VG101 Lab Worksheet

Lab 11

Instructor: Dr. Yifei ZHU
TA: Hangrui CAO
TA: Haoxuan SHAN
TA: Qinhang WU
TA: Muchen XU

Code Quality Requirement

- Ensure proper indentation
- Make naming meaningful
- Include necessary comments
- Split the code over functions
- Test the code as much as you can

Basic Syntax Exercise

C++ File IO

Write a program that copy the content in `in.txt` to `out.txt` line by line. You're required to use `fstream` in c++.

Note: don't forget to include the header file `fstream`.

C++ Memory Allocation

Write a program that allocate memory for

1. a variable
2. an array of 10 int
3. a 2-d array of 10*10 int
4. a 3-d array of 10*10*10 int

And then release the array properly.

C++ String Manipulation

Write a program that read a string and print all of its non-empty substrings by length. You may refer to `string::substr` for constructing a substring.

Input:

```
ABC
```

Output:

```
A
B
C
AB
BC
ABC
```

Simple Class

Define a `Rectangle` class representing a rectangle. The rectangle should have public attributes `length` and `width`, and public methods `int Rectangle::getArea()` and `int Rectangle::getPerimeter()`.

Start with

```cpp
class Rectangle
{
public:
    //TODO
};
```

Advanced IO

This exercise is intended to test the I/O of controlled indentation and precision using cout.

Suppose that a student has six features: name, student ID, exam scores for English, Math, and Physics, a total exam score. Print a table with `std::cout` to represent all of the students' information. Define your own data structure and specify the input by yourself.

Printing the result both to screen and to file (using `fstream`).

Restriction:

- The name should be 15 character long.
- The student ID should be 15 characters long.
- For three scores: precise to 3 decimal point, and 7 characters long, e.g., `100.000`.

Sample Output:

```
|Name           |Student ID     |English|Math   |Physics|Total  |
|Cao Hangrui    |51800000000    |82.540 |154.345|1.345  |238.230|
```

Simple String Exercise

A string s is a circular shift of a string t if it matches when the characters are circularly shifted by any number of positions. For example. `ACTGACG` is a circular shift of `TGACGAC`.

Given a string consisting of alphabets. First, print all the circular shift of the original string following the order of the characters in the original string. Next, sort these strings based on the first character in ascending order and print them correspondingly.

You're required to use `std::string` to complete this exercise.

Sample Input

```
ADAB
```

Sample Output

```
ADAB
DABA
ABAD
BADA

ABAD
ADAB
BADA
DABA
```

Hint

Use `std::sort` to do the sorting. Assume that `strArr` is a string array and `len` is the length of the array, then we may use `sort(strArr,strArr+len)` to sort it.

<!--## Simple String Exercise

Now we are going to write a program to detect whether the starting player can guarantte a win. The function prototype is:

```
bool canWin(string s);
```

If the starting player can guarantee a win output a 1, otherwise output a 0.

Sample Input:

```
++++
```

Output:

```
1
```

The first player can guarantee a win by turning the middle two `++` to `--`. Then the string will be `+--+` and his opponent can not flip a card any more. Thus the first player can guarantee his victory.-->

Simple C++ Class

In this part, we will design two geometric structures using C++ class.

Point

Design a `Point` class to represent a point in a 2D plane and implement some methods to play with different points.

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

```cpp

using namespace std;

class Point
{
public:
    double x,y; // Set varible type for x,y as double

    void setx(double _x)
    {
        //TODO: Set x as _x
    }
    void sety(double _y)
    {
        //TODO: Set y as _y
    }
    double module()
    {
        //TODO: Return the distance to the origin.
    }
    double distance(const Point &p)
    {
        //TODO: Return the distance between p and *this
    }
    void move(double dx, double dy)
    {
        //TODO: Move this point with dx in x-axis and dy in y-axis.
    }
    void print()
    {
        cout << fixed << setprecision(2);
        cout << "This Point is (" << x << "," << y << ")" << endl;
    }
    void neatPrint()
    {
        cout << fixed << setprecision(2);
        cout << "(" << x << "," << y << ")";
    }
};

int main()
{
    Point p1,p2;
    p1.setx(3);
    p1.sety(5);
    p1.print();
    // In print subfunction, set cout with reserving 2 digit after decimal
point

    p2.setx(6);
    p2.sety(9);
    cout << "P1's module = " << p1.module() << endl;
    // module equals the length to the origin point.

    cout << "P1 and P2 distance = " << p1.distance(p2) << endl;

    return 0;
}
```

