Автор: Татаренко А., КІТ-119а

Дата: 17 червня 2020

Лабораторна робота №14.

СОРТУВАННЯ

Тема. STL. Алгоритми зміни послідовності. Сортування. Функтори.

Мета – на практиці порівняти STL-алгоритми, що модифікують послідовність; отримати навички роботи з STL-функторами.

1 Завдання до роботи

Індивідуальне завдання 19.

Поширити попередню лабораторну роботу, додаючи такі можливості діалогового меню:
- об’єднання двох STL-контейнерів типу vector;
- сортувати заданий контейнер з використанням функтора

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

class1.h

```
#pragma once
#include "Header.h"

class Student
{
protected:
	int age;
	int number_stud;
	int middle_mark;
	string name;
	bool debt;
	int prog_d;

public:
	virtual string get_info() const;
	virtual stringstream get_str() const;
	int get_numb() const;
	virtual bool elementOutput(int, string);
	virtual int countElement(int, string);

	Student();
	Student(int, int, int, string, bool, int);
	Student(const Student&);
	virtual ~Student();

	friend ostream& operator<< (ostream&, const Student&);
	virtual bool operator==(const int) const;

};
```

class2.h

```
#pragma once
#include "class1.h"

class Course final : public Student
{
private:
	int course;

public:
	string get_info() const override final;
	stringstream get_str() const override final;
	bool elementOutput(int, string) override final;
	int countElement(int, string) override final;

	Course();
	Course(int, int, int, string, bool, int, int);
	Course(const Course&);
	~Course() override final;

	bool operator==(const int) const override final;

};
```

Functor.h

```
#pragma once

#include "class1.h"
```

```
class Functor
{
private:
	int value;

public:
	bool operator()(const shared_ptr<Student>&, const shared_ptr<Student>&);

	Functor(int);
	~Functor();
};
```

Header.h

```
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE)

#include <string>
#include <iostream>
#include <iomanip>
#include <locale>
#include <fstream>
#include <sstream>
#include <istream>
#include <vector>
#include <memory>
#include <list>
#include <map>
#include <set>
#include <unordered_set>
#include <algorithm>
#include <iterator>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::boolalpha;
using std::setiosflags;
using std::ios;
using std::ifstream;
using std::ostream;
using std::ofstream;
using std::stringstream;
using std::istream;
using std::vector;
using std::list;
using std::map;
using std::set;
using std::unordered_set;
using std::unique_ptr;
using std::shared_ptr;
using std::advance;
using std::stoi;
using std::for_each;
using std::make_move_iterator;
using std::set_intersection;
using std::back_inserter;
```

```
using std::pair;
using std::transform;
using std::inserter;
```

class1.cpp

```
#include "class1.h"

string Student::get_info() const
{
	stringstream temp;

	temp.setf(std::ios::left);
	temp << setw(10) << age << setw(8) << number_stud << setw(16) << middle_mark <<
setw(9)
		<< name << setw(7) << debt << setw(14) << prog_d;

	return temp.str();
}

int Student::get_numb() const
{
	return number_stud;
}

stringstream Student::get_str() const
{
	stringstream temp;
	temp << " " << age << " " << number_stud << " " << middle_mark << " "
		<< name << " " << debt << " " << prog_d;

	return temp;
}

int Student::countElement(int value, string data)
{
	try
	{
		if (value == 1)
		{
			if (this->name == data)
				return 1;
			else
				return 0;
		}
		else if (value == 2)
		{
			int number = stoi(data);
			if (this->age == number)
				return 1;
			else
				return 0;
		}
		else if (value == 3)
		{
			int number = stoi(data);
			if (this->middle_mark == number)
				return 1;
			else
				return 0;
		}
		else if (value == 4)
		{
			int number = stoi(data);
```

```
                        if (this->prog_d == number)
                                return 1;
                        else
                                return 0;
                }
                else if (value == 5)
                {
                        int number = stoi(data);
                        if (this->number_stud == number)
                                return 1;
                        else
                                return 0;
                }
                else if (value == 6)
                {
                        int number = 0;
                        if (data == "true" || data == "true" || data == "1")
                                number = 1;
                        else
                                number = 0;

                        if (this->debt == number)
                                return 1;
                        else
                                return 0;
                }
        }
        catch (const std::exception& ex)
        {
                cout << ex.what() << endl;
                return 0;
        }

        return 0;
}

bool Student::elementOutput(int value, string data)
{
        try
        {
                if (value == 1)
                {
                        if (this->name == data)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 2)
                {
                        int number = stoi(data);
                        if (this->age == number)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 3)
                {
                        int number = stoi(data);
                        if (this->middle_mark == number)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 4)
                {
                        int number = stoi(data);
                        if (this->prog_d == number)
                                cout << *this << endl;
```

```
                return true;
            }
            else if (value == 5)
            {
                int number = stoi(data);
                if (this->number_stud == number)
                    cout << *this << endl;
                return true;
            }
            else if (value == 6)
            {
                int number = 0;
                if (data == "true" || data == "true" || data == "1")
                    number = 1;
                else
                    number = 0;

                if (this->debt == number)
                    return 1;
                else
                    return 0;
            }
        }
        catch (const std::exception& ex)
        {
            cout << ex.what() << endl;
            return 0;
        }

        return 0;
}

ostream& operator<< (ostream& output, const Student& other)
{
        output << other.get_info();
        return output;
}

bool Student::operator==(const int ns) const
{
        return this->number_stud == ns;
}

Student::Student(int a, int n, int m, string na, bool d, int pd) : age(a),
number_stud(n), middle_mark(m), name(na), debt(d), prog_d(pd)
{
        //cout << "\nВызвался конструктор с параметрами";
}
Student::Student() : age(17), number_stud(0), middle_mark(8), name("Bond"), debt(1),
prog_d(15)
{
        //cout << "\nВызвался конструктор по умолчанию.";
}
Student::Student(const Student& other) : age(other.age), number_stud(other.number_stud),
middle_mark(other.middle_mark), name(other.name), debt(other.debt), prog_d(other.prog_d)
{
        //cout << "\nВызвался конструктор копирования.";
}
Student::~Student()
{
        //cout << "\nВызвался деструктор";

}
```

