C++ Cheat Sheet

- **Getting input & outputting:**
  - cout << “This is output!” << endl; --> Will output: This is output!
  - To get input from the user, you have two options:
    - cin: Used directly under a cout statement asking for input, and stores the value the user enters into a variable:
      ```
      string a;
      cout << “Enter a name” << endl;
      cin >> a;
      ```
      This will enter what the user enters into a. Cin does not allow spaces in the entered value!
    - getline function: Allows you to take an entire entered string with spaces:
      ```
      string a;
      cout << “Enter a name” << endl;
      getline(cin, a);
      ```
      This will enter what the user enters into a, and will allow for there to be a space in the name
- **Functions:**
  - Allow you to perform tasks outside of main. There are three important factors to implementing functions that are essential to remember:
    - Function Declaration: Defines the return type, name, and parameters of the function. Put function definitions at the top of your code right under the libraries so that they are defined for the entire program to reference.
      Syntax: return_type function_name(parameters); <--semicolon needed!
      Ex) void playStory(int myName, int myStrength);
    - Function Implementations: These are the actual functions themselves, which hold the code that executes. The signature of the function implementation and declaration must match exactly(parameters and everything) or you will get an error.
      Syntax: return_type function_name(parameters) <--no semicolon!
      ```
      {
          //code written here
      }
      ```
      Ex) void playStory(int myName, int myStrength)
      ```
      {
          //code written here
      }
      ```
    - Function Calls: After your functions are implemented correctly, you can call them from main or other functions to execute the code written inside of them. Function calls have more simplistic syntax than declarations and implementations:

Syntax: function_name(parameters);
Ex) playStory(myName, myStrength);
Notice that the return type does not need to be written, and the types of the parameters don't need to be written either!

- **Arrays:**
  - Allow you to store multiple values of the same type in one place.
  - Creation: int arr[] = {1, 2, 3}; —> creates an integer array with the values 1, 2, and 3
  - Alternate creation: int arr[3]; —> creates an integer array with three spots, but it will be empty until you enter values
  - Putting user input into an array:

```
int length;
cout << "Enter the length for your array" << endl;
cin >> length;
int arr[length]; —>creates an array of size length
cout << "Enter the values for your array" << endl;
for(int i = 0; i < length; i++){
        cin >> arr[i];
}
```

  - Outputting the contents of an array:

```
cout << "Outputting your array" << endl;
for(int i = 0; i < length; i++){
        cout << arr[i];
}
```

- **Switch Statements:**
  - Allow you to make a more compact version of conditional statements, but with limitations. Switch statements can only be made on characters(ex: 'a') or integers(ex: 1).

Syntax:
```
int choice;
cout << "Choose an option: (1) or (2)" << endl;
cin >> choice;
switch(choice){
        case 1:
                //code
                break; —>NEED to have a break, or your switch statement
                           will run the cases under it until it reaches a
                           break!
        case 2:
                //code;
                Break;
        default:  —> the default will run if neither of the cases are run
                //code here optional
                break;
} //switch ends
```

- **Keep getting errors but can't find the source of the issue?** Go through this list to check for easy-to-miss problems that commonly cause errors:
  - Missing semicolons! A common culprit. If the line the error is saying is causing the problem isn't missing one, check the line above it.
  - Missing or extra curly brackets can cause a variety of issues, so if code isn't running correctly, or strange errors are occurring, check to make sure each curly bracket is part of a pair.
  - Undefined variable errors occur when you (1) haven't declared the variable in the function that uses it in or (2) the function doesn't have access to the variable. In the first case, make sure to define your variable at the start of your function (ex: int a;) but after declaring it, you don't need the 'int' signifier in front of the variable name(ex: a = 2; instead of int a = 2; if a is already defined). In the other case, make sure to pass the variable you're using to the function as a parameter, and add the variable to the function's parameter list in the declaration, implementation, and function calls. If you don't pass a variable to a function, it will have no idea what the variable is, or what value it has!
  - If the program isn't recognizing a type(like string) or function(like rand()) that you're trying to use, make sure that the library that contains the information for what you're using is included in your program file, or in a header file that your program file includes. In general, it's always good to start with including the <iostream> library and using namespace std; then building from there based on what you need.