In\_lab

Contact.h

#ifndef SICT\_CONTACT\_H

#define SICT\_CONTACT\_H

namespace sict{

        class Contact{

                private:

                char C\_name[20];

                long long \* C\_phone;

                int phone\_len;

                int j;

                public:

                Contact();

                Contact(const char \*name, const long long phone[], int num);

                void phoneVaild(const int gu, const long long phone[], int num);

                ~Contact();

                bool isEmpty() const;

                void display() const;

        };

}

#endif

Contact.cpp

#include <iostream>

#include <cstring>

#include <iomanip>

#include "Contact.h"

using namespace std;

namespace sict {

        Contact::Contact(){

                C\_name[0] = '\0';

                C\_phone = nullptr;

        }

        Contact::Contact(const char \*name, const long long phone[], int num){

                if(name == nullptr){

                        C\_name[0] = '\0';

                        C\_phone = nullptr;

                }else{

                        strncpy(C\_name, name, 19);

                        C\_name[19] = '\0';

                        phoneVaild(1, phone, num);

                        C\_phone = new long long[phone\_len];

                        phoneVaild(2, phone, num);

                }

        }

        void Contact::phoneVaild(const int gu, const long long phone[], int num){

                phone\_len = 0;

                j = 0;

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

                        long long country = phone[i] / 10000000000LL;

                        long long country\_rest = phone[i] % 10000000000LL;

                        long long area = country\_rest / 1000000000LL;

                        long long tel = country\_rest % 10000000LL;

                                if((country > 0) && (country < 100) && (area > 0) && (tel >= 1000000)){

                                        phone\_len++;

                                        if(gu == 2){

                                                C\_phone[j] = phone[i];

                                                j++;

                                        }

                                }

                }

        }

        Contact::~Contact(){

                delete [] C\_phone;

                C\_phone = nullptr;

        }

        bool Contact::isEmpty() const{

                if(C\_name[0] == '\0'){

                        return true;

                }else{

                        return false;

                }

        }

        void Contact::display() const{

                if(isEmpty()){

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

                }else{

                        cout << C\_name <<endl;

                        for(int i = 0; i < phone\_len; i++){

                                long long country = C\_phone[i] / 10000000000LL;

                                long long int area = (C\_phone[i] % 10000000000LL) / 10000000LL;

                                long long int tel1 = ((C\_phone[i] % 10000000000LL) % 10000000LL) / 10000LL;

                                long long int tel2 = ((C\_phone[i] % 10000000000LL) % 10000000LL) % 10000LL;

                                cout.fill('0');

                                cout << "(+" << country << ") " << area << " " << tel1 << "-" << setw(4) << std::right << tel2 << endl;

                        }

                }

        }

}

At\_home

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

// Workshop 6: Class with a Resource

// File name : Contact.h

// Date : Dec 01, 2017

// Author : Eunhee Kim

//

// Description

// Class definaition and function declarations

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

#ifndef SICT\_CONTACT\_H

#define SICT\_CONTACT\_H

namespace sict{

        class Contact{

                private:

                char C\_name[20];

                long long \* C\_phone;

                int phone\_len;

                int j;

                public:

                Contact();

                Contact(const char \*name, const long long phone[], int num);

                Contact(const Contact& copy);

                void phoneVaild(const int gu, const long long phone[], int num);

                ~Contact();

                bool isEmpty() const;

                void display() const;

                Contact& operator+=(long long telok);

                Contact& operator=(const Contact& con);

        };

}

#endif

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

// Workshop 6: Class with a Resource

// File name : Contact.cpp

// Date : Dec 01, 2017

// Author : Eunhee Kim

//

// Description

// Definition of copy constructor and overload operator

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

#include <iostream>

#include <cstring>

#include <iomanip>

#include "Contact.h"

using namespace std;

namespace sict {

        Contact::Contact(){

                C\_name[0] = '\0';

