ЗВІТ

Виконав:

Заночкин Є., КІТ-119а

Дата: 27 квітня 2020 р.

Лабораторна робота №6

СПАДКУВАННЯ

Мета роботи. Отримати знання про парадигму ООП – спадкування. Навчитися застосовувати отримані знання на практиці.

1.Завдання до роботи

Варіант 6.

Загальне завдання: Модернізувати попередню лабораторну роботу шляхом:

- додавання класу-спадкоємця, котрий буде поширювати функціонал «базового класу» відповідно до індивідуального завдання;

- додавання ще одного класу-списку, що буде керувати лише елементами класу-спадкоємця;

Індивідуальне завдання:

Прикладна галузь – Самостійні роботи студентів;

Додаткові поля у класі-спадкоємці – розрахунково-графічне завдання за науковою тематикою керівника;

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: Task.

Клас-спадкоємець: Inheritor.

Клас, що має в собі масив базового класу та методи для роботи з ним: List.

Клас, що має в собі масив класу-спадкоємця та методи для роботи з ним: InheritorList.

Клас, що відображає агрегативні відносини з базовим класом: Student.

Клас, що відображає композитивні відносини з базовим класом: Date.

2.2 Опис змінних

int mark – оцінка за роботу (Task).

int countOfDoneExercises – кількість виконаних завдань (Task).

int studentIndex – індекс студента (Task).

string name – ім’я студента (Task).

int rgz – кількість ргз (Task).

Student age – вік студента (Student).

Date date – дата написання (Date).

int rgzForTeacher – кількість ргз для керівника (Inheritor).

int listSize – розмір масиву елементів класу List.

Task* stud – масив елементів класу Task.

Inheritor* stud – масив елементів класу Inheritor.

List list – об’єкт класу List.

InheritorList list1 - об’єкт класу InheritorList.

Student age – об’єкт класу Student.

List test – об’єкт класу List.

Task var, add - об’єкти класу Task.

Inheritor var1, add1 - об’єкти класу Inheritor.

2.3 Опис методів

void createList() – створення масиву елементів і заповнення даними (List).

void printAll() const – виведення даних елементів у консоль (List).

void printOneStudent(stringstream&) const – виведення даних одного студента у консоль (List).

void addStudent(const Task) – додавання нового елементу в масив (List).

void addStudent(const Inheritor) – додавання нового елементу в масив (InheritorList).

void deleteStudent(int) – видалення елемента з масиву (List).

int getStudentID(int)const – отримання даних елемента по індексу (List).

int getStudentRGZ(int)const - отримання даних елемента по кількості ргз (List).

void ReadFile(string, int) – читання даних з файлу (List).

int FileString(string) – кількість рядків у файлі (List).

void WriteFile(string) const – запис даних у файл (List).

stringstream getObj(int) const – повернення студента у вигляді рядка (List).

Task enterNewStudent() – введення нового студента з клавіатури (List).

Inheritor enterNewStudent() – введення нового студента з клавіатури (InheritorList).

void regexTask() – виведення на екран об’єктів, які в полі string мають 2 слова (List).

static bool sortAsc(const int&, const int&) – функція, що визначає напрям сортування (List).

static bool sortDesc(const int&, const int&) – функція, що визначає напрям сортування (List).

void sort(comp) – сортування масиву (List).

Task() – конструктор за змовчуванням (Task).

Task(int, string, int, int, int, int, sint, sint, sint) – конструктор з аргументами (Task).

Task(const Task&) – конструктор копіювання (Task).

Inheritor() – конструктор за змовчуванням (Inheritor).

Inheritor (int, string, int, int, int, int, sint, sint, sint, int) – конструктор з аргументами (Inheritor).

Inheritor (const Inheritor&) – конструктор копіювання (Inheritor).

~Task() – деструктор (Task).

~List() – деструктор (List).

~InheritorList() - деструктор (Inheritor).

2.4 Опис функцій

void Menu() – функція меню.

int generateID() – функція, що надає кожному об’єкту класу Task унікальний індекс.

int generateRGZ() – функція, що надає кожному об’єкту класу Task унікальну кількість ргз.

int InherGenerateID() – функція, що надає кожному об’єкту класу Inheritor унікальний індекс.

int InherGenerateRGZ() – функція, що надає кожному об’єкту класу Inheritor унікальну кількість ргз.

int InherGenerateRGZForTeacher() - функція, що надає кожному об’єкту класу Inheritor унікальну кількість ргз для керівника.

Task CreateStudent() – стандартний об’єкт класу Task.

Task CreateStudent2() – стандартний об’єкт класу Task.

Inheritor CreateStudent() – стандартний об’єкт класу Inheritor.

Inheritor CreateStudent2() – стандартний об’єкт класу Inheritor.

void TestAddStudent(List&, int) – тест функції додавання об’єкта до масиву об’єктів.

void TestDeleteStudent(List&, int) – тест функції видалення об’єкта з масиву об’єктів.

void TestGetStudenttID(List&, int) – тест функції повернення індексу студента.

int TestReadFile(List&, int) – тест функції читання даних з файлу.