class2.cpp

```
#include "class2.h"

stringstream Course::get_str() const
{
	stringstream temp;

	temp << " " << age << " " << number_stud << " " << middle_mark << " "
		<< name << " " << debt << " " << prog_d << " " << course;

	return temp;
}

string Course::get_info() const
{
	stringstream temp;

	temp.setf(ios::left);
	temp << setw(10) << age << setw(8) << number_stud << setw(16) << middle_mark <<
setw(9)
		<< name << setw(7) << debt << setw(14) << prog_d << setw(4) << course;

	return temp.str();
}

int Course::countElement(int value, string data)
{
	try
	{
		if (value == 1)
		{
			if (this->name == data)
				return 1;
			else
				return 0;
		}
		else if (value == 2)
		{
			int number = stoi(data);
			if (this->age == number)
				return 1;
			else
				return 0;
		}
		else if (value == 3)
		{
			int number = stoi(data);
			if (this->middle_mark == number)
				return 1;
			else
				return 0;
		}
		else if (value == 4)
		{
			int number = stoi(data);
			if (this->prog_d == number)
				return 1;
			else
				return 0;
		}
		else if (value == 5)
		{
			int number = stoi(data);
```

```
                        if (this->number_stud == number)
                                return 1;
                        else
                                return 0;
                }
                else if (value == 6)
                {
                        int number = 0;
                        if (data == "true" || data == "true" || data == "1")
                                number = 1;
                        else
                                number = 0;

                        if (this->debt == number)
                                return 1;
                        else
                                return 0;
                }
                else if (value == 7)
                {
                        int number = stoi(data);
                        if (this->course == number)
                                return 1;
                        else
                                return 0;
                }
        }
        catch (const std::exception & ex)
        {
                cout << ex.what() << endl;
                return 0;
        }

        return 0;
}

bool Course::elementOutput(int value, string data)
{
        try
        {
                if (value == 1)
                {
                        if (this->name == data)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 2)
                {
                        int number = stoi(data);
                        if (this->age == number)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 3)
                {
                        int number = stoi(data);
                        if (this->middle_mark == number)
                                cout << *this << endl;
                        return true;
                }
                else if (value == 4)
                {
                        int number = stoi(data);
                        if (this->prog_d == number)
                                cout << *this << endl;
```

```
                return true;
        }
        else if (value == 5)
        {
                int number = stoi(data);
                if (this->number_stud == number)
                        cout << *this << endl;
                return true;
        }
        else if (value == 6)
        {
                int number = 0;
                if (data == "true" || data == "true" || data == "1")
                        number = 1;
                else
                        number = 0;

                if (this->debt == number)
                        return 1;
                else
                        return 0;
        }
        else if (value == 7)
        {
                int number = stoi(data);
                if (this->course == number)
                        cout << *this << endl;
                return true;
        }
    }
    catch (const std::exception & ex)
    {
            cout << ex.what() << endl;
            return 0;
    }

    return 0;
}

Course::Course(int a, int n, int m, string na, bool d, int pd, int c) : Student(a, n, m,
na, d, pd), course(c) {}
Course::Course() : Student(), course(1) {}
Course::Course(const Course& other) : Student(other), course(other.course) {}
Course::~Course() {}

bool Course::operator==(const int ns) const
{
    return this->number_stud == ns;
}
```

Functor.cpp

```
#include "Functor.h"

bool Functor::operator() (const shared_ptr<Student>& st1, const shared_ptr<Student>& st2)
{
    if (value % 2 != 0)
            return st1->get_numb() < st2->get_numb();
    else
            return st1->get_numb() > st2->get_numb();
}

Functor::Functor(int value) :value(value) {}
```

```
Functor::~Functor() {}
```

