ФГАОУ ВО Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

**ОТЧЕТ**

**по лабораторной работе №7**

**по теме «Основные концепции STL»**

по дисциплине «Практикум по программированию»

Выполнил

студент гр.23533/2 А.Д. Шурак

Проверил

ассистент В.Э. Ковалевский

Санкт-Петербург

2019

Задание

Лабораторную работу со списками, выполненную в прошлом семестре, реализовать с использованием STL.

Разработка программы

Реализуем программу, которая будет осуществлять работу с динамическими списками, используя STL.

Заменяем шаблонный класс List на класс Vector из библиотеки <vector>.

#include <vector>

Все необходимые классы мы можем перенести, но некоторые методы, например для помещения элементов в список, заменяются на стандартные методы vector.

После определения пустого вектора, можно с помощью различных функций добавлять в его конец элементы.  
Добавление элементов в конец вектора реализовано с помощью метода push\_back.  
  
 void push\_back(\_Ty&& \_Val)

{ // insert by moving into element at end, provide strong guarantee

emplace\_back(\_STD move(\_Val));

}

Часть методов осталась неизменной, некоторые необходимо добавить:

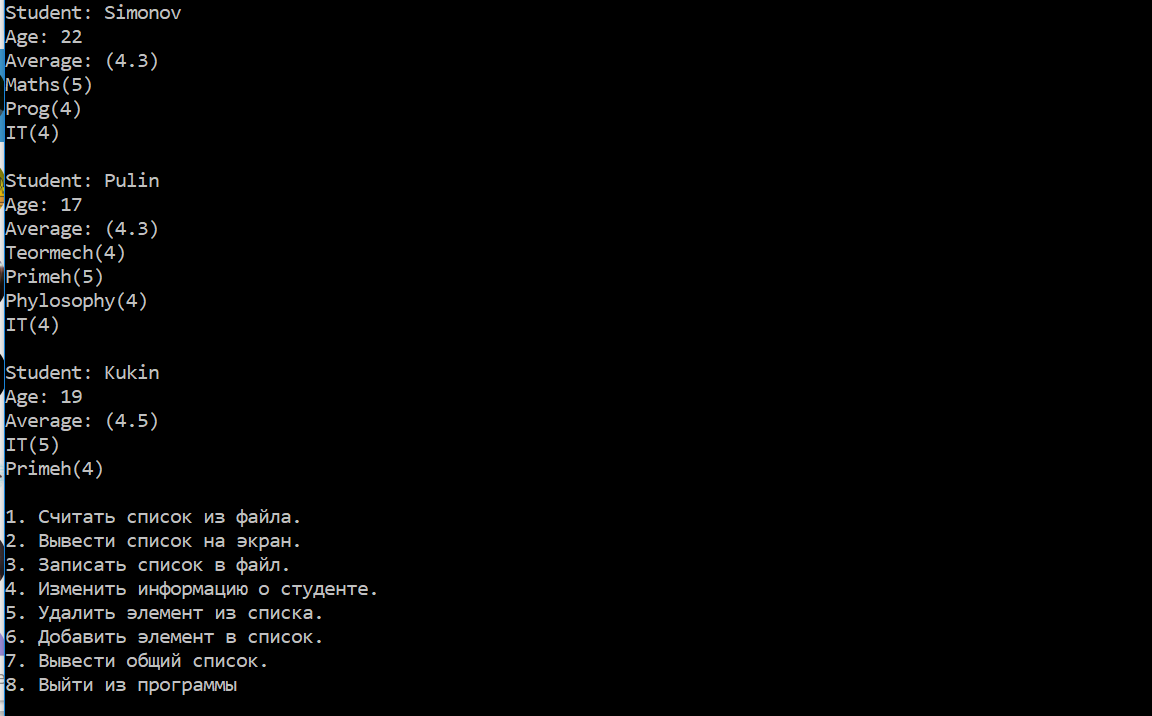
* метод для заполнения списка из файла
* метод для вывода элементов списка на экран
* метод для записи списка в файл
* метод для поиска элемента по имени

Класс vector работает немного иначе, чем List: в качестве аргументов принимается, например, ссылка на объект класса Vector или (и) указатель на файл. В некоторый случаях, как в методе для поиска элемента по имени, аргументом будет являться также объект класса string.

Рассмотрим изменения, совершенные над классами.   
  
Класс Exams больше не имеет статической переменной (объект List). Заменяя объект static на объект класса vector, возникает необходимость добавить ещё два статических метода для вывода на экран и для поиска нужного предмета.

Изменения в остальных классах примерно похожие: заменяем объекты класса List на объекты и методы класса Vector. Немного изменяем методы, не меняя их сути.  
Например, у класса Student в списке сданных экзаменов студента используется объект класса Vector, в который помещен объект класса Exams. Таким образом, чтобы вывести список экзаменов на экран и осуществить поиск, нужно добавить методы и реализовать поиск через проход по списку в цикле.  
  
