C++ Basic Course

06/09/2018

Homework Assignment No. 5

(Total 60%)

1. **(30%)**: (a) Write an `IntegerVector` class for an array of integers. In the class, use `std::vector` as your internal data representation and provide the following constructors:

- `IntegerVector()` // print a message said "I am a default constructor."
- `IntegerVector(unsigned nelems)` // creates an `IntegerVector` with the integers `0...nelems-1`
- `IntegerVector(unsigned start, unsigned end)` // creates an `IntegerVector` with the range [start, end)

In addition, provide a non-member `print` function. Test your program with the client code listed below:

```
#include <iostream>
#include "IntegerVector.h"
using namespace std;

int main()
{
    IntegerVector a;
    IntegerVector b(5);
    print(cout, b);
    IntegerVector c(2, 5);
    print(cout, c);
}
```

```
I am a default constructor.
The elements in the IntegerVector are: 0 1 2 3 4
The elements in the IntegerVector are: 2 3 4
```

(b) Add an `intersection` member function that prints out the elements common to two arrays. Allow a sequential operation. Test your program with the client code listed below:

```
#include <iostream>
#include "IntegerVector.h"
using namespace std;

int main()
{
    IntegerVector a(3);
    IntegerVector b(5);
```

```
    IntegerVector c(2, 6);
    print(cout, c.intersection(b).intersection(a));
}
```

Your output looks like:

```
Intersection elements are: 2 3 4
Intersection elements are: 2
The elements in the IntegerVector are: 2
```

2. **(30%)**: Rational numbers (fractions) are numbers that can be written in the form *a*/*b*, where *a* and *b* are integers and *b* != 0. *a* is known as the *numerator* and *b* the *denominator*. Let the default value for *a* be 0 and for *b* be 1. Implement a class `Fraction` to represent rational numbers and allow their objects to support the following client code:

```
#include <iostream>
#include "Fraction.h"

using namespace std;

int main(){
    Fraction f1; // 0/1
    f1.setName("f1");
    cout << "======" << endl;
    printFraction(cout, f1);
    Fraction f2(3); //3/1
    f2.setName("f");
    Fraction f3(-2, 4); //-2/4
    cout << "======" << endl;
    printFraction(printFraction(cout, f2), f3);
    cout << "======" << endl;
    Fraction f4(cin); // prompt for input
    cout << "======" << endl;
    printFraction(cout, f4);
    cout << "======" << endl;
    printFraction(cout, f4.setName(&f2));
    return 0;
}
```

Below is a sample run:

```
======
Fraction f1: 0/1
======
Fraction f: 3/1
Fraction anonymous: -2/4
======
Enter the name for Fraction: F4
Enter the values for numerator and denominator: 5 3
======
Fraction F4: 5/3
======
Fraction f: 5/3
請按任意鍵繼續 . . .
```

<u>HW Grading Policy:</u>

1. You should consider about exception handling, e.g. error input, file opening fail, etc.
   請注意所有例外狀況的處理，例如：錯誤的符號字串輸入、檔案開啟失敗等。
2. The coding style includes your output format.
   輸出資料的格式將納入格式評分。
3. If your code is not compilable, your score in this problem is zero (including coding style).
   若程式無法編譯，則該題以零分計算。(包含格式分數)
4. Your program will be tested with other data which is not the same as provided samples.
   除了題目所提供的範例測試資料以外，作業程式碼將以額外的測試資料進行測試。

- Coding Style (20%): 編碼格式分數
  1. format
     整體形式與輸出資料的格式
  2. comments
     註解
  3. readability
     可讀性
  4. variables naming
     變數命名方式
  5. typesetting
     型別設定

- Functionality (80%): 功能性分數
  1. run-time performance:
     執行時的表現
     1) samples not passed -> x
        範例測資錯誤 => 此部分零分
     2) samples passed but some tests failed -> partial

範例測資通過但是部分測資失敗 => 部份給分

3) samples and tests all passed

範例測資與所有測資通過 => 此部分滿分

3. excellent method++

綜合以上，又以能展現解決問題的巧思尤佳。