main.cpp

```
#include "class2.h"

#include "Functor.h"



Student* newProgram(int);

void VectorMenu();

void ListMenu();

void MapMenu();

void SetMenu();



vector <shared_ptr<Student>> CombineVectors(vector<shared_ptr<Student>>&, vector <shared_ptr<Student>>&);

map <int, shared_ptr<Student>> CombineMaps(map<int, shared_ptr<Student>>&, map<int,
shared_ptr<Student>>&);



int main()

{

	setlocale(LC_ALL, "ru");



	int choise = 0;

	bool stop = 1;



	while (stop)

	{
```

```
cout << "Выберите STL контейнер:" << endl;
cout << "1) Vector;" << endl;
cout << "2) List;" << endl;
cout << "3) Map;" << endl;
cout << "4) Set;" << endl;
cout << "5) Выход;" << endl;
cout << "======================" << endl;
cout << "Пункт: ";
cin >> choise;

switch (choise)
{
case 1:
	VectorMenu();
	break;

case 2:
	ListMenu();
	break;

case 3:
	MapMenu();
	break;
```

```
		case 4:
			SetMenu();
			break;

		case 5:
			stop = 0;
			break;

		default:
			cout << "Ошибка. Неверная команда. Повторите попытку." << endl;
		}
	}

	if (_CrtDumpMemoryLeaks())
		cout << endl << "Обнаружена утечка памяти!" << endl;
	else
		cout << endl << "Утечки не обнаруженно!" << endl;

	system("PAUSE");
	return 0;
}

Student* newProgram(int value)
{
```

```
	if (value % 2 == 0)
	{
		Student* temp = new Course(17, 4, 10, "Peter", 0, 0, 1);
		return temp;
	}
	else
	{
		Student* temp = new Student(19, 4, 9, "Jhon", 1, 14);
		return temp;
	}
}

vector <shared_ptr<Student>> CombineVectors(vector<shared_ptr<Student>>& first, vector
<shared_ptr<Student>>& second)
{
	vector <shared_ptr<Student>> resultVector;

	resultVector.insert(resultVector.end(), make_move_iterator(first.begin()),
make_move_iterator(first.end()));
	resultVector.insert(resultVector.end(), make_move_iterator(second.begin()),
make_move_iterator(second.end()));

	cout << endl << "Векторы объединены." << endl;
	return resultVector;
}
```

```
map <int, shared_ptr<Student>> CombineMaps(map<int, shared_ptr<Student>>& firstMap, map<int, shared_ptr<Student>>& secondMap)

{

	map <int, shared_ptr<Student>> resultMap;

	vector <shared_ptr<Student>> data1;

	vector <shared_ptr<Student>> data2;

	vector <int> map1Keys;

	vector <int> map2Keys;

	vector <int> temp;

	vector <int> res;


	for (auto const& it : firstMap)

		map1Keys.push_back(it.first);

	for (auto const& it : secondMap)

		map2Keys.push_back(it.first);


	for (auto const& it : firstMap)

		data1.push_back(it.second);

	for (auto const& it : secondMap)

		data2.push_back(it.second);


	sort(map1Keys.begin(), map1Keys.end());

	sort(map2Keys.begin(), map2Keys.end());
```

```
        set_intersection(map1Keys.begin(), map1Keys.end(), map2Keys.begin(), map2Keys.end(),
back_inserter(res));


        temp.insert(temp.end(), map1Keys.begin(), map1Keys.end());

        temp.insert(temp.end(), map2Keys.begin(), map2Keys.end());

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

        temp.erase(unique(temp.begin(), temp.end()), temp.end());


        auto it1 = map1Keys.begin();

        auto it2 = map2Keys.begin();

        stringstream ss1, ss2;

        int count1, count2;


        int age1, age2;

        string debtTF1, debtTF2;

        bool debt;

        int prog_d1, prog_d2;

        int middle_mark1, middle_mark2;

        int number_stud1, number_stud2, number_stud3;

        string name1, name2;

        int course1, course2;


        string value1, value2;
```

```
for (size_t i = 0; i < temp.size(); i++)
{
	if (find(res.begin(), res.end(), i + 1) != res.end())
	{
		auto itnumber_stud1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
		auto itnumber_stud2 = find(map2Keys.begin(), map2Keys.end(), i + 1);

		number_stud1 = distance(map1Keys.begin(), itnumber_stud1);
		number_stud2 = distance(map2Keys.begin(), itnumber_stud2);

		ss1 = data1[number_stud1]->get_str();
		ss2 = data2[number_stud2]->get_str();

		value1 = ss1.str();
		value2 = ss2.str();

		count1 = count(value1.begin(), value1.end(), ' ');
		count2 = count(value2.begin(), value2.end(), ' ');

		ss1 >> age1;
		ss1 >> number_stud1;
		ss1 >> middle_mark1;
		ss1 >> name1;
```

```
ss1 >> debtTF1;

ss1 >> prog_d1;

if (count1 == 6)

        ss1 >> course1;



ss2 >> age2;

ss2 >> number_stud2;

ss2 >> middle_mark2;

ss2 >> name2;

ss2 >> debtTF2;

ss2 >> prog_d2;

if (count2 == 6)

        ss2 >> course2;



name1 += name2;

number_stud1 += number_stud2;

age1 += age2;

prog_d1 += prog_d2;

middle_mark1 += middle_mark2;

if (debtTF1 == "1" || debtTF2 == "1")

        debt = true;

if (count1 == 6 || count2 == 6)

{

        course1 += course2;
```

```
                                resultMap.emplace(i + 1, new Course(age1, number_stud1, middle_mark1,
name1, debt, prog_d1, course1));
                        }
                        else
                                resultMap.emplace(i + 1, new Student(age1, number_stud1, middle_mark1,
name1, debt, prog_d1));


                }
                else
                {
                        it1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
                        if (it1 != map1Keys.end())
                        {
                                number_stud3 = distance(map1Keys.begin(), it1);
                                resultMap.emplace(i + 1, data1[number_stud3]);
                        }
                        else
                        {
                                it2 = find(map2Keys.begin(), map2Keys.end(), i + 1);
                                number_stud3 = distance(map2Keys.begin(), it2);
                                resultMap.emplace(i + 1, data2[number_stud3]);
                        }
                }
        }
```

```
	map1Keys.erase(map1Keys.begin(), map1Keys.end());
	map2Keys.erase(map2Keys.begin(), map2Keys.end());
	temp.erase(temp.begin(), temp.end());
	res.erase(res.begin(), res.end());
	data1.erase(data1.begin(), data1.end());
	data2.erase(data2.begin(), data2.end());

	return resultMap;
}

void VectorMenu()
{
	vector <shared_ptr<Student>> vector;
	std::vector <shared_ptr<Student>> mergeVector;
	std::vector<shared_ptr<Student>>::iterator it;
	stringstream temp;
	string data;
	bool stop = 1, findEl = 0;
	int choise = 0, choise2 = 0, choise3 = 0;
	int value = 0, number = 0, result = 0, sum = 0;

	for (size_t i = 0; i < 4; i++)
	{
		if (i == 0)
```

```
                vector.emplace_back(new Student());
        else if (i == 1)
                vector.emplace_back(new Course(20, 1, 10, "Den", 0, 0, 3));
        else if (i == 2)
                vector.emplace_back(new Student(18, 2, 8, "Dima", 0, 0));
        else if (i == 3)
                vector.emplace_back(new Course(19, 3, 7, "Gordon", 1, 25, 2));
}

while (stop != 0)
{
        if (vector.size() == 0)
        {
                cout << "Вектор пуст. Что вы хотите сделать?" << endl;
                cout << "1) Добавить элемент" << endl;
                cout << "2) Завершение работы" << endl;
                cout << "=====================================" << endl;
                cout << "Ваш выбор: ";
                cin >> choise;
                cout << endl;

                switch (choise)
                {
                case 1:
```

```
cout << "Выберите программу, которую хотите добавить:" << endl;
cout << "1. Элемент класса Student" << endl;
cout << "2. Элемент класса Course" << endl;
cout << "=========================================" << endl;
cout << "Ваш выбор: ";
cin >> value;

try
{
	vector.at(value);

	if (value == 1 || value == 2)
	{
		vector.emplace_back(newProgram(value));
		cout << "Элемент добавлен." << endl;
	}
	else
		cout << "Ошибка. Неверный номер." << endl;
}
catch (const std::exception& ex)
{
	cout << ex.what() << endl;
}
```

```
                break;


        case 2:
                cout << "Завершение работы." << endl;
                stop = 0;
                break;


        default:
                cout << "Неверный номер элемента. Повторите попытку." << endl;
                break;
        }
}
else
{
        cout << endl;
        cout << "1)Вывод на экран" << endl;
        cout << "2)Удаление" << endl;
        cout << "3)Добавление" << endl;
        cout << "4)Объединить векторы" << endl;
        cout << "5)Сортировка" << endl;
        cout << "6)Завершение работы" << endl;
        cout << "====================" << endl;
        cout << "Ваш выбор: ";
```

```
        cin >> choise;
        cout << endl;
}

switch (choise)
{
case 1:
        cout << "Выберите команду:" << endl;
        cout << "1) Вывести весь список на экран" << endl;
        cout << "2) Вывести программу по ID" << endl;
        cout << "3) Вывести количество элементов по критерию" << endl;
        cout << "4) Найти элемент по критерию" << endl;
        cout << "5) Вернуться к выбору действий" << endl;