3. Текст програми

Task.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#define __CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, __FILE__, __LINE__)
#define new DEBUG_NEW
#include <string>
#include <iostream>
#include <iomanip>
#include <sstream>
```

```
#include <fstream>
#include <regex>
#include <cstddef>
#include "Date.h"
#include "Student.h"
using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;
using std::regex;
typedef bool (comp)(const int&, const int&);
class Task
{
private:
int studentIndex;
string name;
int mark;
int countOfDoneExercises;
int rgz;
Date date;
Student age;
public:
int getMark() const;
void setMark(int);
int getCountOfDoneExercises() const;
void setCountOfDoneExercises(int);
int getStudentIndex() const;
void setStudentIndex(int);
int getRgz() const;
void setRgz(int);
string getName() const;
void setName(const string);
sint getDay() const;
void setDay(const sint);
sint getMonth() const;
void setMonth(const sint);
sint getYear() const;
void setYear(const sint);
int getAge() const;
void setAge(const int);
```

```
int generateID();
int generateRGZ();
Task();
Task(int, string, int, int, int, int, sint, sint, sint);
Task(const Task&);
~Task();
};
void Menu();
Task CreateStudent();
Task CreateStudent2();
```

List.h

```
#pragma once
#include "Task.h"
class List
{
private:
int listSize;
public:
Task* stud;
int getListSize() const;
void setListSize(int);
void addStudent(const Task);
void deleteStudent(int);
void printAll() const;
void printOneStudent(stringstream&) const;
void createList();
int getStudentID(int)const;
int getStudentRGZ(int)const;
void ReadFile(string, int);
int FileString(string);
void WriteFile(string) const;
stringstream getObj(int i) const;
Task enterNewStudent();
void regexTask();
static bool sortAsc(const int&, const int&);
static bool sortDesc(const int&, const int&);
void sort(comp);
~List();
};
```

Task.cpp

```
#include "Task.h"
#include "List.h"
int Task::getMark() const { return mark; }
void Task::setMark(int mark1) { mark = mark1; }
int Task::getCountOfDoneExercises() const { return countOfDoneExercises; }
void Task::setCountOfDoneExercises(int count_of_done_exercises1)
{ countOfDoneExercises = count_of_done_exercises1; }
int Task::getStudentIndex() const { return studentIndex; }
void Task::setStudentIndex(int exercises1) { studentIndex = exercises1; }
int Task::getRgz() const { return rgz; }
void Task::setRgz(int rgz1) { rgz = rgz1; }
string Task::getName() const { return name; }
void Task::setName(string name1) { name = name1; }
sint Task::getDay() const { return date.getDay(); }
void Task::setDay(sint day1) { date.setDay(day1); }
sint Task::getMonth() const { return date.getMonth(); }
void Task::setMonth(sint month1) { date.setMonth(month1); }
sint Task::getYear() const { return date.getYear();; }
void Task::setYear(sint year1) { date.setYear(year1); }
int Task::getAge() const { return age.getAge(); }
void Task::setAge(const int age1) { age.setAge(age1); }
Task CreateStudent()
{
Task stud;
stud.setMark(5);
stud.setCountOfDoneExercises(5);
stud.setName("Petrova Katya");
stud.setDay(5);
stud.setMonth(5);
stud.setYear(5555);
stud.setAge(5);
return stud;
}
Task CreateStudent2()
{
Task stud;
stud.setMark(2);
stud.setCountOfDoneExercises(2);
stud.setName("Ivanov Petya");
stud.setDay(2);
stud.setMonth(2);
stud.setYear(2222);
```

```
stud.setAge(2);
return stud;
}
int Task::generateID()
{
	static int id = 1;
	return id++;
}

int Task::generateRGZ()
{
	static int RGZ = 5;
	return RGZ++;
}
```

Task::Task(int student_index, string name, int age, int mark, int countOfDoneExercises, int rgz, sint day, sint month, sint year) : studentIndex(student_index), name(name), age(age), mark(mark), countOfDoneExercises(countOfDoneExercises), rgz(rgz), date(day, month, year){}

Task::Task() : studentIndex(0), name(" "), age(0), mark(0), countOfDoneExercises(0), rgz(0), date(0, 0, 0000) {}

Task::Task(const Task& stud) : studentIndex(stud.studentIndex), name(stud.name), age(stud.age), mark(stud.mark), countOfDoneExercises(stud.countOfDoneExercises), rgz(stud.rgz),date(stud.date) {}

Task::~Task() {}

List.cpp

```
#include "List.h"
#include "Task.h"
#include "Date.h"

int List::getListSize() const { return listSize; }
void List::setListSize(int size) { listSize = size; }

void List::addStudent(const Task task)
{
	if (task.getMark() == 0)
	{
		cout << "Empty object. Error";
		return;
	}
	setListSize(getListSize() + 1);
	Task* newstud = new Task[listSize];
	for (size_t i = 0; i < listSize - 1; i++)
		newstud[i] = stud[i];
	newstud[listSize - 1] = task;
	if (newstud[listSize - 1].getStudentIndex() == 0)
	{
		newstud[listSize - 1].setStudentIndex(stud->generateID());
		newstud[listSize - 1].setRgz(stud->generateRGZ());
	}
	delete[] stud;
	stud = new Task[listSize];
	for (size_t i = 0; listSize > i; i++)
		stud[i] = newstud[i];
	delete[] newstud;
}
```

