/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// OOP244 Workshop 6\_Home:

// File Contact.cpp

// Version 1.0

// Date 10/24/2017

// Student: ALBERTO LAURENZI

// ID:150954162

// email:alaurenzi@myseneca.ca

// Description: Class with a Resource

///////////////////////////////////////////////////////////

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <cstring>

#include "Contact.h"

using namespace std;

namespace sict {

// Implement Default Constructor

Contact::Contact() {

a\_nameCont[0] = '\0';

p\_PhoneNum = nullptr;

numbersPhoneNum = 0;

}

// Implement of the private function to validate the Phone Numbers

bool Contact::isValidPhone(long long phoneVal) {

int phoneNumber = (phoneVal % 10000000);

int areaCode = (phoneVal / 10000000) % 1000;

int countryCode = (phoneVal / 10000000000);

return (phoneNumber >= 1000000 && areaCode >= 100 && countryCode >= 1 &&

countryCode <= 99);

}

//Implementation of Custructor with three parameters

Contact::Contact(const char\* nameContact, const long long\* phoneNums, short count) {

if (nameContact != nullptr && strcmp(nameContact, "") != 0) {

numbersPhoneNum = 0;

strcpy\_s(a\_nameCont, nameContact);

/\*for (int i = 0; i < size; i++) {

a\_nameCont[i] = nameContact[i];

}\*/

a\_nameCont[19] = '\0';

p\_PhoneNum = new long long[count];

for (int i = 0; i < count; i++) {

if (isValidPhone(phoneNums[i])) {

p\_PhoneNum[numbersPhoneNum] = phoneNums[i];

numbersPhoneNum++;

}

}

}

else

{

\*this = Contact();

}

}

//Implementation of the default Contruct

Contact::~Contact() {

delete[] p\_PhoneNum;

}

//Implemetation of Empty function

bool Contact::isEmpty() const {

return (a\_nameCont[0] == '\0' && p\_PhoneNum == nullptr && numbersPhoneNum == 0) ? true : false;

}

// Implemenation Display function

void Contact::display() const {

if (isEmpty()) {

cout << "Empty contact!" << endl;

}

else

{

cout << a\_nameCont << endl;

for (int i = 0; i < numbersPhoneNum; i++) {

cout << "(+" << p\_PhoneNum[i] / 10000000000 << ") ";

cout << (p\_PhoneNum[i] / 10000000) % 1000 << " ";

cout << (p\_PhoneNum[i] / 100) % 100000 / 100 << "-";

cout.width(4);

cout.fill('0');

cout << (p\_PhoneNum[i] % 10000) << endl;

}

}

}

// Implemetation of copy Contruct

Contact::Contact(const Contact& cpyStr) {

p\_PhoneNum = nullptr;

\*this = cpyStr;

}

// Implementation of assignment operator

Contact& Contact::operator=(const Contact& cpyAss)

{

if (this != &cpyAss) {

numbersPhoneNum = cpyAss.numbersPhoneNum;

strcpy\_s(a\_nameCont, cpyAss.a\_nameCont);

if (cpyAss.p\_PhoneNum != nullptr) {

p\_PhoneNum = new long long[numbersPhoneNum];

for (int i = 0; i < numbersPhoneNum; i++) {

p\_PhoneNum[i] = cpyAss.p\_PhoneNum[i];

}

}

else

{

p\_PhoneNum = nullptr;

}

}

return \*this;

}

// implentation of binary operator

Contact& Contact::operator+= (long long new\_PhoneNum) {

if (isValidPhone(new\_PhoneNum) && isEmpty() == false) {

long long\* tempArr;

tempArr = new long long[numbersPhoneNum];

for (int i = 0; i < numbersPhoneNum; i++) {

tempArr[i] = p\_PhoneNum[i];

}

delete[] p\_PhoneNum;

p\_PhoneNum = new long long[numbersPhoneNum + 1];

for (int i = 0; i < numbersPhoneNum; i++)

{

p\_PhoneNum[i] = tempArr[i];

}

p\_PhoneNum[numbersPhoneNum] = new\_PhoneNum;

numbersPhoneNum++;

delete[] tempArr;

}

else

{

p\_PhoneNum[numbersPhoneNum];

}

return \*this;

}

} // namespace sict

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// OOP244 Workshop 6\_Home:

// File Contact.cpp

// Version 1.0

// Date 10/24/2017

// Student: ALBERTO LAURENZI

// ID:150954162

// email:alaurenzi@myseneca.ca

// Description: Class with a Resource

///////////////////////////////////////////////////////////

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef SICT\_CONTACT\_H

#define SICT\_CONTACT\_H

namespace sict {

const char size = 20;

class Contact {

char a\_nameCont[size];

long long\* p\_PhoneNum;

int numbersPhoneNum;

bool isValidPhone(long long phoneVal);

public:

Contact();

Contact(const char\*, const long long\*, short);

~Contact();

bool isEmpty() const;

void display() const;

Contact(const Contact&);

Contact& operator=(const Contact&);

Contact& operator+= (long long);

};

} // namespace sict

#endif // !SICT\_CONTACT\_H