        cout << "====================================" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;

        switch (choise2)
        {
        case 1:
                cout << setw(10) << "Возраст" << setw(8) << "Номер";
                cout << setw(15) << "Средний балл" << setw(7) << "Имя";
                cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
```

```
                cout << setw(7) << "Курс" << endl;


                number = 1;
                for_each(vector.begin(), vector.end(), [&number](const shared_ptr<Student>&
program)
                {
                        cout << number << ". " << *program << endl;
                        number++;
                });
                number = 1;
                break;


        case 2:
                cout << "Введите id элемента, которого вы хотите получить: ";
                cin >> value;
                cout << endl;


                findEl = 0, number = -1;
                for (const auto& element : vector)
                {
                        if (element->get_numb() == value)
                        {
                                number++;
                                findEl = 1;
```

```
                    break;
                }
                else
                    number++;
            }

            if (findEl)
            {
                temp = vector[number]->get_str();
                data = temp.str();
                cout << "Ваш элемент: " << endl;
                cout << data << endl << endl;
            }
            else
                cout << "Элемент с таким ID не найден." << endl;

            break;
        case 3:
            cout << "Выберите критерий, по которому надо искать: " << endl;
            cout << "1) Имя" << endl;
            cout << "2) Возраст" << endl;
            cout << "3) Средний балл" << endl;
            cout << "4) Долг по прог." << endl;
            cout << "5) Номер" << endl;
```

```
cout << "6) Есть ли долг" << endl;

cout << "7) Вернуться назад" << endl;

cout << "=====================================" << endl;

cout << "Ваш выбор: ";

cin >> choise3;

cout << endl;



if (choise3 < 1 || choise3 >= 7)

{

	cout << "Возвращение назад." << endl;

	break;

}



it = vector.begin();



cout << "Введите критерий: ";

cin.ignore();

getline(cin, data);

number = 0, value = 0;



while (number < vector.size())

{

	result = (*it)->countElement(choise3, data);

	number++;
```

```
                it++;
                sum += result;
            }
            if (sum != 0)
                cout << "Количество элементов с данным параметром: " << sum <<
endl;

            break;

        case 4:
            cout << "Выберите критерий, по которому надо искать: " << endl;
            cout << "1) Имя" << endl;
            cout << "2) Возраст" << endl;
            cout << "3) Средний балл" << endl;
            cout << "4) Долг по прог." << endl;
            cout << "5) Номер" << endl;
            cout << "6) Есть ли долг" << endl;
            cout << "7) Вернуться назад" << endl;
            cout << "=====================================" << endl;
            cout << "Ваш выбор: ";
            cin >> choise3;
            cout << endl;

            if (choise3 < 1 || choise3 >= 7)
```

```
		{
			cout << "Возвращение назад." << endl;
			break;
		}

		it = vector.begin();
		cout << "Введите критерий: ";
		cin.ignore();
		getline(cin, data);
		number = 0, value = 0;

		while (number < vector.size())
		{
			result = (*it)->elementOutput(choise3, data);
			number++;
			it++;
		}

		break;
	case 5:
		cout << "Возвращение назад." << endl;
		break;

	default:
```

```
                cout << "Неверный символ. Повторите попытку." << endl;
                break;

        }
        break;

case 2:
        cout << "Введите ID элемента, который хотите удалить: ";
        cin >> value;
        cout << endl;

        findEl = 0, number = -1;
        for (const auto& element : vector)
        {
                if (element->get_numb() == value)
                {
                        number++;
                        findEl = 1;
                        break;
                }
                else
                        number++;
        }
```

```
            if (findEl)
            {
                it = vector.begin();
                advance(it, number);
                vector.erase(it);

                cout << "Удаление выполнено." << endl;
            }
            else
                cout << "Элемент не найден." << endl;

            break;

        case 3:
            cout << "Выберите программу, которую хотите добавить:" << endl;
            cout << "1. Элемент класса Student" << endl;
            cout << "2. Элемент класса Course" << endl;
            cout << "========================================" << endl;
            cout << "Ваш выбор: ";
            cin >> value;

            try
            {
                vector.at(value);
```

```
                if (value == 1 || value == 2)
                {
                        vector.emplace_back(newProgram(value));
                        cout << "Элемент добавлен." << endl;
                }
                else
                        cout << "Ошибка. Неверный номер." << endl;
        }
        catch (const std::exception & ex)
        {
                cout << ex.what() << endl;
        }

        break;
case 4:

        for (size_t j = 0; j < 4; j++)
        {
                if (j == 0)
                        mergeVector.emplace_back(new Student(20, 1, 10, "Dani", 0, 0));
                else if (j == 1)
                        mergeVector.emplace_back(new Student(19, 2, 5, "Devid", 1, 50));
                else if (j == 2)
```

```
                mergeVector.emplace_back(new Course(18, 3, 8, "Chack", 1, 10, 2));
            else if (j == 3)
                mergeVector.emplace_back(new Course(18, 4, 9, "Clark", 0, 0, 2));
        }

        cout << "Контейнер, с которым будет объединение:" << endl << endl;
        cout << setw(10) << "Возраст" << setw(8) << "Номер";
        cout << setw(15) << "Средний балл" << setw(7) << "Имя";
        cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
        cout << setw(7) << "Курс" << endl;

        number = 1;
        for_each(mergeVector.begin(), mergeVector.end(), [&number](const
shared_ptr<Student>& program)
        {
            cout << number << ". " << *program << endl;
            number++;
        });
        number = 1;

        vector = CombineVectors(vector, mergeVector);
        mergeVector.erase(mergeVector.begin(), mergeVector.end());

        break;
```

```
case 5:
		cout << "Сортировать по: " << endl;
		cout << "1) Возрастанию" << endl;
		cout << "2) Убыванию" << endl;
		cout << "3) Вернуться назад" << endl;
		cout << "==================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise2;
		cout << endl;

		if (choise2 == 1 || choise2 == 2)
		{
			Functor funct(choise2);

			sort(vector.begin(), vector.end(), funct);

			cout << "Вектор отсортирован." << endl;
		}
		else if (choise2 == 3)
			cout << "Возвращение назад." << endl;
		else
			cout << "Ошибка. Неверная команда." << endl;
```

```
                break;


            case 6:
                cout << "Завершение работы." << endl << endl;
                stop = 0;
                break;


            default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;


            }
        }
}

void ListMenu()
{
        list <shared_ptr<Student>> list;
        stringstream temp;
        string data;
        bool stop = 1, findEl = 0;
        int choise = 0, choise2 = 0, choise3 = 0;
        int value = 0;
        int number = 0;
```

```
int result = 0, sum = 0;

auto it = list.begin();



for (size_t i = 0; i < 4; i++)

{

	if (i == 0)

		list.emplace_back(new Student());

	else if (i == 1)

		list.emplace_back(new Course(20, 1, 10, "Den", 0, 0, 3));

	else if (i == 2)

		list.emplace_back(new Student(18, 2, 8, "Dima", 0, 0));

	else if (i == 3)

		list.emplace_back(new Course(19, 3, 7, "Gordon", 1, 25, 2));

}



while (stop != 0)

{

	if (list.size() == 0)

	{

		cout << "Вектор пуст. Что вы хотите сделать?" << endl;

		cout << "1) Добавить элемент" << endl;

		cout << "2) Завершение работы" << endl;

		cout << "====================================" << endl;

		cout << "Ваш выбор: ";
```

```
cin >> choise;
cout << endl;

switch (choise)
{
case 1:
	cout << "Выберите программу, которую хотите добавить:" << endl;
	cout << "1. Элемент класса Student" << endl;
	cout << "2. Элемент класса Course" << endl;
	cout << "=========================================" <<
endl;
	cout << "Ваш выбор: ";
	cin >> value;

	try
	{
		if (value == 1 || value == 2)
		{
			list.emplace_front(newProgram(value));
			cout << "Элемент добавлен." << endl;
		}
		else
			cout << "Ошибка. Неверный номер." << endl;
