Review day

Nicholas Matthews

Adam Kohl
What are we going to do?

- Review all the concepts that people aren’t too sure of
- Continue with any uncompleted assignments
- Tougher assignments for those who want to do them
Include Statements

- To specify what standard features we want we use `#include`
- Lets us call and use all the functionality in our own program without having to write the code ourselves
What Are Variables?

- Variables are containers for information.
- Different types of variables store different types of information:
  - bool, char, int, float, double, strings, arrays.

```cpp
// Programming Basics Cont.
bool aBoolVar;

aBoolVar = true;

bool anotherBoolVar = false;
```
User Command Line Input

- What if the user wants to assign a variable value?
- We use the cin stream

```
// Reading in a user defined value
float aFloat;    // Create a variable that holds type float

std::cout << "Enter a float value: " << std::endl;    // Prompt the user to enter a value
std::cin >> aFloat;    // Use cin to read in the value and assign it to the variable aFloat

std::cout << "The float entered is: " << aFloat << std::endl;    // Print the value back out to the user
```

Enter a float value:
3.5
The float entered is: 3.5
Press any key to continue ...
If statements

- Allows for changing code flow depending on conditions
- If (condition is true) { execute something } else {execute something else}
- Elseif

```cpp
if (bank_account < 0){
    cout << "Huh?" << endl;
}
elseif (bank_account > 1000000){
    cout << "WHAAAAAAAAAT?" << endl;
}
else{
    cout << "Welcome to the 99%" << endl;
}
```
For loop

- Convenience function that does a lot for us

for (initialize loop variable ; check condition is true ; increment loop variable) { execute something }

C++ takes care to execute everything properly and in order

```cpp
for (int count = 10; count > 0; count = count - 1){
    //Do something
    cout << count << endl;
}
```
Function definition

```c
int add_me_twice(int a){
    int b = a + a;
    return b;
}
```

- **Return value**: Returns only one thing at a time. Can be anything (int, char, double).
- **Function name**: Has to be a new name never defined before with the same parameters.
- **Input parameters**
FILE I/O – Writing to a file

```cpp
define fstream fpout("test.txt", ios::out);
fpout << "New Text" << endl;
fput << "Newer Text" << endl;
fput.close();
```

- **Variable name of the file handler**
- **File name**
- **Type of access Mode Flag**
- **Same way that we use cout can be used here**
FILE I/O – Reading from a file

- Pretty much the same concept
- Only some things are flipped

```cpp
#include <iostream>
#include <fstream>
#include <string>

int main()
{
    std::string str = "";
    std::ifstream fin; // ofstream
    fin.open("test.txt", std::ios::in);
    fin >> str;
    std::cout << str << std::endl;
    fin >> str;
    std::cout << str << std::endl;
    fin.close();
    return 0;
}
```
class Square  // Class keyword tells compiler to expect a class definition
{
    public:
        Square(float w){  // Class constructor
            width = w;
        }

        ~Square(){  // Class destructor
        }

        float area(){  // Area function definition
            return width*width;
        }

    protected:
        float width;  // Width variable used in the constructor and the area calculation

};

Square s = Square(5.0); // Instantiating a square object
std::cout << s.area() << std::endl; // Prints out 25.0
Challenge: Inheritance

```cpp
class Shape {
public:
    Shape() {
    }
    ~Shape() {
    }
    void setColor(std::string aColor) {
        color = aColor;
    }
    std::string returnColor() {
        return color;
    }
protected:
    std::string color;
};

class Square:public Shape {
public:
    Square(float w) { // Class constructor
        width = w;
    }
    ~Square() { // Class destructor
    }
    float area() { // Area member function definition
        return width*width;
    }

protected:
    float width; // Width is a memeber variable
};

Square s3 = Square(10.0); // Instantiating a square object
std::cout << s3.area() << std::endl; // Prints out 100.0
s3.setColor("blue"); // Inherited from shape
std::cout << s3.returnColor() << std::endl; // Returns blue
```
Let us split up our code into multiple files.

```cpp
#include "Rectangle.h"

class Rectangle {
public:
    Rectangle(); // The constructor prototype
    ~Rectangle(); // The destructor prototype
    void setArea(float area); // Setter
    float returnArea(); // Getter
    protected:
    float m_area; // Member variable
};

// main.cpp : Defines the entry point for the console application.

#include "Rectangle.h" // Copies all the rectangle code into this file
#include <iostream>

int main() {
    Rectangle aRec;
    aRec.setArea(45.0);
    std::cout << "Area: " << aRec.returnArea() << std::endl;
    return 0;
}
```
Assignment

- Make classes for rectangle, circle, triangle that inherit from shape
- Use the functions you have been working on in your classes
- Prompt the user to select a shape and to input values to calculate the area of the shape
- Challenge: Look up model, view, controller and structure your code that way