```
void List::createList()
{
	stud = new Task[listSize];
	for (size_t i = 0; listSize > i; i++)
	{
		stud[i] = CreateStudent();
		stud[i].setStudentIndex(stud->generateID());
		stud[i].setRgz(stud->generateRGZ());
	}
}

void List::deleteStudent(int c)
{
	setListSize(getListSize() - 1);
	Task* newstud = new Task[listSize];
	size_t i = 0;
	for (; i < getListSize(); i++)
	{
		if (stud[i].getStudentIndex() == c)
			break;
		newstud[i] = stud[i];
	}
	for (; i < getListSize(); i++)
		newstud[i] = stud[i + 1];
	delete[] stud;
	stud = new Task[listSize];
	for (size_t i = 0; i < getListSize(); i++)
		stud[i] = newstud[i];
	delete[] newstud;
}

void List::printOneStudent(stringstream& ss) const
{
	int index;
	string name, name2;
	int mark, age;
	int exercises;
	int rgz;
	sint day, month, year;
	cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" <<
setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << "Date" << endl;
	ss >> index;
	ss >> name;
	ss >> name2;
	ss >> age;
	ss >> mark;
	ss >> exercises;
	ss >> rgz;
	ss >> day;
	ss >> month;
	ss >> year;

	if (name2 == "")
		name = name + " ";
	else (name = name + " " + name2);

	cout << std::left;
	cout << setw(6) << index;
	cout << setw(18) << name;
	cout << setw(8) << age;
	cout << setw(13) << mark;
	cout << setw(13) << exercises;
	cout << setw(10) << rgz;
```

```
		cout << setw(3) << day << setw(3) << month << year << endl;
	}

	void List::printAll() const
	{
		cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" << setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << "Date" << endl;
		for (size_t i = 0; List::getListSize() > i; i++)
			cout << std::left << setw(6) << stud[i].getStudentIndex() << setw(18) << stud[i].getName() << setw(8) << stud[i].getAge() << setw(13) << stud[i].getMark() << setw(13) << stud[i].getCountOfDoneExercises() << setw(10) << stud[i].getRgz() << setw(3) << stud[i].getDay() << setw(3) << stud[i].getMonth() << stud[i].getYear() << endl;
	}

	List::~List()
	{
		delete[] stud;
	}

	int List::getStudentID(int id) const
	{
		for (size_t i = 0; i < listSize; i++)
			if (stud[i].getStudentIndex() == id)
				return i;
		cout << "Wrong ID" << endl;
		return -1;
	}

	int List::getStudentRGZ(int a) const
	{
		for (size_t i = 0; i < listSize; i++)
			if (stud[i].getRgz() == a)
				return i;
		cout << "Wrong count of RGZ" << endl;
		return -1;
	}

	void List::ReadFile(string filename, int c)
	{
		std::ifstream fin(filename);
		if (!fin.is_open())
			return;
		string line;
		regex regular("([\\d]* [A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]*)");
		int i = 0, a = 0, b = 0;
		delete[] stud;
		stud = new Task[c];
		while (getline(fin, line) && i < c)
		{
			if (regex_match(line.c_str(), regular))
			{
				int studentIndex;
				string name, name2;
				int mark, age;
				int countOfDoneExercises;
				int rgz;
				sint day, month, year;
				std::istringstream fin(line);
				fin >> studentIndex;
				fin >> name;
				fin >> name2;
				fin >> age;
```

```
                fin >> mark;
                fin >> countOfDoneExercises;
                fin >> rgz;
                fin >> day;
                fin >> month;
                fin >> year;

                if (name2 == "")
                        name = name + " ";
                else (name = name + " " + name2);

                do
                {
                        b = 0;

                        a = name.find("--");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                        a = name.find("  ");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                        a = name.find(",,");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                        a = name.find("::");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                        a = name.find(";;");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                        a = name.find("_");
                        if (a != -1)
                        {
                                name.erase(a, 1);
                                b = 1;
                        }

                } while (b == 1);

                Task ex(studentIndex, name, age, mark, countOfDoneExercises, rgz, day, month,
year);
                stud[i++] = ex;
        }
```

```
		}
		setListSize(c);
		fin.close();
		cout << "Data from file have written" << endl;
		return;
	}

	int List::FileString(string filename)
	{
		int c = 0;
		string line;
		regex regular("([\\d]* [A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]*)");
		std::ifstream fin(filename);
		if (!fin.is_open())
		{
			cout << "Error open file";
			return 0;
		}
		while (getline(fin, line))
		{
			if (regex_match(line, regular))
				c++;
			else cout << "String is not correct" << endl;
		}

		fin.close();
		return c;
	}

	void List::WriteFile(string filename) const
	{
		std::ofstream fout(filename);
		if (!fout.is_open())
		{
			cout << "Error open file";
			return;
		}
		fout << "Base class" << endl << endl;
		fout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" << setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << "Date" << endl;
		for (size_t i = 0; i < listSize; i++)
		{
			fout << std::left;
			fout << setw(6) << stud[i].getStudentIndex();
			fout << setw(18) << stud[i].getName();
			fout << setw(8) << stud[i].getAge();
			fout << setw(13) << stud[i].getMark();
			fout << setw(13) << stud[i].getCountOfDoneExercises();
			fout << setw(10) << stud[i].getRgz();
			fout << setw(3) << stud[i].getDay() << setw(3) << stud[i].getMonth() << stud[i].getYear() << endl;
		}
		cout << "Write to file - correct" << endl;
		return;
	}

	stringstream List::getObj(int i) const
	{
		stringstream ss;
		ss << stud[i].getStudentIndex();
		ss << " " << stud[i].getName();
		ss << " " << stud[i].getAge();
```

```
		ss << " " << stud[i].getMark();
		ss << " " << stud[i].getCountOfDoneExercises();
		ss << " " << stud[i].getRgz();
		ss << " " << stud[i].getDay();
		ss << " " << stud[i].getMonth();
		ss << " " << stud[i].getYear();
		return ss;
	}