Expected output:

```
This Point is (3.00,5.00)
P1's module = 5.83
P1 and P2 distance = 5.00
```

Segment

Take advantage of the `Point` class that we previously implemented to create the following `Segment` class.

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

class Point
{
    // You may simply copy from the previous problem
}

class Segment
{
public:
    Point A, B;
    void set(Point _A,Point _B)
    {
        //TODO: Set the segment with points _A and _B
    }
    double length()
    {
        //TODO: return the length of this segment
    }
    void move(double dx, double dy)
    {
        //TODO: move the segement by (dx,dy)
    }
    void print()
    {
        cout << fixed << setprecision(2) << "This Segment is from ";
        A.neatPrint();
        cout << " to ";
        B.neatPrint();
        cout << endl;
    }
};

int main()
{
    Point p1,p2;
    p1.setx(3);
    p1.sety(5);
    p2.setx(6);
    p2.sety(9);

    Segment seg;
    seg.set(p1, p2);
    seg.print();
    cout << "This segment has length = " << seg.length() << endl;
    seg.move(10, 10);
    seg.print();
    cout << "This segment has length = " << seg.length() << endl;

    return 0;
}
```

Expected output:

```
This Segment is from (3.00,5.00) to (6.00,9.00)
This segment has length = 5.00
This Segment is from (13.00,15.00) to (16.00,19.00)
This segment has length = 5.00
```

This example demostrates the hierachy relationship of different objects.

Simple Vector (C++, OOP version) (Optional)

In this part, we intend to implement a vector class in c++ style. You may refer to Lab 7 Worksheet for the usages of its member function in detail.

Hint:

- use `new` and `delete` to allocate and free the memory dynamically in c++

```
#include <iostream>

using namespace std;

class SimpleVector
{
    int size;
    int num;
    int *array;
public:
    void initArray(int _size = 0);
    // TODO:
    // Construct an array with specified size. Default set size as 0.

    void deleteArray();
    // Destructor. Delete the storage used by the array.

    void expand();
    // TODO:
    // This will be called by insert and set function when
    // the size of array is not enough. When size=0, set size of array to 1.
    // Otherwise, set size as size*2.

    void insert(int x);
    // TODO:
    // Append an element x in the array.
```

```cpp

    void set(int pos,const int &x);
    // TODO:
    // Set i-th element in the array as x. The index starts from 1.

    void printVec();
    // Print the information for the current vector;
};

void SimpleVector::printVec() {
    cout << "This vector has size=" << this->size;
    cout <<" and num=" << this->num << endl;
    for(int i = 0 ; i < num ; ++ i )
        cout << this->array[i] << " ";
    if(num > 0)
        cout << endl;
}

// The comments show the expected output
int main()
{
    SimpleVector vec1,vec2;

    vec1.initArray();
    vec1.printVec();
    // This vector has size=0 and num=0
    vec1.insert(1);
    vec1.insert(2);
    vec1.insert(3);
    vec1.printVec();
    vec1.deleteArray();
    // This vector has size=4 and num=3
    // 1 2 3

    vec2.initArray(3);
    vec2.printVec();
    // This vector has size=3 and num=0
    vec2.insert(1);
    vec2.insert(2);
    vec2.insert(3);
    vec2.set(2,0);
    vec2.set(2,5);
    vec2.printVec();
    // This vector has size=3 and num=3
    // 1 5 3
    vec2.set(7,5);
    vec2.printVec();
    vec2.deleteArray();
    // This vector has size=12 and num=7
    // 1 5 3 0 0 0 5

    return 0;
}

```

Some of the starter files are provided on Canvas.