Part C – Encapsulation

Class with a Resource

Workshop 6

In this workshop, you are to design and code a class that accesses a resource.

Learning Outcomes

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- define a copy constructor;
- define an assignment operator;
- define a destructor;
- avoid duplication in coding these special member functions;
- describe to your instructor what you have learned in completing this workshop.

Submission Policy

The *"in-lab"* section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the *"in-lab"* portion of the workshop during that period, ask your instructor for permission to complete the *"in-lab"* portion after the period. If you do not attend the workshop, you can submit the *"in-lab"* section along with your *"at-home"* section (with a penalty; see below). The *"at-home"* portion of the lab is **due on the day of your next scheduled workshop** (at 23:59).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to regularly back up your work.

Late submission penalties:

- *"in-lab"* submitted late, with *"at-home"* submitted on time: maximum of 20/50 for *"in-lab"* and maximum of 50/50 for "*at-home*"*;*

- *"in-lab"* on time, with *"at-home"* and *"reflection"* submitted at most one week late: maximum of 50/50 for *"in-lab"* and maximum of 20/50 for *"at-home"*;
- workshop late for at most one week: *"in-lab"*, *"at-home"* **and** *"reflection"* must all be submitted for maximum of 50/100;
- workshop late for more than one week: *"in-lab"*, *"at-home"* **and** *"reflection"* must all be submitted for maximum of 30/100;
- if any of *"in-lab"*, *"at-home"* or *"reflection"* is missing, the mark will be **zero**.

IN-LAB (50%):

Design a class named `Contact`, in the namespace `communication`. This class holds information about a person and his/her phone numbers. The type `Contact` should contain the following members (add those that are missing in the provided files):

`m_name`: an array of characters of size 21 (including `'\0'`) that holds the name of the contact;

`m_phoneNumbers`: a **dynamically allocated** array of phone numbers (a phone number is of type `long long`). A valid phone number has one to two digits for the country code, **exactly** three digits for the area code (cannot start with zero) and **exactly** seven digits for the number (cannot start with zero);

`m_noOfPhoneNumbers`: a number that represents how many phone numbers a contact has (this is the size of the `m_phoneNumbers` array);

`bool isEmpty() const`: a query that checks if the current instance is in the **safe empty state**;

default constructor (a constructor with no parameters): this constructor should set the instance to the safe empty state;

constructor with 3 parameters: The first parameter contains the name of the contact, the second parameter is an array of phone numbers and the third parameter contains the number of phone numbers in the array. This constructor should allocate enough memory to the `m_phoneNumbers` attribute to hold **only the valid** phone numbers from the second parameter; then it should copy the valid phone numbers from the second parameter into the `m_phoneNumbers` array;

destructor: the destructor should make sure there are no memory leaks when an instance of type `Contact` goes out of scope;

`void display() const`: a query that prints the content of an instance:

- If the instance is in the safe empty state, this function should display the following message:

  ```
  Empty contact!<ENDL>
  ```

- If the instance is not in the empty state, this function should print the following message:

  ```
  CONTACT_NAME<ENDL>
  (+COUNTRY_CODE) AREA_CODE NNN-NNNN<ENDL>
  (+COUNTRY_CODE) AREA_CODE NNN-NNNN<ENDL>
  •••
  (+COUNTRY_CODE) AREA_CODE NNN-NNNN<ENDL>
  ```

Using the sample implementation of the `w6_in_lab.cpp` main module shown below, test your code and make sure that it works. Below the source code is the expected output from your program. The output of your program should match **exactly** the expected one.

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

using namespace std;
using namespace communication;

int main()
{
  cout << "----------------------------------------" << endl;
  cout << "Testing the default constructor!" << endl;
  cout << "----------------------------------------" << endl;
  communication::Contact empty;
  empty.display();
  cout << "----------------------------------------" << endl << endl;

  cout << "----------------------------------------" << endl;
  cout << "Testing an invalid contact!" << endl;
  cout << "----------------------------------------" << endl;
  Contact bad(nullptr, nullptr, 0);
  bad.display();
  Contact alsoBad("", nullptr, 0);
  alsoBad.display();
  cout << "----------------------------------------" << endl << endl;

  cout << "----------------------------------------" << endl;
  cout << "Testing the constructor with parameters!" << endl;
  cout << "----------------------------------------" << endl;
  Contact temp("A contact with a very looooong name!", nullptr, 0);
  temp.display();
  cout << "----------------------------------------" << endl << endl;
```

