n";

    return 0;
}
```
10.

```
#include <iostream>
using namespace std;

int main() {
    cout << boolalpha;
    double x = 20.15 - 20.0;
    double y = 0.15;
    bool b = (x - y == 0);
    cout << b << "\n";
}
```

```

    return 0;
}

```

Then how do you compare x and y?

```

11. #include <iostream>
    using namespace std;

    int main() {
        int    i = 10    * 100    * 1000    * 10000;
        double d = 10.0 * 100.0 * 1000.0 * 10000.0;

        cout << i << endl;
        cout << d << endl;
        cout << boolalpha;
        cout << (1.0 / (d * 100000.0 * 1000000.0) + 1.0 == 1.0) << endl;

        return 0;
    }

```

```

12. #include <iostream>
    #include <string>
    using namespace std;

    int main() {
        cout << boolalpha;
        cout << static_cast<bool>(8) << endl;
        cout << static_cast<bool>(0) << endl;

        string s = "AAARGH!!!";

        if (s.find("AAA")){cout << 1 << endl;}
        if (s.find("RGH")){cout << 2 << endl;}
        if (s.find("???")){cout << 3 << endl;}

        return 0;
    }

```

```

13. #include <iostream>
    #include <string>
    using namespace std;

```

```

int main() {
    string str1 = "hello!";
    string str2("hello");
    string str3(str2);
    string str4(6, 'h');
    string str5(str1, 2, 3);

    cout << "str1: " << str1 << "\n";
    cout << "str2: " << str2 << "\n";
    cout << "str3: " << str3 << "\n";
    cout << "str4: " << str4 << "\n";
    cout << "str5: " << str5 << "\n";

    return 0;
}

```

14. `#include <iostream>`
`#include <string>`

```
using namespace std;
```

```

int main() {
    string str1 = "hello";
    cout << "str1: " << str1 << "\n";
    str1.clear();
    cout << "str1: " << str1 << "\n";
    bool b = (str1 == "");
    cout << boolalpha;
    cout << b << "\n";

    str1 = "discussion";
    cout << "length of str1: " << str1.length() << "\n";

    return 0;
}

```

15. `#include <iostream>`
`#include <string>`

```
using namespace std;
```

```

int main() {
    string str1 = "nowayhome";

```

```

    char c1 = str1[0];
    char c2 = str1.at(0);
    char c3 = str1.front();

    cout << "c1: " << c1 << ", c2: " << c2 << ", c3: " << c3 << "\n";

    return 0;
}

```

16. `#include <iostream>`
`#include <string>`

```

using namespace std;

int main() {
    string str1 = "nowayhome";
    char c4 = str1[str1.length() - 1];
    char c5 = str1.at(str1.length() - 1);
    char c6 = str1.back();

    cout << "c4: " << c4 << ", c5: " << c5 << ", c6: " << c6 << "\n";

    return 0;
}

```

17. `#include <iostream>`
`#include <string>`

```

using namespace std;

int main() {
    string str1 = "happy ";
    string str2 = "birthday";

    string str3 = str1 + str2;
    cout << "str3: " << str3 << "\n";

    str1 += str2;
    cout << "str1: " << str1 << "\n";

    str1 = "happy ";
    str2 = "birthday!";
}

```

```

    str1.append(str2, 5, 3);
    cout << "str1: " << str1 << "\n";

    return 0;
}

```

18. `#include <iostream>`
`#include <string>`

`using namespace std;`

```

int main() {
    string str1 = "hello world";
    cout << str1.substr(1) << "\n";
    cout << str1.substr(2, 5) << "\n";

    str1.erase(2, 7);
    cout << "str1: " << str1 << "\n";

    str1 = "The Batman by Christopher Nolan";
    str1.replace(4, 6, "Dark Knight");
    cout << "str1: " << str1 << "\n";

    return 0;
}

```

19. `#include <iostream>`
`#include <string>`

`using namespace std;`

