Object-Oriented Programming Language

11/01/2018

Homework Assignment No. 6

Due 11:59 am, Wednesday November 7, 2018

Late submission within 24 hours: score*0.9;

Late submission within one week: score*0.8.

The solutions will be posted after one week of the due date.

(Total 100%)

Please see [submission details] in the back of this document, or you will not be graded for this assignment.

1. **(60%)** `new` and `delete` in C++ allow us to allocate and free a dynamic array and we can use these features to design a class `PFArrayD` for a "partially" filled double array to show like that in `std::vector`:
   1. capacity, and
   2. Currently-in-use

   Use the following client code, you CANNOT change, to test your implementation:

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

using namespace std;

void testPFArrayD(); //Conducts one test of the class PFArrayD.

int main(){
    cout << "This program tests the class PFArrayD.\n";

    char ans;
    do {
        testPFArrayD();
        cout << "Test again? (y/n) ";
        cin >> ans;
    } while ((ans == 'y') || (ans == 'Y'));

    return 0;
}

void testPFArrayD()
{
    int cap;
    cout << "Enter capacity of this partially filled array: ";
    cin >> cap;
    PFArrayD pfa1(cap);
    cout << "Capacity for pfa1: " << pfa1.getCapacity() << endl;
    cout << "Elements used in pfa1: " << pfa1.getNumberUsed() <<
endl;
```

```
    cout << "Enter up to " << cap << " nonnegative numbers.\n";
    cout << "Place a negative number at the end.\n";

    double next;
    cin >> next;
    while ((next >= 0) && (!pfa1.full())){
        pfa1.addElement(next);
        cin >> next;
    }

    cout << "Capacity for pfa1: " << pfa1.getCapacity() << endl;
    cout << "Elements used in pfa1: " << pfa1.getNumberUsed() <<
endl;
    print(cout, pfa1);

    pfa1.emptyArray();
    cout << "Capacity for pfa1: " << pfa1.getCapacity() << endl;
    cout << "Elements used in pfa1: " << pfa1.getNumberUsed() <<
endl;

}
```

Below is a sample run:

n

```
This program tests the class PFArrayD.
Enter capacity of this partially filled array: 30
PFArrayD(unsigned)
Allocate 30 doubles
Capacity for pfa1: 30
Elements used in pfa1: 0
Enter up to 30 nonnegative numbers.
Place a negative number at the end.
2.1 0 1.1 3 6 -1
Capacity for pfa1: 30
Elements used in pfa1: 5
Elements in array: 2.1 0 1.1 3 6
Capacity for pfa1: 30
Elements used in pfa1: 0
~PFArrayD()
Release 30 doubles
Test again? (y/n) n
```

2. **(40%**, You will need to finish Problem 1 first) Write the proper 1. copy constructor, 2. assignment operator and 3. destructor for class `PFArrayD`. Write messages to indicate memory allocation, release and reallocation when you call constructor, copy constructor, assignment operator and destructor. Use the following client code to test your implementation:

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

```
using namespace std;
int main(){
    PFArrayD pfa1;
    pfa1.addElement(1.0);
    pfa1.addElement(2.0);
    cout << "Capacity for pfa1: " << pfa1.getCapacity() << endl;
    cout << "Elements used in pfa1: " << pfa1.getNumberUsed() <<
endl;
    PFArrayD pfa2(30);
    pfa2.addElement(3.0);
    cout << "Capacity for pfa2: " << pfa2.getCapacity() << endl;
    cout << "Elements used in pfa2: " << pfa2.getNumberUsed() <<
endl;
    PFArrayD pfa3 = pfa2;
    cout << "Capacity for pfa3: " << pfa3.getCapacity() << endl;
    cout << "Elements used in pfa3: " << pfa3.getNumberUsed() <<
endl;
    pfa3 = pfa1;
    cout << "Capacity for pfa3: " << pfa3.getCapacity() << endl;
    cout << "Elements used in pfa3: " << pfa3.getNumberUsed() <<
endl;
    print(cout, pfa3);
    return 0;
}
```

```
PFArrayD()
Allocate 50 doubles
Capacity for pfa1: 50
Elements used in pfa1: 2
PFArrayD(unsigned)
Allocate 30 doubles
Capacity for pfa2: 30
Elements used in pfa2: 1
PFArrayD(const PFArrayD&)
Allocate 30 doubles
Capacity for pfa3: 30
Elements used in pfa3: 1
operator = (const PFArrayD&)
Release 30 doubles
Allocate 50 doubles
Capacity for pfa3: 50
Elements used in pfa3: 2
Elements in array: 1 2
~PFArrayD()
Release 50 doubles
~PFArrayD()
Release 30 doubles
~PFArrayD()
Release 50 doubles
```

[submission details]

For your homework submissions, please name the folder by your student ID, and archive the whole folder into a .zip file (DO NOT use .rar or other format).

For example, the folder name is r07521503-HW4, and the file you upload is r07521503-HW4.zip.

For arrangement of files and folder structure, we require you to put the files of a single problem in a folder, and name the main file by the homework number and problem number.

For example:

```
r07521503-HW4/
  hw4_1/
    hw4_1.cpp
    Person.h
    Person.cpp
    PersonVector.h
      PersonVector.cpp
  hw4_2/
    hw4_2.cpp
    Rectangle.h
    Rectangle.cpp
    Square.h
    Square.cpp
```

Another Example (Please follow the rule if you have to upload any lab.)

```
r07521503-Lab4/
  lab4_1/
    lab4_1a.cpp
    lab4_1b.cpp
  lab4_2/
    lab4_2a.cpp
    lab4_2b.cpp
  lab4_3/
    lab4_3.cpp
```