

Stony Brook University
College of Engineering and Applied Science

ESE 224.L02

Lab 1

Ryan Lin

Professor: Xing Wang

Report Date: 16 September 2023

Problem 1:

Main.cpp

```
int main()
{
    cout << "Creating a point with the default constructor: " << endl;
    Point p1;
    cout << "The point is (" << p1.getX() << ", " << p1.getY() << ")" << endl;

    cout << "Creating a point with the parameterized constructor: " << endl;
    cout << "Enter two coordinates (x, y): ";
    double x, y;
    cin >> x >> y;
    Point p2(x, y);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Change the x coordinate of p2, enter the new coordinate: ";
    cin >> x;
    p2.setX(x);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Change the y coordinate of p2, enter the new coordinate: ";
    cin >> y;
    p2.setY(y);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Compute the point's euclidean distance to origin (0, 0): " << endl;
    double dist = p2.dist2origin();
    cout << "Distance to origin is: " << dist << endl;
    system("pause");
    return 0;
}
```

Point.cpp

```
Point::Point()
{
    xCoord = 0;
    yCoord = 0;
}

Point::Point(double x, double y)
{
    xCoord = x;
    yCoord = y;
}

double Point::getX()
{
    return xCoord;
}

double Point::getY()
{
    return yCoord;
}

void Point::setX(double x)
{
    xCoord = x;
}

void Point::setY(double y)
{
    yCoord = y;
}

double Point::dist2origin()
{
    return sqrt(xCoord * xCoord + yCoord * yCoord);
}
```

Point.h

```
class Point
{
private:
    double xCoord, yCoord;

public:
    Point();
    Point(double x, double y);
    double getX();
    double getY();
    void setX(double x);
    void setY(double y);
    double dist2origin();
};
```

Problem 2:

Main.cpp

```
int main()
{
    Player player1;
    player1.setName("Christian Ronaldo");
    player1.setAge(38);
    player1.setHeight(6.2);
    player1.setNationality("Portugal");

    Player player2;
    player2.setName("Leonel Messi");
    player2.setAge(36);
    player2.setHeight(5.7);
    player2.setNationality("Argentina");

    cout << "Player Information:\n";
    cout << "Name: " << player1.getName() << endl;
    cout << "age: " << player1.getAge() << endl;
    cout << "Height: " << player1.getHeight() << endl;
    cout << "Nationality: " << player1.getNationality() << endl << endl;

    cout << "Player2 Information:\n";
    cout << "Name: " << player2.getName() << endl;
    cout << "age: " << player2.getAge() << endl;
    cout << "Height: " << player2.getHeight() << endl;
    cout << "Nationality: " << player2.getNationality() << endl;

    player2.prediction(player1);

    return 0;
}
```

Player.cpp

```

1  Player::Player(){
2      name = "";
3      age = 0;
4      height = 0.0;
5      nationality = "";
6  }
7  string Player::getName() const{
8      return name;
9  }
10 int Player::getAge() const{
11     return age;
12 }
13 double Player::getHeight() const{
14     return height;}
15 string Player::getNationality() const{
16     return nationality;
17 }
18 void Player::setName(const string& newName){
19     name = newName;
20 }
21 void Player::setAge(int newAge){
22     age = newAge;
23 }
24 void Player::setHeight(double newHeight){
25     height = newHeight;
26 }
27 void Player::setNationality(const string& newNationality){
28     nationality = newNationality;
29 }
30 void Player::prediction(Player p2){
31     if (age < p2.age && height > p2.height){
32         cout << "The winner is: " << name << endl;
33     }
34     else if (age > p2.age && height < p2.height){
35         cout << "The winner is: " << p2.name << endl;
36     }
37     else{
38         random_device rd;
39         default_random_engine generator(rd());
40         uniform_real_distribution<double> distribution(0.0, 1.0);
41         double random_number = distribution(generator);
42
43         //code to choose random player that wins
44         if (random_number < 0.5)
45         {
46             cout << "The winner is: " << name << endl;
47             // cout << random_number;
48         }
49         else
50         {
51             cout << "The winner is: " << p2.name << endl;
52             // cout << random_number;
53         }
54     }
55 }

```

Player.h

```
1  class Player
2  {
3  private:
4      string name;
5      int age;
6      double height;
7      string nationality;
8
9  public:
10     //constructor
11     Player();
12
13     //Getter Method
14     string getName() const;
15     int getAge() const;
16     double getHeight() const;
17     string getNationality() const;
18
19     //setter Method
20     void setName(const string& newName);
21     void setAge(int newAge);
22     void setHeight(double newHeight);
23     void setNationality(const string& newNationality);
24     void prediction(Player p2);
25
26 };
```

Problem 3:

Main.cpp

```
1  int main(){
2      Student std;
3      std.getDetails();
4      cout << endl;
5      std.displayDetails();
6      cout << endl;
7      Student std2;
8      std2.getDetails();
9      cout << endl;
10     std2.displayDetails();
11     cout << endl;
12     std2.comparison(std);
13
14     return 0;
15 }
```