```

int main() {
    string str1 = "banana";

    size_t idx1 = str1.find("na");
    size_t idx2 = str1.rfind("na");

    cout << "idx1: " << idx1 << ", idx2: " << idx2 << "\n";

    idx1 = str1.find_first_of("na");
    idx2 = str1.find_last_of("na");

    cout << "idx1: " << idx1 << ", idx2: " << idx2 << "\n";
}

```

```

    return 0;
}

```

```

20. #include <iostream>
    using namespace std;

    int main() {
        cout << "\"hello\\\"\\\" << "\\n";

        return 0;
    }

```

```

21. #include <iostream>
    using namespace std;

    int main() {
        cout << ((21 % 4 != 0) && (10 / 100 == 0)) || (0 > 1) << endl;

        return 0;
    }

```

1.3.3 Error spotting exercises

Error spotting will be on the exam. Look at the follow code below and state:

- What errors occurred (build, run time, or logic) or if there are none.
- In the case of a logic error, what does the code currently accomplish?
- How the code can be fixed (use the line numbers to help with the description).

A description of how the code should perform will be put in a block comment. After you make your initial guess, type the code in your IDE and run it. Copying very carefully may help you find some of the errors. Play around with the code to fix it.

```

22. 1  #include <iostream>
    2  using namespace std;
    3
    4  /* The code assigns a the value 2, b the value 2 and prints
    5     the sum of a and b to the console. */
    6  int main() {
    7      a = 2;

```

```

8      b = 2;
9
10     cout << "2 + 2 = " << a + b << endl;
11     return 0;
12 }

```

23.

```

1  #include <iostream>
2  using namespace std;
3
4  /* Knowing that ch += 'A' - 'a'; would capitalize ch = 'g' to 'G'...
5     This code is meant to get ch to go from upper to lowercase. */
6  int main() {
7      char ch = 'G';
8      ch -= 'A' + 'a';
9      return 0;
10 }

```

24.

```

1  #include <iostream>
2  using namespace std;
3
4  /* Prints out the string "not a char". */
5  int main() {
6      cout << 'not a char' << endl;
7      return 0;
8  }

```

25.

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  /* The code is meant to extract the char at the fourth index. */
7  int main() {
8      string s = "My ice cream is cold.";
9      ch = s.at(4);
10     cout << ch << endl;
11
12     return 0;
13 }

```

26.

```

1  #include <iostream>
2  #include <string>

```

```

3
4  using namespace std;
5
6  /* The code is meant to extract the char at the fourth index. */
7  int main() {
8      string s = "My ice cream is cold.";
9      char ch = s[40];
10     cout << ch << endl;
11
12     return 0;
13 }

```

Try this on a computer with a different operating system (try with VS if you used XCode and vice versa). What are the implications if you use XCode and turn in a homework with an indexing error? How can you make XCode be more useful in evaluating your homework?

27.

```

1  #include <iostream>
2  using namespace std;
3
4  /* This code is meant to print 10000000000, 1e+10, false. */
5  int main() {
6      int    i = 10    * 100    * 1000    * 10000;
7      double d = 10.0 * 100.0 * 1000.0 * 10000.0;
8
9      cout << i << endl;
10     cout << d << endl;
11     cout << boolalpha;
12     cout << (1.0 / (d * 100000.0 * 1000000.0) + 1.0 == 1.0) << endl;
13
14     return 0;
15 }

```

28.

```

1  #include <iostream>
2  using namespace std;
3
4  /* This code is meant to get the char stored as ch to go
5     from lower to uppercase. */
6  int main() {
7      char ch = 'G';
8      ch += "A" - "a";
9      return 0;
10 }

```

```
29. 1  #include <iostream>
    2  using namespace std;
    3
    4  /* This code is meant to print: -1 */
    5  int main() {
    6      size_t idx = -1;
    7      cout << idx << "\n";
    8
    9      return 0;
   10 }
```