	Task List::enterNewStudent()
	{
		Task add, error;
		int index, mark, rgz, exercises, age;
		string name, surname, data;
		sint day, month, year;
		regex regular("([\\d]* [A-Z]+[\\wA-Za-z,.;:-]* [A-Z]+[\\wA-Za-z,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]*
[\\d]* [\\d]* [\\d]*)");
		cout << "Enter student data (ID, Surname, Name, Age, Mark, Exercises, RGZ,
Date(day,month,year)):" << endl;
		cin.ignore();
		getline(cin, data);
		std::istringstream temp(data);
		temp >> index;
		temp >> surname;
		temp >> name;
		temp >> age;
		temp >> mark;
		temp >> exercises;
		temp >> rgz;
		temp >> day;
		temp >> month;
		temp >> year;

		if (name == "")
			surname = surname + " ";
		else (surname = surname + " " + name);

		if (!regex_match(data, regular))
		{
			cout << "You enter wrong data";
			return error;
		}

		add.setStudentIndex(index);
		add.setName(surname);
		add.setAge(age);
		add.setMark(mark);
		add.setCountOfDoneExercises(exercises);
		add.setRgz(rgz);
		add.setDay(day);
		add.setMonth(month);
		add.setYear(year);
		return add;
	}

	void List::regexTask()
	{
		stringstream ss;
		int index, mark, rgz, exercises, age;
		string name, name2;
		sint day, month, year;
		regex regular("(^[A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]+)");
		int listSize = getListSize();
```

```
		cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" << setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << "Date" << endl;
		for (size_t i = 0; i < listSize; i++)
			if (regex_match(stud[i].getName(), regular))
			{
				ss = getObj(i);
				ss >> index >> name >> name2 >> age >> mark >> exercises >> rgz >> day >> month >> year;

				if (name2 == "")
					name = name + " ";
				else (name = name + " " + name2);

				cout << std::left;
				cout << setw(6) << index;
				cout << setw(18) << name;
				cout << setw(8) << age;
				cout << setw(13) << mark;
				cout << setw(13) << exercises;
				cout << setw(10) << rgz;
				cout << setw(3) << day << setw(3) << month << year << endl;
			}
	}

	bool List::sortAsc(const int& a, const int& b) { return a > b; }
	bool List::sortDesc(const int& a, const int& b) { return a < b; }

	void List::sort(comp condition)
	{
		Task temp;
		int pr;
		do
		{
			pr = 0;
			for (size_t i = 0; i < getListSize() - 1; i++)
			{
				if (condition(stud[i].getMark(), stud[i + 1].getMark()))
				{
					temp = stud[i];
					stud[i] = stud[i + 1];
					stud[i + 1] = temp;
					pr = 1;
				}
			}
		} while (pr == 1);
	}
```

InherTask.h

```
	#include "Task.h"
	#pragma once
	class Inheritor : public Task
	{
	private:
		int rgzForTeacher;
	public:
		int getRgzForTeacher() const;
		void setRgzForTeacher(int);
		int InherGenerateID();
		int InherGenerateRGZ();
		int InherGenerateRGZForTeacher();
		Inheritor();
```

```
        Inheritor(int, string, int, int, int, int, sint, sint, sint, int);
        Inheritor(const Inheritor&);
        ~Inheritor();
};
Inheritor InherCreateStudent();
Inheritor InherCreateStudent2();
```

InherList.h

```
#pragma once
#include "InherTask.h"
class InheritorList
{
private:
        int listSize;
public:
        Inheritor* stud;
        int getListSize() const;
        void setListSize(int);
        void addStudent(const Inheritor);
        void deleteStudent(int);
        void printAll() const;
        void printOneStudent(stringstream&) const;
        void createList();
        int getStudentID(int)const;
        int getStudentRGZ(int)const;
        void ReadFile(string, int);
        int FileString(string);
        void WriteFile(string) const;
        stringstream getObj(int i) const;
        Inheritor enterNewStudent();
        void regexTask();
        static bool sortAsc(const int&, const int&);
        static bool sortDesc(const int&, const int&);
        void sort(comp);
        ~InheritorList();
};
```

InherTask.cpp

```
#include "InherTask.h"
int Inheritor::getRgzForTeacher() const { return rgzForTeacher; }
void Inheritor::setRgzForTeacher(int rgz1) { rgzForTeacher = rgz1; }
Inheritor InherCreateStudent()
{
        Inheritor stud;
        stud.setMark(5);
        stud.setCountOfDoneExercises(5);
        stud.setName("Petrova Katya");
        stud.setDay(5);
        stud.setMonth(5);
        stud.setYear(5555);
        stud.setAge(5);
        return stud;
}
Inheritor InherCreateStudent2()
{
        Inheritor stud;
        stud.setMark(2);
        stud.setCountOfDoneExercises(2);
```

```
		stud.setName("Ivanov Petya");
		stud.setDay(2);
		stud.setMonth(2);
		stud.setYear(2222);
		stud.setAge(2);
		return stud;
	}
	int Inheritor::InherGenerateID()
	{
		static int id = 1;
		return id++;
	}
	int Inheritor::InherGenerateRGZ()
	{
		static int RGZ = 5;
		return RGZ++;
	}
	int Inheritor::InherGenerateRGZForTeacher()
	{
		static int rgzForTeacher = 5;
		return rgzForTeacher++;
	}
	Inheritor::Inheritor() : Task(), rgzForTeacher(0) {}
	Inheritor::Inheritor(int student_index, string name, int age, int mark, int countOfDoneExercises, int rgz, sint day, sint month, sint year,int rgzForTeacher) : Task(student_index, name, age, mark, countOfDoneExercises, rgz, day, month, year), rgzForTeacher(rgzForTeacher) {}
	Inheritor::Inheritor(const Inheritor& object) : Task(object), rgzForTeacher(object.rgzForTeacher) {}
	Inheritor::~Inheritor() {}
```

InherList.cpp

