Автор: Татаренко А., КІТ-119а

Дата: 17 червня 2020

Лабораторна робота №8.

ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

Тема. Перевантаження операторів. Серіалізація.

Мета – отримати знання про призначення операторів, визначити їх ролі у житті об’єкта та можливість перевизначення.

1 Завдання до роботи

Індивідуальне завдання 19.

Поширити попередню лабораторну роботу таким чином:
- у базовому класі, та класі/класах-спадкоємцях перевантажити:
- оператор присвоювання;
- оператор порівняння (на вибір: == , < , > , >= , <= , !=);
- оператор введення / виведення;
- у класі-списку перевантажити:
- оператор індексування ([ ]);
- оператор введення / виведення з акцентом роботи, у тому числі і з файлами.
При цьому продовжувати використовувати регулярні вирази для валідації введених даних.

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

audience.h

```
#pragma once
#include <string>

using std::string;
using std::ostream;
using std::istream;

typedef short sint;

class Aud
{
private:
	sint aud_numb;

public:
	sint get_aud_numb() const;
	void set_aud_numb(sint);

	Aud();
	Aud(sint);
	Aud(const Aud& other);
	~Aud();
//////////////////////////////////////////////////
	friend istream& operator>> (istream&, Aud&);
	friend ostream& operator<< (ostream&, const Aud&);
	bool operator==(const sint) const;
};
```

basic_class.h

```
#pragma once

#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>

#include <string>
#include <iostream>
#include <iomanip>
#include <locale>
#include <fstream>
#include <sstream>
#include <regex>

#include "audience.h"
#include "faculty.h"

#define El 3

using std::cout;
using std::cin;
using std::endl;
using std::stringstream;
using std::ofstream;
using std::ifstream;
using std::regex;
using std::ostream;
using std::istream;
```

```
using std::ios;

typedef bool (comp)(const int&, const int&);

class Student
{
protected:

	int age;
	int number_stud;
	int middle_mark;
	string name;
	bool debt;
	int prog_d;
	Aud audience;
	Fac faculty;

public:

	virtual int get_number_stud() const;

	virtual int get_age() const;

	virtual int get_middle_mark() const;

	virtual string get_name() const;

	virtual bool get_debt() const;

	virtual int get_prog_d() const;

	virtual sint get_aud() const;

	virtual string get_fac() const;

	virtual void print() const = 0;
	virtual stringstream get_str() const = 0;
	virtual void write_in_file(ofstream&) = 0;

	Student();

	Student(int a, int n, int m, string na, bool d, int pd, sint ad, string fc);

	Student(const Student &other);

	~Student()
	{
		cout << "Вызвался деструктор!" << endl;
	}

	virtual string getInfo() const = 0;
	virtual void input(istream&) = 0;
	virtual stringstream getStr() const = 0;
	//////////////////////////////////////////////////////////
	friend ofstream& operator<< (ofstream&, const Student&);
	friend ostream& operator<< (ostream&, const Student&);
	friend istream& operator>> (istream&, Student&);
	virtual bool operator==(const string) const;
	Student& operator= (Student&);

};
```

class_list.h:

```
#pragma once
```

```
#include "surnames.h"
#include "course.h"

class Arr
{
private:

	Student** array_stud;

	int count = El;

public:

	void create_array();

	void print_array(int) const;

	void delete_one();

	void add();

	string select() const;

	void fill_array();

	int count_plus();

	int count_minus();

	void print_one(Student*) const;

	void delete_array();

	Student* Construct(int);

	int prog_d_rand(bool);

	void find_debt();

	void in_f();

	void from_f(int);

	int get_count() const;

	int str_in_file(string) const;

	void set_count(int);

	void regex_task();

	void sort(comp);

	static bool sort_forward(const int&, const int&);

	static bool sort_back(const int&, const int&);
	//////////////////////////////////////////
	Student* operator[] (int);
};
```

