CSCI 1300 CS1: Starting Computing

Naidu, Correll, Yeh, Hoefer - Fall 2021

Recitation 10 - the week of Oct. 25

# Exercises

1. Spot the errors

Your group will compete against another group to find all the errors in this code. Fastest wins! Compare your solutions with your nemesis’ after you are done.

#include <iostream>

using namespace std;

class bank\_account {

private:

double balance;

public:

bank\_account(){

balance = 0;

}

bank\_account(double initial\_balance){

balance = initial\_balance;

}

void deposit(double amount){

balance += amount;

}

void withdraw(double amount){

balance -= amount;

}

double get\_balance() const{

return balance;

}

};

int main(){

bank\_account acct1(400);

// Print the current balance in acct1

cout << "Account 1 balance: " << acct1.balance << endl;

// Deposit $800 into acct1

balance = acct1.get\_balance;

balance += 800;

cout << acct1.get\_balance << endl;

bank\_account accts[20];

// Print the current balance of all the accounts in accts

cout << accts.get\_balance() << endl;

int sumAccts = 0;

for(int i = 0; i <= 20; i++)

{

sumAccts += accts.get\_balance[i];

}

cout << sumAccts << endl;

}

1. **Array of squares**

*Your goal is to implement (as a team) a buggy version of Square.h [7 minutes]. You will then swap your buggy implementation with another team and they have to correct your implementation of Square.h, then use it to implement Square.cpp [12 minutes]. Fastest team to execute the provided example output correctly wins!*

You want to write a computer graphics program which draws squares of different sizes and colors on the screen at some (x,y) location. First you’ll want to create an interface (square.h) and implementation (square.cpp), which consists of the attributes below.

Next you will want to create a squareDriver.cpp class which will create an array of 5 squares of different si-zes, locations, and colors. The size, location, and color can be user input or hard coded in main. Since we can’t actually draw on the screen yet, your squareDriver.cpp can just loop over the newly created array of square objects to call their print function!

| **Data members (private):** | |
| --- | --- |
| double: side | The length of the side of the square |
| string: color | The color of the square |
| double: x | The x position of the square |
| double: y | The y position of the square |
| **Member functions (public):** | |
| Default constructor | Sets side, x, y to 0 and color to empty string |
| Parameterized constructor | Takes a double side, x, y, and string color |
| getSide() | Returns side as a double |
| setSide(double) | (void) Assigns side the value of the input double |
| getX() | Returns x as a double |
| getY() | Returns y as a double |
| setCoords(double, double) | (void) Assigns x,y to the value of the input double |
| getColor() | Returns color as a string |
| setColor(string) | (void) Assigns color the value of the input string |
| getArea() | Returns the area as a double |
| print() | (void) Prints a string “color:<color>, size:<size>, position:<position>” |

Example output (**bold** is user input)

| Sample **main()** | Expected Output |
| --- | --- |
| Square s(10, 2, 5, “red”);  s.print();  cout << “Area: “ << s.getArea() << endl; | color: red, size: 10, position: (2,5)  Area: 100 |

Develop your solution in C++ by implementing header and .cpp files. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do.

Make sure you test for the values mentioned in the example output, and also test your code on sample test cases inclusive of boundary conditions.

| **Sample Run 1:** |
| --- |

| **Sample Run 2:** |
| --- |

| **Sample Run 3:** |
| --- |

| **Boundary Conditions:** |
| --- |

1. **List of Students in a class:** Create a Student class with the following attributes and methods. Create a Student array of size 30, and populate it with student details provided in students.txt. Write a function that filters Students based on their grade being in a range [low, high] as specified below.

*Function specifications:*

* The function name: **filterStudents**
* The function parameters (in this order):
  + Array of all students, Student
  + Array of filtered students, Student
  + Size of student arrays, int
  + The lowest grade in the filter range, double
  + The highest grade in the filter range, double
* The function should return the number of students in the filtered array, int

The Student class has the following attributes and methods:

| **Data Members (Private)** | |
| --- | --- |
| grade: double | The student’s current grade in the class. |
| name: string | The student’s name. |
| **Member Functions (Public)** | |
| Default Constructor | Set grade to 0 and string to an empty string. |
| Parameterized Constructor | Set the values of name, grade as provided to the constructor. |
|  | |
| getGrade() | Returns grade data member as a double |
| getName() | Returns name data member as a string |
| setGrade(double) | Set grade data member to the double argument that was passed. |
| setName(string) | Set name data member to the string argument that was passed. |
| print() | Formats and prints the contents of the Student object in the following format: name: grade%  *For example*: an object with name = “Louis” and grade = 95.5 will return the following string:  Louis: 95.5% |

Example output

| Sample **main()** | Expected Output |
| --- | --- |
| Student stud = Student("Kelly", 82.2); stud.print();    Student listStudents[30];  // Populate the array from students.txt here  Student listFilteredStudents[30]; double rangeLow = 93.0, rangeHigh = 100.0;  int numFilteredStudents =  filterStudents(  listStudents,  listFilteredStudents,  30,  rangeLow,  rangeHigh );  cout << numFilteredStudents  << " got an A grade." << endl;  for (int i=0; i<numfilteredStudents; i++) {  listFilteredStudents[i].print();  } | Kelly: 82.2%  2 students got an A grade.  Diana: 99.6%  Arthur: 97.3% |

Develop your solution in C++ by implementing header and .cpp files. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do.

Make sure you test for the values mentioned in the example output, and also test your code on sample test cases inclusive of boundary conditions.

| **Sample Run 1:** |
| --- |

| **Sample Run 2:** |
| --- |

| **Sample Run 3:** |
| --- |

| **Boundary Conditions:** |
| --- |