```
	#include "InherList.h"
	#include "InherTask.h"
	int InheritorList::getListSize() const { return listSize; }
	void InheritorList::setListSize(int size) { listSize = size; }
	void InheritorList::addStudent(const Inheritor task)
	{
		if (task.getMark() == 0)
		{
			cout << "Empty object. Error";
			return;
		}
		setListSize(getListSize() + 1);
		Inheritor* newstud = new Inheritor[listSize];
		for (size_t i = 0; i < listSize - 1; i++)
			newstud[i] = stud[i];
		newstud[listSize - 1] = task;
		if (newstud[listSize - 1].getStudentIndex() == 0)
		{
			newstud[listSize - 1].setStudentIndex(stud->InherGenerateID());
			newstud[listSize - 1].setRgz(stud->InherGenerateRGZ());
			newstud[listSize - 1].setRgzForTeacher(stud->InherGenerateRGZForTeacher());
		}
		delete[] stud;
		stud = new Inheritor[listSize];
		for (size_t i = 0; listSize > i; i++)
			stud[i] = newstud[i];
		delete[] newstud;
	}
	void InheritorList::createList()
	{
		stud = new Inheritor[listSize];
		for (size_t i = 0; listSize > i; i++)
		{
```

```
                        stud[i] = InherCreateStudent();
                        stud[i].setStudentIndex(stud->InherGenerateID());
                        stud[i].setRgz(stud->InherGenerateRGZ());
                        stud[i].setRgzForTeacher(stud->InherGenerateRGZForTeacher());
                }
        }
        void InheritorList::deleteStudent(int c)
        {
                setListSize(getListSize() - 1);
                Inheritor* newstud = new Inheritor[listSize];
                size_t i = 0;
                for (; i < getListSize(); i++)
                {
                        if (stud[i].getStudentIndex() == c)
                                break;
                        newstud[i] = stud[i];
                }
                for (; i < getListSize(); i++)
                        newstud[i] = stud[i + 1];
                delete[] stud;
                stud = new Inheritor[listSize];
                for (size_t i = 0; i < getListSize(); i++)
                        stud[i] = newstud[i];
                delete[] newstud;
        }
        void InheritorList::printOneStudent(stringstream& ss) const
        {
                int index;
                string name, name2;
                int mark, age;
                int exercises;
                int rgz, rgzForTeacher;
                sint day, month, year;
                cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" <<
setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << setw(20) << "Date" << "RGZ for
Teacher" << endl;
                ss >> index;
                ss >> name;
                ss >> name2;
                ss >> age;
                ss >> mark;
                ss >> exercises;
                ss >> rgz;
                ss >> day;
                ss >> month;
                ss >> year;
                ss >> rgzForTeacher;
                if (name2 == "")
                        name = name + " ";
                else (name = name + " " + name2);
                cout << std::left;
                cout << setw(6) << index;
                cout << setw(18) << name;
                cout << setw(8) << age;
                cout << setw(13) << mark;
                cout << setw(13) << exercises;
                cout << setw(10) << rgz;
                cout << setw(3) << day << setw(3) << month << setw(20) << year;
                cout << rgzForTeacher << endl;
        }
        void InheritorList::printAll() const
        {
```

```
	cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" << setw(10) <<
"Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" <<setw(20)<< "Date" << "RGZ for Teacher"<< endl;
		for (size_t i = 0; InheritorList::getListSize() > i; i++)
			cout << std::left << setw(6) << stud[i].getStudentIndex() << setw(18) <<
stud[i].getName() << setw(8) << stud[i].getAge() << setw(13) << stud[i].getMark() << setw(13) <<
stud[i].getCountOfDoneExercises() << setw(10) << stud[i].getRgz() << setw(3) << stud[i].getDay() << setw(3) <<
stud[i].getMonth() <<setw(20)<< stud[i].getYear()<<stud[i].getRgzForTeacher() << endl;
	}
	InheritorList::~InheritorList()
	{
		delete[] stud;
	}
	int InheritorList::getStudentID(int id) const
	{
		for (size_t i = 0; i < listSize; i++)
			if (stud[i].getStudentIndex() == id)
				return i;
		cout << "Wrong ID" << endl;
		return -1;
	}
	int InheritorList::getStudentRGZ(int a) const
	{
		for (size_t i = 0; i < listSize; i++)
			if (stud[i].getRgz() == a)
				return i;
		cout << "Wrong count of RGZ" << endl;
		return -1;
	}
	void InheritorList::ReadFile(string filename, int c)
	{
		std::ifstream fin(filename);
		if (!fin.is_open())
			return;
		string line;
		regex regular("([\\d]* [A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]*
[\\d]* [\\d]* [\\d]*)");
		int i = 0, a = 0, b = 0;
		delete[] stud;
		stud = new Inheritor[c];
		while (getline(fin, line) && i < c)
		{
			if (regex_match(line.c_str(), regular))
			{
				int studentIndex;
				string name, name2;
				int mark, age;
				int countOfDoneExercises;
				int rgz, rgzForTeacher;
				sint day, month, year;
				std::istringstream fin(line);
				fin >> studentIndex;
				fin >> name;
				fin >> name2;
				fin >> age;
				fin >> mark;
				fin >> countOfDoneExercises;
				fin >> rgz;
				fin >> day;
				fin >> month;
				fin >> year;
				fin >> rgzForTeacher;
				if (name2 == "")
					name = name + " ";
```