faculty.h

```
#pragma once
#include <string>
```

```
using std::string;
using std::ostream;
using std::istream;

class Fac
{
private:
	string fname;

public:
	string get_fname() const;
	void set_fname(string);

	Fac();
	Fac(string);
	Fac(const Fac& other);
	~Fac();
	////////////////////////////////////////////
	friend istream& operator>> (istream&, Fac&);
	friend ostream& operator<< (ostream&, const Fac&);
};
```

course.h

```
#pragma once
#include "basic_class.h"
#include <string>

using std::string;
using std::ostream;
using std::istream;

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

public:
	int get_course();
	void set_course(int);

	void print() const override;
	stringstream get_str() const override;
	void write_in_file(ofstream& el) override;

	Course();
	Course(int, int, int, string, bool, int, sint, string, int);
	Course(const Course& other);
	~Course();
	////////////////////////////////////////////////////
	string getInfo() const override final;
	stringstream getStr() const override final;
	void input(istream&) override final;

	Course& operator= (Course&);
	bool operator==(const string) const override final;
};
```

menu.h:

```
#pragma once
```

```
#include "class_list.h"

void menu(Arr);
```

surname.h

```
#pragma once
#include "basic_class.h"
#include <string>

using std::string;
using std::ostream;
using std::istream;

class Surname final : public Student
{
private:
	string sur_star;
	string sur_cur;

public:
	string get_sur_star() const;
	string get_sur_cur() const;

	void set_sur_star(string);
	void set_sur_cur(string);

	void print() const override;
	stringstream get_str() const override;
	void write_in_file(ofstream& el) override;

	Surname();
	Surname(int, int, int, string, bool, int, sint, string, string, string);
	Surname(const Surname& other);
	~Surname();
	////////////////////////////////////////
	string getInfo() const override final;
	stringstream getStr() const override final;
	void input(istream&) override final;

	Surname& operator= (Surname&);
	bool operator==(const string) const override final;
};
```

test.h:

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

bool test_count_plus(Arr);
bool test_count_minus(Arr);
```

audience.cpp

```
#include "audience.h"

sint Aud::get_aud_numb() const
{
	return aud_numb;
}
```

```
void Aud::set_aud_numb(sint a_n)
{
	aud_numb = a_n;
}

Aud::Aud() : aud_numb(0)
{
}

Aud::Aud(sint aud_numb) : aud_numb(aud_numb)
{
}

Aud::Aud(const Aud& other) : aud_numb(other.aud_numb)
{
}

Aud::~Aud()
{
}
////////////////////////////////////////////////
istream& operator>> (istream& input, Aud& author)
{
	input >> author.aud_numb;
	return input;
}
ostream& operator<< (ostream& output, const Aud& author)
{
	output << author.aud_numb;
	return output;
}
bool Aud::operator==(const sint author) const
{
	return this->aud_numb == author;

}
```

basic_class.cpp:

```
#include "basic_class.h"

Student::Student() : age(0), number_stud(0), middle_mark(0), name("Name"), debt(0),
prog_d(0), audience(0), faculty("Non")
{
	cout << "Вызван стандартый конструктор!" << endl;
}

Student::Student(int a, int n, int m, string na, bool d, int pd, sint an, string fc) :
age(a), number_stud(n), middle_mark(m), name(na), debt(d), prog_d(pd), audience(an),
faculty(fc)
{
	cout << "Вызван конструктор с параметрами!" << endl;
}