Лабораторная работа изменилась «внутри», сократилось количество кода, но не изменился функционал.

**Демонстрация работы программы**



**Заключение**

В ходе выполнения лабораторной работы была преобразована программа с прошлого семестра с использованием STL.

**Приложение**

#include <vector>

#include <iostream>

#include "Exams.h"

#include "People.h"

#include "Subject.h"

#define INSTUDENTS "students.txt"

#define INPROFESSORS "professors.txt"

#define OUTLIST "output.txt"

#define INSUBJ "subjects.txt"

using namespace std;

std::vector<Subject> Exams::SubjectList1;

template<class T>

void FromFile(vector<T> &a, const char\* file)

{

ifstream fin(file);

while (fin)

{

T newc;

bool flag = newc.Scan(fin);

if (!flag)

{

a.push\_back(newc);

}

}

fin.close();

}

template<class T>

void OutList(vector<T> &a)

{

for (int i = 0; i < (a.end() - a.begin()); i++)

cout << a[i];

}

template <class T>

T\* FindPerson(vector<T> &a, string name1)

{

for (int i = 0; i < (a.end() - a.begin()); i++)

{

if (a[i] == name1)

return &a[i];

}

return NULL;

}

template <class T>

void ToFile(vector<T>& a)

{

ofstream fout(OUTLIST);

for (int i = 0; i < (a.end() - a.begin()); i++)

fout << a[i];

fout.close();

}

int main()

{

char cntrl, ch;

string name1;

setlocale(LC\_ALL, "");

FromFile(Exams::SubjectList1, INSUBJ);

vector<Student> S;

vector<Teacher> P;

do

{

cout << "1. Считать список из файла.\n";

cout << "2. Вывести список на экран.\n";

cout << "3. Записать список в файл.\n";

cout << "4. Изменить информацию о студенте.\n";

cout << "5. Удалить элемент из списка.\n";

cout << "6. Добавить элемент в список.\n";

cout << "7. Вывести общий список.\n";

cout << "8. Выйти из программы\n";

cin >> cntrl;

switch (cntrl)

{

case'1':

cout << "Выберите список для использования:\n1. Студенты.\n2. Преподаватели.\n";

cin >> ch;

if (ch == '1')

FromFile(S, INSTUDENTS);

if (ch == '2')

FromFile(P, INPROFESSORS);

break;

case'2':

cout << "Выберите список для использования:\n1. Студенты.\n2. Преподаватели.\n";

cin >> ch;

if (ch == '1')

{

if (S.empty())

{

cout << "Список пустой.\n"; break;

}

OutList(S);

}

if (ch == '2')

{

if (P.empty())

{

cout << "Список пустой.\n"; break;

}

OutList(P);

}

break;

case'3':

cout << "Выберите список для использования:\n1. Студенты.\n2. Преподаватели.\n";

cin >> ch;

if (ch == '1')

{

if (S.empty())

{

cout << "Список пустой.\n"; break;

}

ToFile(S);

}

if (ch == '2')

{

if (P.empty())

{

cout << "Список пустой.\n"; break;

}

ToFile(P);

}

break;

case'4':

{

if (S.empty())

{

cout << "Список пустой.\n"; break;

}

cout << "Введите имя:\n";

cin >> name1;

Student\* old\_one = NULL;

old\_one = FindPerson(S, name1);

if (old\_one == NULL)

{

cout << "Студент не обнаружен.\n"; break;

}

old\_one->Edit();

break;

}

case'5':

{

cout << "Выберите список для использования:\n1. Студенты.\n2. Преподаватели.\n";

cin >> ch;

if (ch == '1')

{

if (S.empty())

{

cout << "Список пустой.\n"; break;

}

cout << "Введите имя:\n";

cin >> name1;

for (int i = 0; i < (S.end() - S.begin()); i++)

{

if (S[i] == name1)

{

S[i].DeleteEx();

S.erase(S.begin() + i);

break;

}

}

}

if (ch == '2')

{

if (P.empty())

{

cout << "Список пустой.\n"; break;

}

cout << "Введите имя:\n";

cin >> name1;

for (int i = 0; i < (P.end() - P.begin()); i++)

{

if (P[i] == name1)

{

P[i].Delete();

P.erase(P.begin() + i);

break;

}

}

}

break;

}

case'6':

{

cout << "Выберите список для использования:\n1. Студенты.\n2. Преподаватели.\n";

cin >> ch;

if (ch == '1')

{

cout << "Введите имя:\n";

cin >> name1;

Student new\_comer;

if (FindPerson(S, name1) == NULL)