```
                                        else (name = name + " " + name2);
                                        do
                                        {
                                                b = 0;
                                                a = name.find("--");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }
                                                a = name.find("  ");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }
                                                a = name.find(",,");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }
                                                a = name.find("::");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }
                                                a = name.find(";;");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }
                                                a = name.find("_");
                                                if (a != -1)
                                                {
                                                        name.erase(a, 1);
                                                        b = 1;
                                                }

                                        } while (b == 1);
                                        Inheritor ex(studentIndex, name, age, mark, countOfDoneExercises, rgz, day,
month, year, rgzForTeacher);
                                        stud[i++] = ex;
                                }
                        }
                        setListSize(c);
                        fin.close();
                        cout << "Data from file have written" << endl;
                        return;
                }
                int InheritorList::FileString(string filename)
                {
                        int c = 0;
                        string line;
                        regex regular("([\\d]* [A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]* [\\d]*
[\\d]* [\\d]* [\\d]*)");
                        std::ifstream fin(filename);
                        if (!fin.is_open())
                        {
                                cout << "Error open file";
                                return 0;
```

```
		}
		while (getline(fin, line))
		{
			if (regex_match(line, regular))
				c++;
			else cout << "String is not correct" << endl;
		}

		fin.close();
		return c;
	}
	void InheritorList::WriteFile(string filename) const
	{
		std::ofstream fout(filename, std::ios::app);
		if (!fout.is_open())
		{
			cout << "Error open file";
			return;
		}
		fout << endl << "Inheritor class" << endl << endl;
		fout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" <<
setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << setw(20) << "Date" << "RGZ for
Teacher" << endl;
		for (size_t i = 0; i < listSize; i++)
		{
			fout << std::left;
			fout << setw(6) << stud[i].getStudentIndex();
			fout << setw(18) << stud[i].getName();
			fout << setw(8) << stud[i].getAge();
			fout << setw(13) << stud[i].getMark();
			fout << setw(13) << stud[i].getCountOfDoneExercises();
			fout << setw(10) << stud[i].getRgz();
			fout << setw(3) << stud[i].getDay() << setw(3) << stud[i].getMonth() << setw(20) <<
stud[i].getYear();
			fout << stud[i].getRgzForTeacher() << endl;
		}
		cout << "Write to file - correct" << endl;
		fout.close();
		return;
	}
	stringstream InheritorList::getObj(int i) const
	{
		stringstream ss;
		ss << stud[i].getStudentIndex();
		ss << " " << stud[i].getName();
		ss << " " << stud[i].getAge();
		ss << " " << stud[i].getMark();
		ss << " " << stud[i].getCountOfDoneExercises();
		ss << " " << stud[i].getRgz();
		ss << " " << stud[i].getDay();
		ss << " " << stud[i].getMonth();
		ss << " " << stud[i].getYear();
		ss << " " << stud[i].getRgzForTeacher();
		return ss;
	}
	Inheritor InheritorList::enterNewStudent()
	{
		Inheritor add, error;
		int index, mark, rgz, exercises, age, rgzForTeacher;
		string name, surname, data;
		sint day, month, year;
		regex regular("([\\d]* [A-Z]+[\\wA-Za-z,.;:-]* [A-Z]+[\\wA-Za-z,.;:-]* [\\d]* [\\d]* [\\d]* [\\d]*
[\\d]* [\\d]* [\\d]* [\\d]*)");
```

```
		cout << "Enter student data (ID, Surname, Name, Age, Mark, Exercises, RGZ, Date(day,month,year), RGZ for Teacher):" << endl;
		getline(cin, data);
		std::istringstream temp(data);
		temp >> index;
		temp >> surname;
		temp >> name;
		temp >> age;
		temp >> mark;
		temp >> exercises;
		temp >> rgz;
		temp >> day;
		temp >> month;
		temp >> year;
		temp >> rgzForTeacher;
		if (name == "")
			surname = surname + " ";
		else (surname = surname + " " + name);
		if (!regex_match(data, regular))
		{
			cout << "You enter wrong data";
			return error;
		}
		add.setStudentIndex(index);
		add.setName(surname);
		add.setAge(age);
		add.setMark(mark);
		add.setCountOfDoneExercises(exercises);
		add.setRgz(rgz);
		add.setDay(day);
		add.setMonth(month);
		add.setYear(year);
		add.setRgzForTeacher(rgzForTeacher);
		return add;
	}
	void InheritorList::regexTask()
	{
		stringstream ss;
		int index, mark, rgz, exercises, age, rgzForTeacher;
		string name, name2;
		sint day, month, year;
		regex regular("(^[A-ZA-Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]+)");
		int listSize = getListSize();
		cout << std::left << setw(10) << "Index" << setw(13) << "Name" << setw(8) << "Age" << setw(10) << "Mark" << setw(16) << "Exercises" << setw(13) << "RGZ" << setw(20) << "Date" << "RGZ for Teacher" << endl;
		for (size_t i = 0; i < listSize; i++)
			if (regex_match(stud[i].getName(), regular))
			{
				ss = getObj(i);
				ss >> index >> name >> name2 >> age >> mark >> exercises >> rgz >> day >> month >> year >> rgzForTeacher;
				if (name2 == "")
					name = name + " ";
				else (name = name + " " + name2);
				cout << std::left;
				cout << setw(6) << index;
				cout << setw(18) << name;
				cout << setw(8) << age;
				cout << setw(13) << mark;
				cout << setw(13) << exercises;
				cout << setw(10) << rgz;
				cout << setw(3) << day << setw(3) << month << setw(20) << year;
```