```
  cout << "----------------------------------------" << endl;
  cout << "Testing a valid contact!" << endl;
  cout << "----------------------------------------" << endl;
  long long phoneNumbers[] = { 1416123456, 14161234567, 1416234567890,
                               14162345678, -1, 124163456789};
  Contact someContact("John Doe", phoneNumbers, 6);
  someContact.display();
  cout << "----------------------------------------" << endl << endl;

  return 0;
}
```

IN-LAB OUTPUT:

```
----------------------------------------
Testing the default constructor!
----------------------------------------
Empty contact!
----------------------------------------

----------------------------------------
Testing an invalid contact!
----------------------------------------
Empty contact!
Empty contact!
----------------------------------------

----------------------------------------
Testing the constructor with parameters!
----------------------------------------
A contact with a ver
----------------------------------------

----------------------------------------
Testing a valid contact!
----------------------------------------
John Doe
(+1) 416 123-4567
(+1) 416 234-5678
(+12) 416 345-6789
----------------------------------------
```

IN-LAB SUBMISSION:

To submit the *in-lab* section, demonstrate execution of your program with the exact output as in the example above. Upload `Contact.h`, `Contact.cpp`, and `w6_in_lab.cpp` to your matrix account. Compile and run your code and make sure everything works properly. To submit, run the following script from your account (and follow the instructions):

~profname.proflastname/submit 200_w6_lab<ENTER>

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

AT-HOME (40%):

To the `Contact` class you have created for the *"in-lab"* portion, add the following member functions (and implement them):

copy constructor: this constructor should safely make a copy of an existing instance;

copy assignment operator: this operator should set the current instance to another `Contact` instance, received as a parameter.

`void addPhoneNumber(long long phoneNumber)`: a modifier that adds a new phone number to the contact (this function should resize the array). The phone number must be validated.

NOTE: When you implement the copy assignment operator, make sure that the `Contact` instance is not being set to itself.

Using the sample implementation of the `w6_at_home.cpp` main module shown below, test your code and make sure that it works. Below the source code is the expected output from your program. The output of your program should match **exactly** the expected one.

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

using namespace std;
using namespace communication;

int main() {
  communication::Contact theContact("John Doe", nullptr, 0);
  theContact.addPhoneNumber(14161234567);
  theContact.addPhoneNumber(14162345678);
  theContact.addPhoneNumber(14163456789);
  theContact.addPhoneNumber(114164567890);
  theContact.display();

  cout << endl << "Testing! Please wait:" << endl;
```

```
  for (int i = 1; i <= 5000000; i++)
  {
    Contact temp = theContact;
    theContact = temp;
    theContact = theContact;
    if (!(i % 10000))
      cout << ".";
    if (!(i % 500000))
      cout << endl;
  }
  cout << endl;
  theContact.display();

  return 0;
}
```

AT-HOME OUTPUT:

```
John Doe
John Doe
(+1) 416 123-4567
(+1) 416 234-5678
(+1) 416 345-6789
(+11) 416 456-7890

Testing! Please wait:
..........................................
..........................................
..........................................
..........................................
..........................................
..........................................
..........................................
..........................................
..........................................
..........................................

John Doe
(+1) 416 123-4567
(+1) 416 234-5678
(+1) 416 345-6789
(+11) 416 456-7890
```

AT-HOME REFLECTION (10%):

Create a file `reflect.txt` that contains the following:

1) Explain why the copy assignment operator must check for self-assignment before doing anything else.
2) Explain why a copy constructor and copy assignment operator are needed when a class uses dynamic memory allocation to hold its resources.

3) Explain why is it important to avoid duplication of code and how did you accomplish it in this workshop.

AT-HOME SUBMISSION:

To submit the *at-home* section, demonstrate execution of your program with the exact output as in the example above. Upload `contact.h`, `contact.cpp`, and `w6_at_home.cpp` to your matrix account. Compile and run your code and make sure everything works properly. To submit, run the following script from your account (and follow the instructions):

`~profname.proflastname/submit 200_w6_home``<ENTER>`

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.