	}
```

```
                catch (const std::exception & ex)
                {
                        cout << ex.what() << endl;
                }

                break;

        case 2:
                cout << "Завершение работы." << endl;
                stop = 0;
                break;

        default:
                cout << "Неверный номер элемента. Повторите попытку." << endl;
                break;
        }
}
else
{
        cout << endl;
        cout << "1)Вывод на экран" << endl;
        cout << "2)Удаление элемента" << endl;
        cout << "3)Добавление элементов" << endl;
        cout << "4)Сортировка элементов" << endl;
```

```
        cout << "5)Завершение работы" << endl;
        cout << "======================" << endl;
        cout << "Ваш выбор: ";
        cin >> choise;
        cout << endl;
}

switch (choise)
{
case 1:
        cout << "Выберите команду:" << endl;
        cout << "1) Вывести весь список на экран" << endl;
        cout << "2) Вывести программу по ID" << endl;
        cout << "3) Вывести количество элементов по критерию" << endl;
        cout << "4) Найти элемент по критерию" << endl;
        cout << "5) Вернуться к выбору действий" << endl;
        cout << "====================================" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;

        switch (choise2)
        {
        case 1:
```

```
                cout << setw(10) << "Возраст" << setw(8) << "Номер";
                cout << setw(15) << "Средний балл" << setw(7) << "Имя";
                cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
                cout << setw(7) << "Курс" << endl;

                number = 1;
                for_each(list.begin(), list.end(), [&number](const shared_ptr<Student>&
program)
                {
                        cout << number << ". " << *program << endl;
                        number++;
                });
                number = 1;
                break;

        case 2:
                cout << "Введите id элемента, которого вы хотите получить: ";
                cin >> value;
                cout << endl;

                findEl = 0, number = -1;
                for (const auto& element : list)
                {
                        if (element->get_numb() == value)
```

```
            {
                number++;
                findEl = 1;
                break;
            }
            else
                number++;
        }

        if (findEl)
        {
            it = list.begin();
            advance(it, number);

            temp = (*it)->get_str();
            data = temp.str();

            cout << "Ваш элемент: " << endl;
            cout << data << endl << endl;
        }
        else
            cout << "Элемент с таким ID не найден." << endl;

        break;
```

```
case 3:
        cout << "Выберите критерий, по которому надо искать: " << endl;
        cout << "1) Имя" << endl;
        cout << "2) Возраст" << endl;
        cout << "3) Средний балл" << endl;
        cout << "4) Долг по прог." << endl;
        cout << "5) Номер" << endl;
        cout << "6) Есть ли долг" << endl;
        cout << "7) Вернуться назад" << endl;
        cout << "=====================================" << endl;
        cout << "Ваш выбор: ";
        cin >> choise3;
        cout << endl;

        if (choise3 < 1 || choise3 >= 7)
        {
                cout << "Возвращение назад." << endl;
                break;
        }

        it = list.begin();
        result = 0, sum = 0;
        cout << "Введите критерий: ";
```

```
                cin.ignore();
                getline(cin, data);
                number = 0, value = 0;

                while (number < list.size())
                {
                        result = (*it)->countElement(choise3, data);
                        number++;
                        it++;
                        sum += result;
                }
                if (sum != 0)
                        cout << "Количество элементов с данным параметром: " << sum <<
endl;

                break;

        case 4:
                cout << "Выберите критерий, по которому надо искать: " << endl;
                cout << "1) Имя" << endl;
                cout << "2) Возраст" << endl;
                cout << "3) Средний балл" << endl;
                cout << "4) Долг по прог." << endl;
                cout << "5) Номер" << endl;
```

```
cout << "6) Есть ли долг" << endl;

cout << "7) Вернуться назад" << endl;

cout << "=====================================" << endl;

cout << "Ваш выбор: ";

cin >> choise3;

cout << endl;



if (choise3 < 1 || choise3 >= 7)

{

	cout << "Возвращение назад." << endl;

	break;

}



it = list.begin();

cout << "Введите критерий: ";

cin.ignore();

getline(cin, data);

number = 0, value = 0;



while (number < list.size())

{

	result = (*it)->elementOutput(choise3, data);

	number++;

	it++;
```

```
                }


                break;
            case 5:
                cout << "Возвращение назад." << endl;
                break;


            default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;


            }
            break;


    case 2:
            cout << "Введите ID элемента, который хотите удалить: ";
            cin >> value;
            cout << endl;


            findEl = 0, number = -1;
            for (const auto& element : list)
            {
                if (element->get_numb() == value)
                {
```

```
                        number++;
                        findEl = 1;
                        break;
                }
                else
                        number++;
        }

        if (findEl)
        {
                it = list.begin();
                advance(it, number);
                list.erase(it);

                cout << "Удаление выполнено." << endl;
        }
        else
                cout << "Элемент не найден." << endl;

        break;

case 3:
        cout << "Выберите программу, которую хотите добавить:" << endl;
        cout << "1. Элемент класса Student" << endl;
```

```
		cout << "2. Элемент класса Course" << endl;
		cout << "=========================================" << endl;
		cout << "Ваш выбор: ";
		cin >> value;

		try
		{
			if (value == 1 || value == 2)
			{
				list.emplace_front(newProgram(value));
				cout << "Элемент добавлен." << endl;
			}
			else
				cout << "Ошибка. Неверный номер." << endl;
		}
		catch (const std::exception & ex)
		{
			cout << ex.what() << endl;
		}

		break;
case 4:
		cout << "Сортировать по: " << endl;
		cout << "1) Возрастанию" << endl;
```

```
            cout << "2) Убыванию" << endl;
            cout << "3) Вернуться назад" << endl;
            cout << "==================" << endl;
            cout << "Ваш выбор: ";
            cin >> choise2;
            cout << endl;

            if (choise2 == 1 || choise2 == 2)
            {
                  Functor funct(choise2);

                  list.sort(funct);

                  cout << "Список отсортирован." << endl;
            }
            else if (choise2 == 3)
                  cout << "Возвращение назад." << endl;
            else
                  cout << "Ошибка. Неверная команда." << endl;

            break;

case 5:
            cout << "Завершение работы." << endl << endl;
```

```
                stop = 0;
                break;

            default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;

            }
        }
}

void MapMenu()
{
        map <int, shared_ptr<Student>> map;
        stringstream temp;
        string data;
        bool stop = 1, findEl = 0;
        int choise = 0, choise2 = 0, choise3 = 0;
        int value = 0;
        int i = 0;
        int number = 0, sum = 0, result = 0;
        auto it = map.begin();

        for (; i < 4; i++)
```

```
{
	if (i == 0)
		map.emplace(i + 1, new Student());
	else if (i == 1)
		map.emplace(i + 1, new Course(20, 1, 10, "Den", 0, 0, 3));
	else if (i == 2)
		map.emplace(i + 1, new Student(18, 2, 8, "Dima", 0, 0));
	else if (i == 3)
		map.emplace(i + 1, new Course(19, 3, 7, "Gordon", 1, 25, 2));
}

while (stop != 0)
{
	if (map.size() == 0)
	{
		cout << "Вектор пуст. Что вы хотите сделать?" << endl;
		cout << "1) Добавить элемент" << endl;
		cout << "2) Завершение работы" << endl;
		cout << "=====================================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise;
		cout << endl;

		switch (choise)
```

```
{
case 1:
        cout << "Выберите программу, которую хотите добавить:" << endl;
        cout << "1. Элемент класса Student" << endl;
        cout << "2. Элемент класса Course" << endl;
        cout << "=========================================" <<
endl;
        cout << "Ваш выбор: ";
        cin >> value;

        try
        {
            if (value == 1 || value == 2)