Student.cpp

```
1  Student::Student(){
2      name = "";
3      roll_n = 0;
4      mark = 0;
5      percent = 0.0;
6  }
7
8  void Student::getPercent(){
9      int max_mark = 500;
10     if (mark == 0)
11     {
12         cout << "Warning: Mark is 0. Make sure to set a valid mark before calling getPercent." << endl;
13     }
14     percent = static_cast<double>(mark) / max_mark * 100.00;
15 }
16
17 void Student::comparison(Student s2){
18     if(mark > s2.mark){
19         cout << name << " is Rank one\n";
20     }
21     else{
22         cout << s2.name << " is Rank one\n";
23     }
24 }
25
26 void Student::getDetails(){
27     cout << "Enter name of student: \n";
28     cin >> name;
29     cout << "Enter roll number: \n";
30     cin >> roll_n;
31     cout << "Enter marks: \n";
32     cin >> mark;
33     getPercent();
34 }
35
36 void Student::displayDetails(){
37     cout << "Student information";
38     cout << "\nStudent name: " << name;
39     cout << "\nRoll Number: " << roll_n;
40     cout << "\nMark: " << mark;
41     cout << "\nPercent: " << percent << "%" << endl;
42 }
```

student.h

```
1  class Student
2  {
3  private:
4      string name;
5      int roll_n;
6      int mark;
7      double percent;
8  public:
9      //constructor
10     Student();
11
12     void getPercent();
13     void comparison(Student s2);
14
15     void getDetails();
16     void displayDetails();
17 };
```

Problem 4:

main.cpp

```

1  int main(int argc, const char * argv[]){
2      cout << "constructing 5 book objects" << endl;
3      Book b1(1,"b1", "b1a", "b1g", 1);
4      Book b2(2,"b2", "b2a", "b2g", 10);
5      Book b3(3,"b3", "b3a", "b3g", 11);
6      Book b4(4,"b4", "b4a", "b4g", 100);
7      Book b5(5,"b5", "b5a", "b5g", 101);
8
9      b1.display();
10     b2.display();
11     b3.display();
12     b4.display();
13     b5.display();
14     b5 = b1;
15     b5.display();
16
17     return 0;
18 }

```

book.cpp

```

1  Book::Book(int num, string title, string author, string genre, int page){
2      this->num = num;
3      this->title = title;
4      this->author = author;
5      this->genre = genre;
6      this->page = page;
7
8  }
9
10 void Book::display() {
11     cout << "Book Number: " << num << endl;
12     cout << "Title: " << title << endl;
13     cout << "Author: " << author << endl;
14     cout << "Genre: " << genre << endl;
15     cout << "Pages: " << page << endl << endl;
16 }

```

Book.h

```

1  class Book{
2  private:
3      int num;
4      string title;
5      string author;
6      string genre;
7      int page;
8  public:
9      Book(int num, string title, string author, string genre, int page);
10
11     void display();
12 };

```

Problem 5:

Main.cpp

```
1  int main(){
2      WaterLevelMonitor monitor;
3      cout << "Current water level:" << monitor.getCurrentLevel() << endl;
4      cout << "Average water level:" << monitor.getAverageLevel() << endl;
5      cout << "Highest water level:" << monitor.getHighestLevel() << endl;
6      cout << "Lowest water level:" << monitor.getLowestLevel() << endl;
7
8      double level;
9
10     for (int i = 0; i<10; i++)
11     {
12         cout << "input the " <<i+1<< "th data: ";
13         cin >> level;
14         monitor.receiveData(level);
15         cout << "Current water level:" << monitor.getCurrentLevel() << endl;
16         cout << "Average water level:" << monitor.getAverageLevel() << endl;
17         cout << "Highest water level:" << monitor.getHighestLevel() << endl;
18         cout << "Lowest water level:" << monitor.getLowestLevel() << endl <<endl;
19         cout << "count level: "<< monitor.getCountLevel() << endl;
20         cout << "sum level: "<< monitor.getSumLevel() << endl;
21     }
22     system("pause");
23     return 0;
24 }
```

WaterLevelMonitor.cpp

```
1  WaterLevelMonitor::WaterLevelMonitor() {
2      currentLevel = 0.0;
3      averageLevel = 0.0;
4      highestLevel = 0.0;
5      lowestLevel = -1.0;
6      countLevel = 0;
7      sumLevel = 0.0;
8  }
9  void WaterLevelMonitor::receiveData(double level) {
10     currentLevel = level;
11     countLevel++;
12     if (level > highestLevel) {
13         highestLevel = level;
14     }
15
16     if (level < lowestLevel) {
17         lowestLevel = level;
18     }
19     if (lowestLevel == -1.0)
20         lowestLevel = level;
21     sumLevel += level;
22     averageLevel = sumLevel / countLevel;
23     // cout << "count level: " << countLevel << endl;
24     // cout << "sum level: " << sumLevel << endl;
25 }
26 double WaterLevelMonitor::getCurrentLevel() const {
27     return currentLevel;
28 }
29 double WaterLevelMonitor::getAverageLevel() const {
30     return averageLevel;
31 }
32 double WaterLevelMonitor::getHighestLevel() const {
33     return highestLevel;
34 }
35 double WaterLevelMonitor::getLowestLevel() const {
36     return lowestLevel;
37 }
38 double WaterLevelMonitor::getCountLevel(){
39     return countLevel;
40 }
41 double WaterLevelMonitor::getSumLevel(){
42     return sumLevel;
43 }
```

WaterLevelMonitor.h

```
1  class WaterLevelMonitor
2  {
3  private:
4      double currentLevel;
5      double averageLevel;
6      double highestLevel;
7      double lowestLevel;
8      int countLevel;
9      double sumLevel;
10 public:
11     WaterLevelMonitor();
12
13     void receiveData(double level);
14     double getCurrentLevel() const;
15     double getAverageLevel() const;
16     double getHighestLevel() const;
17     double getLowestLevel() const;
18     double getCountLevel();
19     double getSumLevel();
20 };
```