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),
audience(other.audience), faculty(other.faculty)
{
	cout << "Вызван конструктор копирования!" << endl;
}

int Student::get_number_stud() const
{
	return number_stud;
}
```

```cpp
int Student::get_age() const
{
	return age;
}

int Student::get_middle_mark() const
{
	return middle_mark;
}


string Student::get_name() const
{
	return name;
}

bool Student::get_debt() const
{
	return debt;
}

int Student::get_prog_d() const
{
	return prog_d;
}

sint Student::get_aud() const
{
	return audience.get_aud_numb();
}

string Student::get_fac() const
{
	return faculty.get_fname();
}
////////////////////////////////////////////////////////////////
ofstream& operator<< (ofstream& output, const Student& program)
{
	output << program.getInfo();
	return output;
}

ostream& operator<< (ostream& output, const Student& program)
{
	output << program.getInfo();
	return output;
}

istream& operator>> (istream& input, Student& program)
{
	program.input(input);
	return input;
}

bool Student::operator==(const string name) const
{
	return this->name == name;
}

Student& Student::operator= (Student& temp)
{
	if (this == &temp)
		return *this;
```

```
        int age = temp.age;
        int number_stud = temp.number_stud;
        int middle_mark = temp.middle_mark;
        string name = temp.name;
        bool debt = temp.debt;
        int prog_d = temp.prog_d;
        Aud audience = temp.audience;
        Fac faculty = temp.faculty;

        return *this;
}
```

class_list.cpp:

```
#include "class_list.h"

void Arr::create_array()
{
        int c = get_count();

        array_stud = new Student*[c];
}

void Arr::fill_array()
{
        int c = get_count();

        for (size_t i = 0; i < c; i++)
        {
                *(array_stud + i) = Construct(i);
        }

}

Student* Arr::Construct(int i)
{
        bool d = 0;

        if (i == 0)
        {
                d = rand() % 2;
                Student *st0 = new Surname(i + 17, i + 1, i + 10 / 2, "Dani", d,
prog_d_rand(d), 15, "KIT", "Jhordon", "Pavlov");
                return st0;
        }
        else if (i == 1)
        {
                d = rand() % 2;
                Student *st1 = new Course(i + 17, i + 1, i + 10 / 2, "Peter", d,
prog_d_rand(d), 15, "KIT", 1);
                return st1;
        }
        else if (i == 2)
        {
                d = rand() % 2;
                Student *st2 = new Course(i + 17, i + 1, i + 10 / 2, "Donald Tramp", d,
prog_d_rand(d), 36, "PIT", 2);
                return st2;
        }
        else if (i == 3)
        {
                d = rand() % 2;
```

```
			Student *st2 = new Course(i + 17, i + 1, i + 10 / 2, "Vladimir", d,
prog_d_rand(d), 36, "PIT", 3);
			return st2;
		}

		d = rand() % 2;
		Student *st = new Course(i + 17, i + 1, i + 10 / 2, "Edgar", d, prog_d_rand(d),
17, "GM", 3);
		return st;
}

void Arr::print_array(int c) const
{
		for (size_t i = 0; i < 50; i++)
			cout << "-";
		cout << endl;
		/////////////////
		for (size_t i = 0; i < c; i++)
		{
			array_stud[i]->print();
		}
		/////////////////
		for (size_t i = 0; i < 50; i++)
			cout << "-";
		cout << endl;
}

void Arr::delete_one()
{
		int c = get_count();
		int input, i_arr = 0;
		cout << "Введите номер: ";
		cin >> input;

		for (size_t i = 0; i < c; i++)
			if (input == array_stud[i]->get_number_stud())
			{
				count_minus();
				int c = get_count();

				Student** array_stud_new = new Student *[c];
				Student** s_arr = array_stud;

				for (size_t i = 0; i < input; i++)
					*(array_stud_new + i) = *(s_arr + i);

				for (size_t i = input, j = input + 1; j < c+1; i++, j++)
					*(array_stud_new + i) = *(s_arr + j);

				delete* (array_stud + input);
				array_stud = array_stud_new;
				delete s_arr;

				print_array(c);

				return;
			}
		/////////////////
		for (size_t i = 0; i < 50; i++)
			cout << "-";
		cout << endl;
		/////////////////
		cout << "Этого студента не существует!" << endl;
		/////////////////
		for (size_t i = 0; i < 50; i++)
```

