Pro1

#include<iostream>

using namespace std;

class Browser

{

public:

virtual void open(const char\* ch) const = 0;

virtual ~Browser() {}

};

class IE :public Browser

{

public:

void open(const char\* ch) const

{

cout << ch << endl;

cout << "opened by ie" << endl;

}

~IE()

{

cout << "~IE" << endl;

}

};

class Firefox :public Browser

{

void open(const char\* ch) const

{

cout << ch << endl;

cout << "opened by Firefox" << endl;

}

~Firefox()

{

cout << "~Firefox" << endl;

}

};

int main()

{

Browser\* pb = new IE();

pb->open("www.microsoft.com");

delete pb;

pb = NULL;

pb = new Firefox();

pb->open("www.firefox.com");

delete pb;

pb = NULL;

}

Pro2

#include<iostream>

using namespace std;

class MyString

{

public:

MyString():str(NULL), length(0) { }

MyString(const char\* \_str)

{

if (\_str)

{

length = strlen(\_str);

str = new char[length + 1];

strcpy\_s(str, length + 1, \_str);

}

else

{

length = 0;

str = NULL;

}

}

MyString(const MyString& temp)

{

if (str == temp.str)

throw invalid\_argument("Error!");

length = temp.length;

if (length)

{

str = new char[length + 1];

strcpy\_s(str, length + 1, temp.str);

}

else

{

str = NULL;

}

}

MyString& operator=(const MyString& temp)

{

length = temp.length;

if (str == temp.str)

return \*this;

if (str)

delete[] str;

if (length)

{

str = new char[length + 1];

strcpy\_s(str, length + 1, temp.str);

}

else

{

str = NULL;

}

}

MyString& operator+=(const MyString& temp)

{

char\* temp2;

int len = length + temp.length;

temp2 = new char[len + 1];

strcpy\_s(temp2, length + 1, str);

strcat\_s(temp2, len + 1, temp.str);

if (str)

delete[] str;

str = temp2;

length = len;

return \*this;

}

friend ostream& operator<<(ostream& o, const MyString& temp)

{

o << temp.str;

return o;

}

~MyString()

{

if (str)

delete[] str;

}

private:

char\* str;

int length;

};

int main()

{

MyString s1("Hello,");

MyString s2(s1);

MyString s3("SEU!");

MyString s4;

s4 = s3;

s2 += s4;

cout << s1 << endl << s2 << endl

<< s3 << endl << s4 << endl;

return 0;

}

Pro3

#include<iostream>

#include<fstream>

#include<iomanip>

#include<vector>

#include<algorithm>

using namespace std;

class Person

{

public:

Person() :name(NULL), number(NULL) {}

Person(const char\* \_name, const char\* \_number)

{

if (\_name)

{

int len = strlen(\_name);

name = new char[len + 1];

strcpy\_s(name, len + 1, \_name);

}

else

{

name = NULL;

}

if (\_number)

{

int len = strlen(\_number);

number = new char[len + 1];

strcpy\_s(number, len + 1, \_number);

}

else

{

number = NULL;

}

}

void setName(const char\* \_name)

{

if (name)

delete[] name;

if (\_name)

{

int len = strlen(\_name);

name = new char[len + 1];

strcpy\_s(name, len + 1, \_name);

}

else

{

name = NULL;

}

}

char\* getName() const

{

return name;

}

void setNumber(const char\* \_number)

{

if (number)

delete[] number;

if (\_number)

{

int len = strlen(\_number);

number = new char[len + 1];

strcpy\_s(number, len + 1, \_number);

}

else

{

number = NULL;

}

}

char\* getNumber() const

{

return number;

}

void display() const

{

cout << "Name: " << name << ", Tel " << number << endl;

}

bool operator<(const Person& p)

{

return name < p.name;

}

private:

char\* name;

char\* number;

};

class Contacts

{

public:

Contacts()

{

fstream read\_file("contacts.txt", ios::in);

if (!read\_file)

{

throw invalid\_argument("OPEN ERROR!");

}

char\* name,\* number;

name = new char[15];

number = new char[15];

while (read\_file >> name >> number)

{

Person person(name, number);

PersonList.push\_back(person);

delete[] name;

delete[] number;

name = new char[15];

number = new char[15];

}

}

void input()

{

char\* name,\* number;

name = new char[15];

number = new char[15];

cout << "Please input name: " << endl;

cin >> name;

cout << "Please input number: " << endl;

cin >> number;

Person person(name, number);

PersonList.push\_back(person);

fstream write\_file("contacts.txt", ios::out | ios::trunc);

if (!write\_file)

{

throw invalid\_argument("OPEN ERROR!");

}

for (int i = 0; i < int(PersonList.size()); i++)

{

write\_file << PersonList[i].getName() << " " << PersonList[i].getNumber() << endl;

}

write\_file.close();

}

void del(const char\* name)

{

int index = -1;

for (int i = 0; i < int(PersonList.size()); i++)

{

if (\*name == \*PersonList[i].getName())

{

index = i;

cout << "Find " << name << "successfully" << endl;

break;

}

}

if (index >= 0)

{

vector<Person> temp;

for (int i = 0; i < int(PersonList.size()); i++)

{

if (i != index)

{

temp.push\_back(PersonList[i]);

}

}

PersonList = temp;

fstream write\_file("contacts.txt", ios::out | ios::trunc);

if (!write\_file)

{

throw invalid\_argument("OPEN ERROR!");

}

for (int i = 0; i < int(PersonList.size()); i++)

{

write\_file << PersonList[i].getName() << " " << PersonList[i].getNumber() << endl;

}

write\_file.close();

}

else

cout << "Name: " << name << " doesn't exit!" << endl;

}

void modify(const char\* name)

{

int index = -1;

for (int i = 0; i < int(PersonList.size()); i++)

{

if (\*name == \*PersonList[i].getName())

{

index = i;

cout << "Find " << name << "successfully" << endl;

break;

}

}

if (index >= 0)

{

fstream write\_file("contacts.txt", ios::out | ios::trunc);

if (!write\_file)

{

throw invalid\_argument("OPEN ERROR!");

}

char\* number;

number = new char[15];

cout << "Please input number" << endl;

cin >> number;

PersonList[index].setNumber(number);

delete[] number;

for (int i = 0; i < int(PersonList.size()); i++)

{

write\_file << PersonList[i].getName() << " " << PersonList[i].getNumber() << endl;

}

write\_file.close();

}

else

cout << "Name: " << name << " doesn't exit!" << endl;

}

void display()

{

sort(PersonList.begin(),PersonList.end());

for (int i = 0; i < int(PersonList.size()); i++)

PersonList[i].display();

}

private:

vector<Person> PersonList;

};

int main()

{

while (true)

{

Contacts info;

int flag;

char\* name;

cout << "1 - display all person information sorted by name\n"

<< "2 - input a new person's information\n"

<< "3 - delet a person's information specified by name\n"

<< "4 - modify a person's mobile phone number specified by name\n"

<< "5 - exit the system" << endl;

cin >> flag;

switch (flag)

{

case 1:

info.display();

break;

case 2:

info.input();

break;

case 3:

cout << "Please input the name you want to delete: " << endl;

name = new char[15];

cin >> name;

info.del(name);

delete[] name;

break;

case 4:

cout << "Please input the name you want to modify: " << endl;

name = new char[15];

cin >> name;

info.modify(name);

delete[] name;

break;

case 5:

break;

}

}

return 0;

}