            {
                map.emplace(++i, newProgram(value));
                cout << "Элемент добавлен." << endl;
            }
            else
                cout << "Ошибка. Неверный номер." << endl;
        }
        catch (const std::exception & ex)
        {
            cout << ex.what() << endl;
        }
```

```
                break;


        case 2:
                cout << "Завершение работы." << endl;
                stop = 0;
                break;


        default:
                cout << "Неверный номер элемента. Повторите попытку." << endl;
                break;
        }
}
else
{
        cout << endl;
        cout << "1)Вывод на экран" << endl;
        cout << "2)Удаление элемента" << endl;
        cout << "3)Добавление элементов" << endl;
        cout << "4)Сортировать контейнеры" << endl;
        cout << "5)Объеденить контейнеры" << endl;
        cout << "6)Завершение работы" << endl;
        cout << "====================" << endl;
        cout << "Ваш выбор: ";
```

```
		cin >> choise;
		cout << endl;
}

switch (choise)
{
case 1:
		cout << "Выберите команду:" << endl;
		cout << "1) Вывести весь список на экран" << endl;
		cout << "2) Вывести программу по ID" << endl;
		cout << "3) Вывести количество элементов по критерию" << endl;
		cout << "4) Найти элемент по критерию" << endl;
		cout << "5) Вернуться к выбору действий" << endl;
		cout << "====================================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise2;
		cout << endl;

		switch (choise2)
		{
		case 1:
			cout << setw(10) << "Возраст" << setw(8) << "Номер";
			cout << setw(15) << "Средний балл" << setw(7) << "Имя";
			cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
```

```
                cout << setw(7) << "Курс" << endl;


                for_each(map.begin(), map.end(), [](const std::pair<const int,
shared_ptr<Student>>& program)
                {
                        cout << program.first << ". " << *program.second << endl;
                });


                break;


        case 2:
                cout << "Введите номер элемента, которого вы хотите получить: ";
                cin >> value;
                cout << endl;


                findEl = 0;
                it = map.find(value);


                if (it != map.end())
                {
                        temp = (*it).second->get_str();
                        data = temp.str();


                        cout << "Ваш элемент: " << endl;
```

```
                cout << data << endl << endl;
            }
            else
                cout << "Элемент с таким ID не найден." << endl;

            break;

        case 3:
            cout << "Выберите критерий, по которому надо искать: " << endl;
            cout << "1) Имя" << endl;
            cout << "2) Возраст" << endl;
            cout << "3) Средний балл" << endl;
            cout << "4) Долг по прог." << endl;
            cout << "5) Номер" << endl;
            cout << "6) Есть ли долг" << endl;
            cout << "7) Вернуться назад" << endl;
            cout << "=====================================" << endl;
            cout << "Ваш выбор: ";
            cin >> choise3;
            cout << endl;

            if (choise3 < 1 || choise3 >= 7)
            {
                cout << "Возвращение назад." << endl;
```

```
                break;
            }

            it = map.begin();
            result = 0, sum = 0;
            cout << "Введите критерий: ";
            cin.ignore();
            getline(cin, data);
            number = 0, value = 0;

            while (number < map.size())
            {
                result = it->second->countElement(choise3, data);
                number++;
                it++;
                sum += result;
            }
            if (sum != 0)
                cout << "Количество элементов с данным параметром: " << sum <<
endl;

            break;

        case 4:
```

```
cout << "Выберите критерий, по которому надо искать: " << endl;
cout << "1) Имя" << endl;
cout << "2) Возраст" << endl;
cout << "3) Средний балл" << endl;
cout << "4) Долг по прог." << endl;
cout << "5) Номер" << endl;
cout << "6) Есть ли долг" << endl;
cout << "7) Вернуться назад" << endl;
cout << "=====================================" << endl;
cout << "Ваш выбор: ";
cin >> choise3;
cout << endl;

if (choise3 < 1 || choise3 >= 7)
{
	cout << "Возвращение назад." << endl;
	break;
}

it = map.begin();
cout << "Введите критерий: ";
cin.ignore();
getline(cin, data);
number = 0, value = 0;
```

```
                while (number < map.size())
                {
                        result = it->second->elementOutput(choise3, data);
                        number++;
                        it++;
                }

                break;
        case 5:
                cout << "Возвращение назад." << endl;
                break;

        default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;

        }
        break;

case 2:
        cout << "Введите номер элемента, который хотите удалить: ";
        cin >> value;
        cout << endl;
```

```
            findEl = 0;
            it = map.find(value);

            if (it != map.end())
            {
                map.erase(it);
                cout << "Удаление выполнено." << endl;
            }
            else
                cout << "Элемент не найден." << endl;

            break;

case 3:
            cout << "Выберите программу, которую хотите добавить:" << endl;
            cout << "1. Элемент класса Student" << endl;
            cout << "2. Элемент класса Course" << endl;
            cout << "=========================================" << endl;
            cout << "Ваш выбор: ";
            cin >> value;

            try
            {
```

```
                if (value == 1 || value == 2)
                {
                        map.emplace(++i, newProgram(value));
                        cout << "Элемент добавлен." << endl;
                }
                else
                        cout << "Ошибка. Неверный номер." << endl;
        }
        catch (const std::exception & ex)
        {
                cout << ex.what() << endl;
        }

        break;
case 4:
        cout << "Сортировать по: " << endl;
        cout << "1) Возрастанию" << endl;
        cout << "2) Убыванию" << endl;
        cout << "3) Вернуться назад" << endl;
        cout << "==================" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;
```

```
if (choise2 == 1 || choise2 == 2)
{
	value = 1;
	Functor funct(choise2);
	vector <shared_ptr<Student>> temp;

	for (auto const& it : map)
		temp.push_back(it.second);

	sort(temp.begin(), temp.end(), funct);

	map.erase(map.begin(), map.end());

	transform(temp.begin(), temp.end(), inserter(map, map.end()), [&value](const shared_ptr<Student>& a)
	{
		return make_pair(value++, a);
	});

}
else if (choise2 == 3)
	cout << "Возвращение." << endl;
else
	cout << "Ошибка. Неверный номер элемента." << endl;
```

```
            break;


case 5:
{
            std::map <int, shared_ptr<Student>> mergeMap;
            std::map <int, shared_ptr<Student>> result;
            for (size_t j = 0; j < 4; j++)
            {
                        if (j == 0)
                                    mergeMap.emplace(j + 1, new Student(20, 1, 10, "Dani", 0, 0));
                        else if (j == 1)
                                    mergeMap.emplace(j + 1, new Student(19, 2, 5, "Devid", 1, 50));
                        else if (j == 2)
                                    mergeMap.emplace(j + 1, new Course(18, 3, 8, "Chack", 1, 10, 2));
                        else if (j == 3)
                                    mergeMap.emplace(j + 1, new Course(18, 4, 9, "Clark", 0, 0, 2));
            }

            cout << "Контейнер, с которым будет объединение:" << endl << endl;
            cout << setw(10) << "Возраст" << setw(8) << "Номер";
            cout << setw(15) << "Средний балл" << setw(7) << "Имя";
            cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
            cout << setw(7) << "Курс" << endl;
```

```
                    for_each(mergeMap.begin(), mergeMap.end(), [](const pair<const int,
shared_ptr<Student>>& program)
                    {
                        cout << program.first << ". " << *program.second << endl;
                    });

                    map = CombineMaps(map, mergeMap);

                    cout << "Объединение выполнено." << endl;

                    break;
                }
                case 6:
                    cout << "Завершение работы." << endl << endl;
                    stop = 0;
                    break;

                default:
                    cout << "Неверный символ. Повторите попытку." << endl;
                    break;

                }