```
                cout << "-";
        cout << endl;
        /////////////////
}

void Arr::add()
{
        count_plus();
        int c = get_count();

        Student** array_stud_new = new Student*[c];
        Student** s_arr = array_stud;

        size_t i = 0;

        for (i; i < c - 1; i++)
        {
                *(array_stud_new + i) = *(s_arr + i);
        }

        array_stud_new[c - 1] = Construct(i);

        array_stud = array_stud_new;

        delete s_arr;

        print_array(c);
}

int Arr::count_plus()
{
        count = count + 1;

        return count;
}

int Arr::count_minus()
{
        count = count - 1;

        return count;
}

string Arr::select() const
{
        int input = 0;
        cout << "Введите номер: ";
        cin >> input;
        int c = get_count();

        for (size_t i = 0; i < c; i++)
        {
                if (input == array_stud[i]->get_number_stud())
                {
                        string info;
                        stringstream ss;
                        ss << "Номер: " << array_stud[i]->get_number_stud() << "\tВозраст: "
<< array_stud[i]->get_age() << "\t\tСредний балл: " << array_stud[i]->get_middle_mark()
<< "\t\tИмя: " << array_stud[i]->get_name() << endl;
                        info = ss.str();

                        return info;
                }
        }
        /////////////////
```

```
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	/////////////////
	cout << "Этого студента не существует!" << endl;
	/////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	/////////////////
}

void Arr::print_one(Student *stud) const
{
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	////////////////////////
	cout << "Номер: " << stud->get_number_stud() << "\tВозраст: " << stud->get_age()
<< "\t\tСредний балл: " << stud->get_middle_mark() << "\t\tИмя: " << stud->get_name() <<
"\t\tАудитория: " << stud->get_aud() << "\t\tФакультет: " << stud->get_fac() << endl;
	////////////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
}

void Arr::delete_array()
{
	int c = get_count();
	for (int i = 0; i < c; i++)
		delete *(array_stud + i);

	delete array_stud;
}

int Arr::prog_d_rand(bool d)
{
	int pd = 0;

	if (d == 1)
		pd = rand() % 50;

	return pd;
}

void Arr::find_debt()
{
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	////////////////////////
	for(size_t i = 0; i < count; i++)
		if (array_stud[i]->get_debt() == 1)
		{
			cout << "Номер: " << array_stud[i]->get_number_stud() << "\tВозраст:
" << array_stud[i]->get_age() << "\t\tСредний балл: " << array_stud[i]->get_middle_mark()
<< "\t\tИмя: " << array_stud[i]->get_name() << "\t\tДолг: Есть" << "\t\tДолг по
программированию: " << array_stud[i]->get_prog_d() << "%" << endl;
		}
	////////////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
```

```
}

void Arr::in_f()
{
	ofstream file;
	file.open("of.txt", ofstream::app);

	if (!file.is_open())
	{
		cout << "Файл не открыт!" << endl;
		return;
	}

	int c = get_count();

	for (size_t i = 0; i < c; i++)
	{
		array_stud[i]->write_in_file(file);
	}

	file.close();

	////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	////////////////
	cout << "Файл записан успешно!" << endl;
	////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	////////////////

}

void Arr::from_f(int size)
{
	string s;
	regex varEn("([\\d]* [\\d]* [\\d]* [A-Z]+[\\w,.;:-]* [0|1] [\\d]* [\\d]* [A-Z]*
[\\d]*)");

	ifstream file;
	file.open("if.txt");

	if (!file.is_open())
	{
		cout << "Файл не открыт!" << endl;
		return;
	}

	delete[] array_stud;
	array_stud = new Student*[size];

	for (size_t i = 0; i < size; i++)
	{
		getline(file, s);
		if (regex_match(s, varEn))
		{
			std::istringstream iss(s);
			int age;
			int number;
			int middle_mark;
			string name;
			bool debt;
```