                C\_phone = nullptr;

                phone\_len = 0;

                j = 0;

        }

        //Copy contructor

        Contact::Contact(const Contact& copy){

                strcpy(C\_name, copy.C\_name);

                phone\_len = copy.phone\_len;

                j = copy.j;

                if(phone\_len > 0){

                        C\_phone = new long long[phone\_len];

                        for(int i = 0; i < phone\_len; i++){

                                C\_phone[i] = copy.C\_phone[i];

                        }

                }else{

                        C\_phone = nullptr;

                }

        }

        Contact::Contact(const char \*name, const long long phone[], int num){

                if(name == nullptr){

                        C\_name[0] = '\0';

                        C\_phone = nullptr;

                }else{

                        phone\_len = 0;

                        j = 0;

                        strncpy(C\_name, name, 19);

                        C\_name[19] = '\0';

                        if(phone == nullptr){

                                C\_phone = nullptr;

                        }else{

                                C\_phone = new long long[phone\_len];

                                phoneVaild(2, phone, num);

                        }

                }

        }

        void Contact::phoneVaild(const int gu, const long long phone[], int num){

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

                        long long country = phone[i] / 10000000000LL;

                        long long country\_rest = phone[i] % 10000000000LL;

                        long long area = country\_rest / 1000000000LL;

                        long long tel = country\_rest % 10000000LL;

                                if((country > 0) && (country < 100) && (area > 0) && (tel >= 1000000)){

                                        phone\_len++;

                                        if(gu == 2){

                                                C\_phone[j] = phone[i];

                                                j++;

                                        }

                                }

                }

        }

        Contact::~Contact(){

                delete [] C\_phone;

                C\_phone = nullptr;

        }

        bool Contact::isEmpty() const{

                if(C\_name[0] == '\0'){

                        return true;

                }else{

                        return false;

                }

        }

        void Contact::display() const{

                if(isEmpty()){

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

                }else{

                        cout << C\_name <<endl;

                        for(int i = 0; i < phone\_len; i++){

                                long long country = C\_phone[i] / 10000000000LL;

                                long long int area = (C\_phone[i] % 10000000000LL) / 10000000LL;

                                long long int tel1 = ((C\_phone[i] % 10000000000LL) % 10000000LL) / 10000LL;

                                long long int tel2 = ((C\_phone[i] % 10000000000LL) % 10000000LL) % 10000LL;

                                cout.fill('0');

                                cout << "(+" << country << ") " << area << " " << tel1 << "-" << setw(4) << std::right << tel2 << endl;

                        }

                }

        }

        // the += operator

        Contact& Contact::operator+=(long long telok) {

//        phone\_len = 0;

//        j = 0;

                long long country = telok / 10000000000LL;

                long long country\_rest = telok % 10000000000LL;

                long long area = country\_rest / 1000000000LL;

                long long tel = country\_rest % 10000000LL;

                long long \*Copy\_C\_phone;

                        if((country > 0) && (country < 100) && (area > 0) && (tel >= 1000000)){

                                Copy\_C\_phone = new long long[phone\_len + 1];

                                for(int i = 0; i <phone\_len; i++){

                                        Copy\_C\_phone[i] = C\_phone[i];

                                }

                                        Copy\_C\_phone[phone\_len] = telok;

                                        delete [] C\_phone;

                                        C\_phone = Copy\_C\_phone;

                                        phone\_len++;

                         }

                        return \*this;

                }

        //the copy assignment operator

        Contact& Contact::operator=(const Contact& con){

           if(this !=  &con){

                strcpy(C\_name, con.C\_name);

                phone\_len = con.phone\_len;

                j = con.j;

                if(phone\_len > 0){

                        C\_phone = new long long[phone\_len];

                        for(int i = 0; i < phone\_len; i++){

                                C\_phone[i] = con.C\_phone[i];

                        }

                }else{

                        C\_phone = nullptr;

                }

            }

                return \*this;

        }

}