            }
}
```

```
void SetMenu()
{
	set <shared_ptr<Student>> set;
	stringstream ss;
	string data;
	bool stop = 1, findEl = 0;
	int choise = 0, choise2 = 0, choise3 = 0;
	int value = 0, number = 0, result = 0, sum = 0;
	auto it = set.begin();

	for (size_t i = 0; i < 4; i++)
	{
		if (i == 0)
			set.emplace(new Student());
		else if (i == 1)
			set.emplace(new Course(20, 1, 10, "Den", 0, 0, 3));
		else if (i == 2)
			set.emplace(new Student(18, 2, 8, "Dima", 0, 0));
		else if (i == 3)
			set.emplace(new Course(19, 3, 7, "Gordon", 1, 25, 2));
	}

	while (stop != 0)
```

```
{
	if (set.size() == 0)
	{
		cout << "Вектор пуст. Что вы хотите сделать?" << endl;
		cout << "1) Добавить элемент" << endl;
		cout << "2) Завершение работы" << endl;
		cout << "=====================================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise;
		cout << endl;

		switch (choise)
		{
		case 1:
			cout << "Выберите программу, которую хотите добавить:" << endl;
			cout << "1. Элемент класса Student" << endl;
			cout << "2. Элемент класса Course" << endl;
			cout << "=========================================" <<
endl;
			cout << "Ваш выбор: ";
			cin >> value;

			try
			{
```

```
                if (value == 1 || value == 2)
                {
                        set.emplace(newProgram(value));
                        cout << "Элемент добавлен." << endl;
                }
                else
                        cout << "Ошибка. Неверный номер." << endl;
        }
        catch (const std::exception & ex)
        {
                cout << ex.what() << endl;
        }

        break;

case 2:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;

default:
        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
}
```

```
}
else
{
	cout << endl;
	cout << "1)Вывод на экран" << endl;
	cout << "2)Удаление элемента" << endl;
	cout << "3)Добавление элементов" << endl;
	cout << "4)Сортировка элементов" << endl;
	cout << "5)Завершение работы" << endl;
	cout << "======================" << endl;
	cout << "Ваш выбор: ";
	cin >> choise;
	cout << endl;
}

switch (choise)
{
case 1:
	cout << "Выберите команду:" << endl;
	cout << "1) Вывести весь список на экран" << endl;
	cout << "2) Вывести программу по ID" << endl;
	cout << "3) Вывести количество элементов по критерию" << endl;
	cout << "4) Найти элемент по критерию" << endl;
	cout << "5) Вернуться к выбору действий" << endl;
```

```
cout << "====================================" << endl;
cout << "Ваш выбор: ";
cin >> choise2;
cout << endl;

switch (choise2)
{
case 1:
	cout << setw(10) << "Возраст" << setw(8) << "Номер";
	cout << setw(15) << "Средний балл" << setw(7) << "Имя";
	cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
	cout << setw(7) << "Курс" << endl;

	number = 1;
	for_each(set.begin(), set.end(), [&number](const shared_ptr<Student>&
program)
	{
		cout << number << ". " << *program << endl;
		number++;
	});
	number = 1;
	break;

case 2:
```

```
cout << "Введите id элемента, которого вы хотите получить: ";
cin >> value;
cout << endl;

findEl = 0, number = -1;
for (const auto& element : set)
{
	if (element->get_numb() == value)
	{
		number++;
		findEl = 1;
		break;
	}
	else
		number++;
}

if (findEl)
{
	it = set.begin();
	advance(it, number);

	ss = (*it)->get_str();
	data = ss.str();
```

```
                cout << "Ваш элемент: " << endl;
                cout << data << endl << endl;
            }
            else
                cout << "Элемент с таким ID не найден." << endl;

            break;

case 3:
            cout << "Выберите критерий, по которому надо искать: " << endl;
            cout << "1) Имя" << endl;
            cout << "2) Возраст" << endl;
            cout << "3) Средний балл" << endl;
            cout << "4) Долг по прог." << endl;
            cout << "5) Номер" << endl;
            cout << "6) Есть ли долг" << endl;
            cout << "7) Вернуться назад" << endl;
            cout << "=====================================" << endl;
            cout << "Ваш выбор: ";
            cin >> choise3;
            cout << endl;

            if (choise3 < 1 || choise3 >= 7)
```

```
{
	cout << "Возвращение назад." << endl;
	break;
}

it = set.begin();
result = 0, sum = 0;
cout << "Введите критерий: ";
cin.ignore();
getline(cin, data);
number = 0, value = 0;

while (number < set.size())
{
	result = (*it)->countElement(choise3, data);
	number++;
	it++;
	sum += result;
}
if (sum != 0)
	cout << "Количество элементов с данным параметром: " << sum <<
endl;

break;
```

```
case 4:
		cout << "Выберите критерий, по которому надо искать: " << endl;
		cout << "1) Имя" << endl;
		cout << "2) Возраст" << endl;
		cout << "3) Средний балл" << endl;
		cout << "4) Долг по прог." << endl;
		cout << "5) Номер" << endl;
		cout << "6) Есть ли долг" << endl;
		cout << "7) Вернуться назад" << endl;
		cout << "=====================================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise3;
		cout << endl;

		if (choise3 < 1 || choise3 >= 7)
		{
			cout << "Возвращение назад." << endl;
			break;
		}

		it = set.begin();
		cout << "Введите критерий: ";
		cin.ignore();
```

```
                getline(cin, data);
                number = 0, value = 0;

                while (number < set.size())
                {
                        result = (*it)->elementOutput(choise3, data);
                        number++;
                        it++;
                }

                break;
        case 5:
                cout << "Возвращение назад." << endl;
                break;

        default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;

        }
        break;

case 2:
        cout << "Введите ID элемента, который хотите удалить: ";
```

```
cin >> value;
cout << endl;

findEl = 0, number = -1;
for (const auto& element : set)
{
	if (element->get_numb() == value)
	{
		number++;
		findEl = 1;
		break;
	}
	else
		number++;
}

if (findEl)
{
	it = set.begin();
	advance(it, number);
	set.erase(it);

	cout << "Удаление выполнено." << endl;
}
```

```
            else
                cout << "Элемент не найден." << endl;

            break;

        case 3:
            cout << "Выберите программу, которую хотите добавить:" << endl;
            cout << "1. Элемент класса Student" << endl;
            cout << "2. Элемент класса Course" << endl;
            cout << "=========================================" << endl;
            cout << "Ваш выбор: ";
            cin >> value;

            try
            {
                if (value == 1 || value == 2)
                {
                    set.emplace(newProgram(value));
                    cout << "Элемент добавлен." << endl;
                }
                else
                    cout << "Ошибка. Неверный номер." << endl;
            }
            catch (const std::exception & ex)
```

```
            {
                cout << ex.what() << endl;
            }

            break;
        case 4:
            cout << "Сортировать по: " << endl;
            cout << "1) Возрастанию" << endl;
            cout << "2) Убыванию" << endl;
            cout << "3) Вернуться назад" << endl;
            cout << "==================" << endl;
            cout << "Ваш выбор: ";
            cin >> choise2;
            cout << endl;

            if (choise2 == 1 || choise2 == 2)
            {
                vector <shared_ptr<Student>> temp(set.begin(), set.end());
                set.erase(set.begin(), set.end());

                Functor funct(choise2);
                sort(temp.begin(), temp.end(), funct);

                set.insert(temp.begin(), temp.end());
```

```
                cout << "Отсортированный set" << endl;
                cout << setw(10) << "Возраст" << setw(8) << "Номер";
                cout << setw(15) << "Средний балл" << setw(7) << "Имя";
                cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
                cout << setw(7) << "Курс" << endl;

                number = 1;
                for_each(temp.begin(), temp.end(), [&number](const shared_ptr<Student>&
program)
                {
                        cout << number << ". " << *program << endl;
                        number++;
                });
                number = 1;
                temp.erase(temp.begin(), temp.end());
        }
        else if (choise2 == 3)
                cout << "Возвращение." << endl;
        else
                cout << "Ошибка. Неверный номер элемента." << endl;

        break;
```

```
                case 5:
                        cout << "Завершение работы." << endl << endl;
                        stop = 0;
                        break;

                default:
                        cout << "Неверный символ. Повторите попытку." << endl;
                        break;

                }
        }
}
```