```
                        int prog_d;
                        sint audience;
                        string faculty;
                        int course;

                        iss >> age;
                        iss >> number;
                        iss >> middle_mark;
                        iss >> name;
                        iss >> debt;
                        iss >> prog_d;
                        iss >> audience;
                        iss >> faculty;
                        iss >> course;

                        array_stud[i] = new Course(age, number, middle_mark, name, debt,
prog_d, audience, faculty, course);
                }
        }

        file.close();

        cout << "Чтение с файла успешно!" << endl << endl;

        set_count(size);

        print_array(size);

}

int Arr::get_count() const
{
        return count;

}

int Arr::str_in_file(string fileName) const
{
        int size = 0;
        int c = get_count();
        string line;
        ifstream file(fileName);

        if (!file.is_open() || c == 0)
        {
                cout << "There is no such file" << endl << endl;
                return 0;
        }

        while (getline(file, line))
        {
                size++;
        }
        file.close();

        return size;
}

void Arr::set_count(int c)
{
        count = c;

}

void Arr::regex_task()
```

```
{
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
	////////////////
	regex regular("(^[A-ZÀ-ß]+[\\wÀ-ßà-ÿ,.;:-]* [\\wÀ-ßà-ÿ,.;:-]+)");
	int listSize = get_count();

	for (size_t i = 0; i < listSize; i++)
	{
		if (regex_match(array_stud[i]->get_name(), regular))
			print_one(array_stud[i]);

	}

	cout << endl;
	////////////////
	for (size_t i = 0; i < 50; i++)
		cout << "-";
	cout << endl;
}

bool Arr::sort_forward(const int& a, const int& b)
{
	return a > b;
}

bool Arr::sort_back(const int& a, const int& b)
{
	return a < b;
}

void Arr::sort(comp condition)
{
	Student *temp;
	int size = get_count();
	bool pr;

		do {
			pr = 0;
			for (size_t i = 0; i < size - 1; i++)
			{
				if (condition(array_stud[i]->get_number_stud(), array_stud[i
+ 1]->get_number_stud()))
				{
					temp = array_stud[i];
					array_stud[i] = array_stud[i + 1];
					array_stud[i + 1] = temp;
					pr = 1;
				}
			}
		} while (pr == 1);

	print_array(size);
}
//////////////////////////////////////////
Student* Arr::operator[] (int i)
{
	int c = get_count();

	if (c > i)
		return array_stud[i];

}
```

course.cpp

```
#include "course.h"

int Course::get_course()
{
	return course;
}

void Course::set_course(int c)
{
	course = c;
}

void Course::print() const
{
	cout << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tКурс: " << course << endl;
}

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

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tКурс: " << course << endl;

	return temp;
}

void Course::write_in_file(ofstream& el)
{
	el << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tКурс: " << course << endl;
}

Course::Course() : Student(0, 0, 0, "Name", 0, 0, 0, "Non"), course(0)
{

}

Course::Course(int a, int n, int m, string na, bool d, int pd, sint an, string fc, int c)
: Student(a, n, m, na, d, pd, an, fc), course(c)
{

}

Course::Course(const Course& other) : Student(other), course(other.course)
{

}

Course::~Course()
{
```

```
}
//////////////////////////////////////////////////
Course& Course::operator=(Course& temp)
{
	if (this == &temp) return *this;

	Student::operator=(temp);
	int course = temp.course;

	return *this;
}

bool Course::operator==(const string name) const
{
	return this->name == name;
}

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

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tКурс: " << course << endl;

	return temp;
}

string Course::getInfo() const
{
	stringstream temp;
	temp.setf(ios::left);

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tКурс: " << course << endl;

	return temp.str();
}

void Course::input(istream& input)
{
	input >> age >> number_stud >> middle_mark >> name >> debt >> prog_d
		>> audience >> faculty >> course;
}
```

main.cpp:

```
/**
 * @mainpage
 * <b> Лабораторна робота № 8. <br/> ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ </b>
 * <br/><b><i>Мета роботи:</i></b>:  отримати знання про базові регулярні вирази та
досвід роботи із застосування їх на практиці. <br/>
 * <b><i>Індивідуальне завдання 19:</i></b>
 *     Поширити попередню лабораторну роботу. <br/>
 * <br/>
 * <br/> <b> Arr p; class Student; class Arr; </b> Класи, методи, функції, конструктори,
потоки </b>
 * @author Tatarenko A.
 * @date 29-may-2020
```

```
 * @version 1.0
 */

#include "menu.h"

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

	Arr stud_array;

	auto input = 0;

	cout << "Введите 1 чтобы создать массив, 2 чтобы выйти: ";
	cin >> input;

	if (input == 1)
	{
		stud_array.create_array();
		//cout << "Список созданый с помощью конструктора:" << endl;
		//stud_array.print_array(stud_array.get_count());

		cout << "Рандомное заполнение списка:" << endl;
		stud_array.fill_array();
		stud_array.print_array(stud_array.get_count());

		menu(stud_array);
	}

	////////////////////////////////////////////////////////
	int l = _CrtDumpMemoryLeaks(); // Контроль витоку пам'яті
	if (l == 0)
		cout << "Утечки памяти не обнаруженно!" << endl;
	else
		cout << "Обнаружена утечка памяти!" << endl;

	system("PAUSE");
	return 0;

}
```

faculty.cpp

```
#include "faculty.h"

string Fac::get_fname() const
{
	return fname;
}

void Fac::set_fname(string fn)
{
	fname = fn;
}

Fac::Fac() : fname("Non")
{
}

Fac::Fac(string fn) : fname(fn)
{
}

Fac::Fac(const Fac& other) : fname(other.fname)
{
```

```
}

Fac::~Fac()
{
}
//////////////////////////////////////////////////////
istream& operator>> (istream& input, Fac& date)
{
	input >> date.fname;

	return input;
}
ostream& operator<< (ostream& output, const Fac& date)
{
	output << date.fname;

	return output;
}
```

surnames.cpp

```
#include "surnames.h"

string Surname::get_sur_star() const
{
	return sur_star;
}

string Surname::get_sur_cur() const
{
	return sur_cur;
}

void Surname::set_sur_star(string s_s)
{
	sur_star = s_s;
}

void Surname::set_sur_cur(string s_c)
{
	sur_cur = s_c;
}

void Surname::print() const
{
	cout << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tСтароста: " << sur_star <<
"\tКуратор: " << sur_cur << endl;
}

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

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tСтароста: " << sur_star <<
"\tКуратор: " << sur_cur << endl;
```

```
	return temp;
}

void Surname::write_in_file(ofstream& el)
{
	el << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tСтароста: " << sur_star <<
"\tКуратор: " << sur_cur << endl;
}

Surname::Surname() : Student(0, 0, 0, "Name", 0, 0, 0, "Non"), sur_star("Surname"),
sur_cur("Surname")
{

}

Surname::Surname(int a, int n, int m, string na, bool d, int pd, sint an, string fc,
string s_s, string s_c) : Student(a, n, m, na, d, pd, an, fc), sur_star(s_s),
sur_cur(s_c)
{

}

Surname::Surname(const Surname& other) : Student(other), sur_star(other.sur_star),
sur_cur(other.sur_cur)
{

}

Surname::~Surname()
{

}
//////////////////////////////////////////////////
Surname& Surname::operator=(Surname& temp)
{
	if (this == &temp) return *this;

	Student::operator=(temp);
	string sur_star = temp.sur_star;
	string sur_cur = temp.sur_cur;

	return *this;
}

bool Surname::operator==(const string name) const
{
	return this->name == name;
}

stringstream Surname::getStr() const
{
	stringstream temp;