```
                        cout << rgzForTeacher << endl;
                }
}
bool InheritorList::sortAsc(const int& a, const int& b) { return a > b; }
bool InheritorList::sortDesc(const int& a, const int& b) { return a < b; }
void InheritorList::sort(comp condition)
{
        Inheritor temp;
        int pr;
        do
        {
                pr = 0;
                for (size_t i = 0; i < listSize - 1; i++)
                {
                        if (condition(stud[i].getMark(), stud[i + 1].getMark()))
                        {
                                temp = stud[i];
                                stud[i] = stud[i + 1];
                                stud[i + 1] = temp;
                                pr = 1;
                        }
                }
        } while (pr == 1);
}
```

Menu.cpp

```
#include "Task.h"
#include "List.h"
#include "InherList.h"
#include "InherTask.h"
void Menu()
{
        List list;
        InheritorList list1;
        Student age;
        int c = 0, a = 0, b = 0, value = 0;
        auto count_of_students = 1;
        int menu_number = 1;
        int delete_number;
        string fileName;
        stringstream ss,ss1;
        Task var, add;
        Inheritor var1, add1;
        int file;
        int* studentAge;
        list.setListSize(count_of_students);
        list1.setListSize(count_of_students);
        studentAge = age.createList(count_of_students);
        list.createList();
        list1.createList();
        while (menu_number)
        {
                menu_number = 0;
                cout << endl << "Menu:" << endl;
                cout << "1.Add a new student" << endl;
                cout << "2.Add a new student (enter from keyboard)" << endl;
                cout << "3.Delete one student" << endl;
                cout << "4.Show all student" << endl;
                cout << "5.Show student via his index" << endl;
                cout << "6.Show student via his count of RGZ" << endl;
                cout << "7.Read array from file" << endl;
```

```
cout << "8.Write to file" << endl;
cout << "9.Surname+Name in object" << endl;
cout << "10.Sort (mark)" << endl;
cout << "11.End program" << endl;
cin >> menu_number;
switch (menu_number)
{
case 1:
	var = CreateStudent2();
	list.addStudent(var);
	var1 = InherCreateStudent2();
	list1.addStudent(var1);
	break;
case 2:
	list.addStudent(list.enterNewStudent());
	list1.addStudent(list1.enterNewStudent());
	break;
case 3:
	cout << "Enter a index of student who you want to delete:" << endl;
	cin >> delete_number;
	if (delete_number < 1)
	{
		cout << "Wrong student index" << endl;
		break;
	}
	for (size_t i = 0; list.getListSize() > i; i++)
		if (delete_number == list.stud[i].getStudentIndex())
		{
			list.deleteStudent(delete_number);
			list1.deleteStudent(delete_number);
			break;
		}
	break;
case 4:
	cout << "Base class" << endl << endl;
	list.printAll();
	cout << endl << "Inheritor class" << endl << endl;
	list1.printAll();
	break;
case 5:
	cout << "Enter a index of student:";
	cin >> c;
	b = list.getStudentID(c);
	if (b == -1)
		break;
	ss = list.getObj(b);
	ss1 = list1.getObj(b);
	cout << "Base class" << endl << endl;
	list.printOneStudent(ss);
	cout << endl << "Inheritor class" << endl << endl;
	list1.printOneStudent(ss1);
	break;
case 6:
	cout << "Enter a count of RGZ:";
	cin >> a;
	b = list.getStudentRGZ(a);
	if (b == -1)
		break;
	ss = list.getObj(b);
	ss1 = list1.getObj(b);
	cout << "Base class" << endl << endl;
	list.printOneStudent(ss);
	cout << endl << "Inheritor class" << endl << endl;
```

```
                        list1.printOneStudent(ss1);
                        break;
                case 7:
                        cout << "Enter file name for base class:";
                        cin >> fileName;
                        list.ReadFile(fileName, list.FileString(fileName));
                        cout << "Enter file name for inheritor class:";
                        cin >> fileName;
                        list1.ReadFile(fileName, list1.FileString(fileName));
                        break;
                case 8:
                        cout << "Enter file name:";
                        cin >> fileName;
                        list.WriteFile(fileName);
                        list1.WriteFile(fileName);
                        break;
                case 9:
                        cout << "Base class" << endl << endl;
                        list.regexTask();
                        cout << endl << "Inheritor class" << endl << endl;
                        list1.regexTask();
                        break;
                case 10:
                        cout << "1) Increasing" << endl;
                        cout << "2) Decreasing" << endl;
                        cin >> value;
                        cout << endl;
                        if (value == 1)
                        {
                                list.sort(list.sortAsc);
                                list1.sort(list1.sortAsc);
                        }
                        else if (value == 2)
                        {
                                list.sort(list.sortDesc);
                                list1.sort(list1.sortDesc);
                        }
                        else cout << "Wrong number." << endl;
                        break;
                case 11:
                        menu_number = 0;
                        break;
                default:
                        cout << "You have chosen the wrong number of the menu";
                        break;
                }
        }
        age.deleteList();
        return;
}
```

main.cpp

```
#include "Task.h"

int main()

{

Menu();

if (_CrtDumpMemoryLeaks())

cout << endl << "WARNING! Memory leak" << endl;

else
```

```
cout << endl << "There is no memory leak" << endl;
return 0;
}
```

Student.h