test.cpp

```
#include "Header.h"

void VectorTest();
void ListTest();
void MapTest();
void SetTest();

int main()
{
        setlocale(LC_ALL, "ru");

        VectorTest();
        ListTest();
        MapTest();
        SetTest();

        if (_CrtDumpMemoryLeaks())
                cout << endl << "Обнаружена утечка памяти!" << endl;
        else
                cout << endl << "Утечки не обнаруженно!" << endl;

        system("PAUSE");
        return 0;
}
void VectorTest()
{
        vector<int> vector = { 1, -5, 20, 555, 0 };
```

```
	int vectorSize = vector.size();
	int newVectorSize;
	int value;
	std::vector<int>::iterator it;
	cout << "Vector" << endl;

	vector.push_back(155);
	newVectorSize = vector.size();
	if (vectorSize != newVectorSize && vector[newVectorSize - 1] == 155)
		cout << "Тест добавления элемента\tвыполнен успешно.\n";
	else
		cout << "Тест добавления элемента\tне выполнен успешно.\n";

	it = vector.begin();
	value = vector[2];
	vector.erase(it + 2);
	newVectorSize = vector.size();
	if (vectorSize == newVectorSize && vector[2] != value)
		cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

	if (vector[0] == 1)
		cout << "Тест получения элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест получения элемента\t\tне выполнен успешно.\n";
}
void ListTest()
{
	list<int> list = { 1, -5, 20, 555, 0 };
	int listSize = list.size();
	int value;
	std::list<int>::iterator it;
	std::list<int>::iterator it2;
	cout << endl << "List" << endl;

	list.push_back(155);
	list.push_front(228);
	it = list.begin();
	it2 = list.begin();
	std::advance(it2, list.size() - 1);
	if (listSize != list.size() && *it == 228 && *it2 == 155)
		cout << "Тест добавления элемента\tвыполнен успешно.\n";
	else
		cout << "Тест добавления элемента\tне выполнен успешно.\n";

	it2 = list.begin();
	std::advance(it2, 2);
	list.erase(it2);
	it = list.begin();
	std::advance(it, 2);
	if (list.size() == listSize + 1 && it != it2)
		cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

	if (*it == 20)
		cout << "Тест получения элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест получения элемента\t\tне выполнен успешно.\n";
}
void SetTest()
{
	set<int> set = { 1, -5, 20, 555, 0 };
	int setSize = set.size();
```

```
	int value;
	std::set<int>::iterator it;
	std::set<int>::iterator it2;
	cout << endl << "Set" << endl;

	set.insert(155);
	it2 = set.begin();
	std::advance(it2, 4);
	if (setSize != set.size() && *it2 == 155)
		cout << "Тест добавления элемента\tвыполнен успешно.\n";
	else
		cout << "Тест добавления элемента\tне выполнен успешно.\n";

	it2 = set.begin();
	set.erase(it2);
	it = set.begin();
	if (set.size() == setSize && it != it2 && *it == 0)
		cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

	if (*it == 0)
		cout << "Тест получения элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест получения элемента\t\tне выполнен успешно.\n";
}
void MapTest()
{
	map <int, int> map = { {1, 1}, {-5,2}, {20, 3}, {555, 4}, {0, 5} };
	int mapSize = map.size();
	std::map<int, int>::iterator it;
	std::map<int, int>::iterator it2;
	cout << endl << "Map" << endl;

	map.insert(std::pair<int, int>(155, 6));
	if (mapSize < map.size())
		cout << "Тест добавления элемента\tвыполнен успешно.\n";
	else
		cout << "Тест добавления элемента\tне выполнен успешно.\n";

	it = map.begin();
	map.erase(it);
	if (mapSize == map.size())
		cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

	it = map.begin();
	if (map.find(0) == it)
		cout << "Тест получения элемента\t\tвыполнен успешно.\n";
	else
		cout << "Тест получения элемента\t\tне выполнен успешно.\n";
}
```

4 Результати тестування

```
Выберите STL контейнер:
1) Vector;
2) List;
3) Map;
4) Set;
5) Выход;
======================
Пункт: 1

1)Вывод на экран
2)Удаление
3)Добавление
4)Объединить векторы
5)Сортировка
6)Завершение работы
======================
Ваш выбор: 1

Выберите команду:
1) Вывести весь список на экран
2) Вывести программу по ID
3) Вывести количество элементов по критерию
4) Найти элемент по критерию
5) Вернуться к выбору действий
====================================
Ваш выбор: 1

   Возраст   Номер   Средний балл    Имя      Долг   Долг(прог.)   Курс
1. 17        0       8               Bond     1      15
2. 20        1       10              Den      0      0             3
3. 18        2       8               Dima     0      0
4. 19        3       7               Gordon   1      25            2

1)Вывод на экран
2)Удаление
3)Добавление
4)Объединить векторы
5)Сортировка
6)Завершение работы
======================
Ваш выбор:
```

5 Опис результатів

На практиці порівняли STL-алгоритми, що модифікують послідовність; отримали навички роботи з STL-функторами. Було створенно меню за варіантами вибору контейнеру. Реалізовані методи роботи з контейнером.