{

new\_comer.AddData(name1);

S.push\_back(new\_comer);

}

else

cout << "Такой студент уже есть в списке.\n";

}

if (ch == '2')

{

cout << "Введите имя:\n";

cin >> name1;

Teacher new\_comer;

if (FindPerson(P, name1) == NULL)

{

new\_comer.AddData(name1);

P.push\_back(new\_comer);

}

else

cout << "Такой преподаватель уже есть в списке.\n";

}

break;

}

case'7':

{

if (P.empty() && S.empty())

{

cout << "Списки пустые.\n"; break;

}

vector<People\*> H;

for (int i = 0; i < (S.end() - S.begin()); i++)

{

H.push\_back(&S[i]);

}

for (int i = 0; i < (P.end() - P.begin()); i++)

{

H.push\_back(&P[i]);

}

for (int i = 0; i < (H.end() - H.begin()); i++)

cout << \*H[i];

break;

}

}

} while (cntrl != '8');

}

#pragma once

#ifndef People\_H

#define People\_H

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <vector>

#include "Exams.h"

#include "Subject.h"

class Exams;

class Subject;

class People

{

protected:

std::string name;

int age;

public:

bool operator==(std::string s\_name) { return(name == s\_name); }

bool ScanPeople(std::ifstream &f);

virtual std::ostream& OutData(std::ostream &f)

{

f << "Имя: " << name << "\nВозраст: " << age << "\n";

return f;

}

virtual void AddData(std::string input\_name)

{

name = input\_name;

std::cout << "Введите возраст: ";

std::cin >> age;

}

friend std::ostream& operator<<(std::ostream & f, People &data);

};

class Student : public People

{

private:

int size;

float avg;

std::vector<Exams> pass;

public:

Student()

{

size = 0;

avg = 0;

}

void DeleteEx();

std::ostream& OutData(std::ostream &f)

{

f << "Student: " << name << "\nAge: " << age << "\nAverage: (" << avg << ")\n";

OutPass(f);

return f;

}

void AddEx();

std::ostream& OutPass(std::ostream &f);

bool Scan(std::ifstream &f);

void average();

void AddData(std::string input\_name)

{

name = input\_name;

std::cout << "Введите возраст: ";

std::cin >> age;

AddEx();

}

void Edit();

};

class Teacher : public People

{

private:

Subject \*subj;

public:

bool Scan(std::ifstream &f);

std::ostream& OutData(std::ostream &f)

{

f << "Teacher: " << name << "\nAge: " << age << "\n" << subj << "\n";

return f;

}

void AddData(std::string input\_name)

{

name = input\_name;

std::cout << "Введите возраст: ";

std::cin >> age;

AddSubject();

}

void Delete();

void AddSubject();

};

#endif

#include "Subject.h"

bool Subject::Scan(std::ifstream &f)

{

try

{

f >> (SubjName);

if (SubjName.length() == 0) return 1;

return 0;

}

catch (...)

{

return 1;

}

}

std::ostream& operator<<(std::ostream & f, Subject \*data)

{

f << data->SubjName;

return f;

}

std::ostream& operator<<(std::ostream & f, Subject &data)

{

f << data.SubjName << "\n";

return f;

}

#pragma once

#ifndef Subject\_H

#define Subject\_H

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

class Subject

{

private:

std::string SubjName;

public:

bool operator==(std::string subjectname) { return(SubjName == subjectname); }

friend std::ostream& operator<<(std::ostream & f, Subject \*data);

friend std::ostream& operator<<(std::ostream & f, Subject &data);

std::string GetSubjectName() { return SubjName; }

bool Scan(std::ifstream &f);

};

#endif

#pragma once

#ifndef Exams\_H

#define Exams\_H

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <vector>

#include "Subject.h"

class Subject;

class Exams

{

private:

int mark;

Subject\* subject;

public:

static std::vector<Subject> SubjectList1;

static void OutSubject()

{

for (int i = 0; i < (Exams::SubjectList1.end() - Exams::SubjectList1.begin()); i++)

std::cout << Exams::SubjectList1[i];

}

static Subject\* ToFindSubject(std::string s\_name)

{

for (int i = 0; i < (SubjectList1.end() - SubjectList1.begin()); i++)

{

if (Exams::SubjectList1[i] == s\_name)

return &Exams::SubjectList1[i];

}

return NULL;

}

Exams()

{

subject = NULL;

mark = 0;

}

bool operator==(std::string name1) { return(\*subject == name1); }

friend std::ostream& operator<<(std::ostream & f, Exams &data);

bool Scan(std::ifstream &f);

void AddData();

int getmark() { return mark; }

void setmark(int inpmark) { mark = inpmark; }

void setsubject(Subject \*inpsubj) { subject = inpsubj; }

};

#endif