```
#pragma once
#include <iostream>
#include <string>
using std::string;
class Student {
private:
int age;
int listSize;
int* list;
public:
int getAge()const;
void setAge(const int);
void deleteList();
int* createList(int size);
int students(int value);
Student();
Student(int age);
Student(const Student& other);
~Student();
};
```

Date.h

```
#pragma once
typedef short sint;
class Date
{
private:
sint day;
sint month;
sint year;
public:
sint getDay() const;
void setDay(sint day);
sint getMonth() const;
void setMonth(sint month);
sint getYear() const;
```

```
void setYear(sint year);
Date();
Date(sint, sint, sint);
Date(const Date& date);
~Date();
};
```

Student.cpp

```
#include "Student.h"
int Student::getAge() const { return age; }
void Student::setAge(const int age1) { age = age1; }
int* Student::createList(int size)
{
listSize = size;
list = new int[size];
for (size_t i = 0; i < size; i++)
list[i] = students(i);
return list;
}
int Student::students(int value)
{
int age;
switch (value)
{
case 1:
age = 5;
return age;
case 2:
age = 15;
return age;
}
}
void Student::deleteList()
{
delete[] list;
}
Student::Student() : age(0) {}
Student::Student(int age) : age(age) {}
Student::Student(const Student& student) : age(student.age) {}
Student::~Student() {}
```

Date.cpp

```
#include "Date.h"
sint Date::getDay() const { return day; }
void Date::setDay(sint day1) { day = day1; }
sint Date::getMonth() const { return month; }
void Date::setMonth(sint month1) { month = month1; }
sint Date::getYear() const { return year; }
void Date::setYear(sint year1) { year = year1; }
Date::Date() : day(2), month(2), year(2002) {}
Date::Date(sint day, sint month, sint year) : day(day), month(month), year(year) {}
Date::Date(const Date& date) : day(date.day), month(date.month), year(date.year) {}
Date::~Date() {}
```

Test.cpp

```
#include "List.h"
#include "Task.h"
int TestAddStudent(List&, int);
int TestDeleteStudent(List&, int);
int TestGetStudentID(List&, int);
int TestReadFile(List&, int);
int main()
{
{
List test;
int count = 0;
test.setListSize(2);
test.createList();
count = TestAddStudent(test, count);
count = TestDeleteStudent(test, count);
count = TestGetStudentID(test, count);
count = TestReadFile(test, count);
if (count == 4)
cout << endl << "All tests are successful" << endl;
else
cout << endl << "Not all tests are successful" << endl;
}
if (_CrtDumpMemoryLeaks())
cout << endl << "WARNING! Memory leak" << endl;
else
cout << endl << "There is no memory leak" << endl;
return 0;
```

```
}
int TestAddStudent(List& test, int count)
{
test.addStudent(CreateStudent2());
if (test.getListSize() == 3)
{
cout << endl << "Test: Add_student - successful" << endl;
count++;
}
else
cout << endl << "Test: Add_student - unsuccessful" << endl;
return count;
}
int TestDeleteStudent(List& test, int count)
{
test.deleteStudent(test.getListSize());
if (test.getListSize() == 2)
{
cout << endl << "Test: Delete_student - successful" << endl;
count++;
}
else
cout << endl << "Test: Delete_student - unsuccessful" << endl;
return count;
}
int TestGetStudentID(List& test, int count)
{
int num = test.getStudentID(2);
if (num == 1)
{
cout << endl << "Test: GetStudentId - successful" << endl;
count++;
}
else
cout << endl << "Test: GetStudentId - unsuccessful" << endl;
return count;
}
int TestReadFile(List& test, int count)
{
string filename = "Text.txt";
int a = 3;
test.ReadFile(filename,a);
```

```
string expected = "Ivanov Vasya";
string real = test.stud[0].getName();
if (expected == real)
{
cout << endl << "Test: Readfile - successful" << endl;
count++;
}
else
cout << endl << "Test: Readfile - unsuccessful" << endl;
return count;
}
```

4. Результати роботи програми

```
Base class

Index     Name         Age     Mark      Exercises       RGZ         Date
1     Petrova Katya     5       5           5           5        5  5  5555

Inheritor class

Index     Name         Age     Mark      Exercises       RGZ         Date              RGZ for Teacher
1     Petrova Katya     5       5           5           5        5  5  5555                 5

Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Show student via his count of RGZ
7.Read array from file
8.Write to file
9.Surname+Name in object
10.Sort (mark)
11.End program
1

Menu:
1.Add a new student
2.Add a new student (enter from keyboard)
3.Delete one student
4.Show all student
5.Show student via his index
6.Show student via his count of RGZ
7.Read array from file
8.Write to file
9.Surname+Name in object
10.Sort (mark)
11.End program
4
Base class

Index     Name         Age     Mark      Exercises       RGZ         Date
1     Petrova Katya     5       5           5           5        5  5  5555
2     Ivanov Petya      2       2           2           6        2  2  2222

Inheritor class

Index     Name         Age     Mark      Exercises       RGZ         Date              RGZ for Teacher
1     Petrova Katya     5       5           5           5        5  5  5555                 5
2     Ivanov Petya      2       2           2           6        2  2  2222                 6
```

Рисунок 1 – Робота класу-спадкоємця

5. Висновки

Під час виконання лабораторної роботи було додано клас-спадкоємець, котрий поширює функціонал базового класу відповідно до індивідуального завдання, а саме: має додаткове поле RGZ for Teacher; додано ще один клас-список, що керує лише елементами класу-спадкоємця.

Програма протестована, виконується без помилок, витоків пам’яті немає.