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tСтароста: " << sur_star <<
"\tКуратор: " << sur_cur << endl;

	return temp;
```

```
}

string Surname::getInfo() const
{
	stringstream temp;
	temp.setf(ios::left);

	temp << "Возраст: " << age << "\tНомер студента: " << number_stud << "\tСредний
балл: " << middle_mark << "\tИмя: "
		<< name << "\tДолг: " << debt << "\tДолг с прог: " << prog_d << "\tНомер
аудитории: " << audience.get_aud_numb()
		<< "\tФакультет: " << faculty.get_fname() << "\tСтароста: " << sur_star <<
"\tКуратор: " << sur_cur << endl;

	return temp.str();
}

void Surname::input(istream& input)
{
	input >> age >> number_stud >> middle_mark >> name >> debt >> prog_d
		>> audience >> faculty >> sur_star >> sur_cur;
}
```

menu.cpp:

```
#include "menu.h"

void menu(Arr stud_array)
{
	int input = 0;

	cout << "1) Добавить объект в конец;" << endl << "2) Удалить объект;" << endl <<
"3) Выбрать объект;" << endl << "4) Вывести все объекты;" << endl << "5) Поиск
должников;" << endl << "6) Запись в файл;" << endl << "7) Читать с файла;" << endl << "8)
REGEX TASK;" << endl << "9) Сортировать;" << endl << "10) Выйти;" << endl << "Ваше
действие: ";
	cin >> input;

	switch (input)
	{
	case 1:
		stud_array.add();
		break;
	case 2:
		stud_array.delete_one();
		break;
	case 3:
		cout << stud_array.select();
		break;
	case 4:
		stud_array.print_array(stud_array.get_count());
		break;
	case 5:
		stud_array.find_debt();
		break;
	case 6:
		stud_array.in_f();
		break;
	case 7:
		stud_array.from_f(stud_array.str_in_file("if.txt"));
		break;
	case 8:
		stud_array.regex_task();
		break;
```

```
	case 9:
		cout << "Выберете тип сортировки:" << endl << "Сортировка по возрастанию;"
<< endl << "Сортировка по убыванию;" << endl << endl;
		int c;
		cin >> c;

		if (c == 1)
			stud_array.sort(stud_array.sort_forward);
		else if (c == 2)
			stud_array.sort(stud_array.sort_back);
		else
		{
			cout << "НЕВЕРНОЕ ДЕЙСТВИЕ!" << endl;
			////////////////
			for (size_t i = 0; i < 50; i++)
				cout << "-";
			cout << endl;
		}
		break;
	case 10:
		return;
	default:
		break;
	}

	menu(stud_array);
}
```

test.cpp:

```
#include "test.h"

int main()
{
	Arr stud_array;

	if (test_count_plus(stud_array))
		cout << "Function count_plus is right!" << endl;
	else
		cout << "Function count_plus is not right!" << endl;

	if (test_count_minus(stud_array))
		cout << "Function count_minus is right!" << endl;
	else
		cout << "Function count_minus is not right!" << endl;

	system("PAUSE");
}

bool test_count_plus(Arr p)
{
	int exp = El + 1;
	int cnt = p.count_plus();

	if (cnt == exp)
		return 1;
	else
		return 0;
}

bool test_count_minus(Arr p)
{
	int exp = El - 1;
	int cnt = p.count_minus();
```

```
        if (cnt == exp)
                return 1;
        else
                return 0;
}
```

4 Результати тестування

5 Опис результатів

Програма виводить список студентів. Виконує видалення, додавання, вибір об’єкта, виводить інформацію про заборгованість. А також виконує запис і читання з файлу, та перевіряє читання через базові регулярні вирази.

Висновок:

Протягом цієї роботи було отримано базові знання про призначення операторів, визначити їх ролі у житті об’єкта та можливість перевизначення.