МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХПІ»

Кафедра «Обчислювальна техніка та програмування»

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ.119а.

Розробники
Виконав:
студент групи КІТ-119а
_____/ Момот Р.Є./

Перевірив:
_____/аспірант Бартош М. В./

Харків 2020

ЗАТВЕРДЖЕНО

КІТ.119а.

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ.119а.

Аркушів _36_

Харків 2020

ЗМІСТ

ВСТУП

Поняття «Інформаційна система»

Інформаційно-довідкові системи – це сукупність організаційних і технічних засобів, що призначені для керування базами даних і використовуються, наприклад, для ведення статистики, складання каталогів тощо. Вони полегшують оперування великими об’ємами професійно цінної інформації, виступаючи як засіб надійного збереження професійних знань, забезпечуючи зручний і швидкий пошук необхідних відомостей.

Призначення та галузь застосування

Призначення розробки – оперування даними про прикладну галузь література, а саме про підручники. Розроблена з використанням ієрархії класів програма дозволяє виконувати такі завдання: читання даних з файлу та їх запис у контейнер, запис даних з контейнера у файл, сортування елементів у контейнері за вказаними критеріями (поле та напрям задаються користувачем з клавіатури), виконання особистого завдання. Також було розроблено декілька інших класів, які слугують для: відображення діалогового меню, тестування розроблених методів класу, сортування.

Постановка завдання до розробки

В основі функціонування інформаційно-довідкових систем лежить обробка інформації. Режими її обробки можуть бути такими: пакетний, діалоговий, реального часу.

Пакетний режим визначає операції та їх послідовність з формування даних в ЕОМ і формування розрахунків безпосередньо на обчислювальному центрі чи відповідною системою.

Діалоговий режим забезпечує безпосередню взаємодію користувача з системою. Ініціатором діалогу може бути як користувач, так і ЕОМ. В останньому випадку на кожному кроці користувачу повідомляється, що треба робити.

Режим реального часу — режим обробки інформації системою при взаємодії з зовнішніми процесами в темпі ходу цих процесів.

В роботі буде реалізовано діалоговий режим обробки інформації, де ініціатором виступає ЕОМ.

Дані, що обробляються, в оперативній пам’яті можуть зберігатися у вигляді масиву або лінійного (одно- або двонаправленого) списку.

До переваг масиву можна віднести:

1. Ефективність при звертанні до довільного елементу, яке відбувається за постійний час O(1),
2. Можливість компактного збереження послідовності їх елементів в локальній області пам'яті, що дозволяє ефективно виконувати операції з послідовного обходу елементів таких масивів.
3. Масиви є дуже економною щодо пам'яті структурою даних.

До недоліків:

1. Операції, такі як додавання та видалення елементу, потребують часу O(n), де n — розмір масиву.
2. У випадках, коли розмір масиву є досить великий, використання звичайного звертання за індексом стає проблематичним.
3. Масиви переважно потребують непреривної області для зберігання.

До переваг списку можна віднести:

1. Списки досить ефективні щодо операцій додавання або видалення елементу в довільному місці списку, виконуючи їх за постійний час.
2. В списках також не існує проблеми «розширення», яка рано чи пізно виникає в масивах фіксованого розміру, коли виникає необхідність включити в нього додаткові елементи.
3. Функціонування списків можливо в ситуації, коли пам'ять комп'ютера фрагментована.

До недоліків:

1. Для доступу до довільного елементу необхідно пройти усі елементи перед ним.
2. Необхідність разом з корисною інформацією додаткового збереження інформації про вказівники, що позначається на ефективності використання пам'яті цими структурами.

Виходячи з переваг та недоліків зазначених вище в розроблюваній програмі для подання даних буде реалізовано вектор, який є абстрактною моделлю, що імітує динамічний масив.

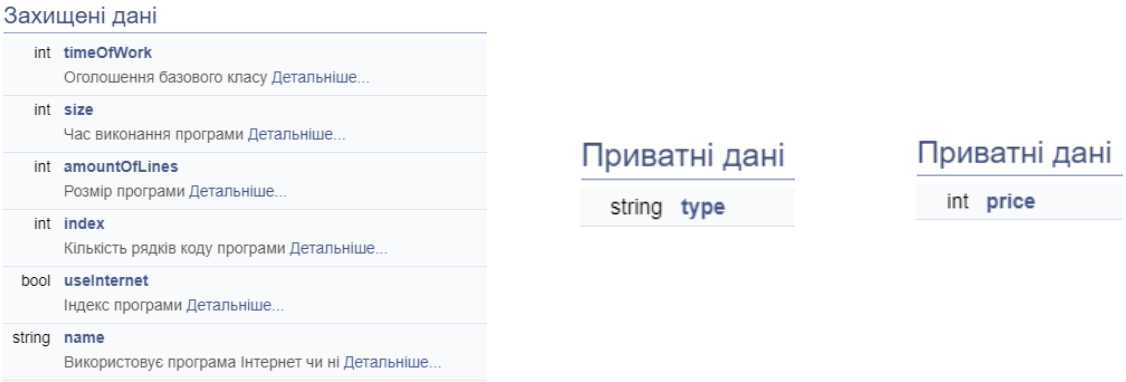
Для реалізації поставленого завдання було обрано об’єктно-орієнтовану мову програмування C++, через те, що вона засновує програми як сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. А середовищем програмування – Microsoft Visual Studio.

РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ

Розробка алгоритмів програми

При розробленні структур даних було створено: базовий клас *CProgram* та класи-спадкоємці *CMalware* та *CProgramForSale*, які наслідують поля базового класу.

На рис. 1 показано внутрішню структуру, а на рис. 2 - відносини розроблених класів у вигляді UML-діаграми.



а) б) в)

Рис. 1. Поля базового класу та класів-спадкоємців (рис. 1а-в)

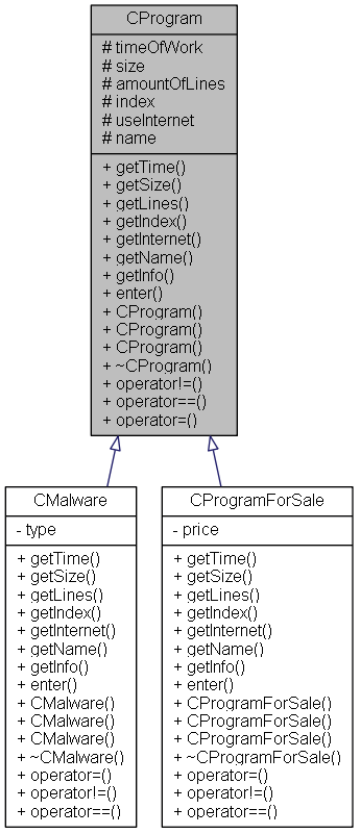


Рис. 2. Схема ієрархії розроблених класів

Дані про програми будуть заноситися до списку. Для цього було розроблено клас-контролер *CList* з полями, показаними на рис. 3, і методами на рис. 4.

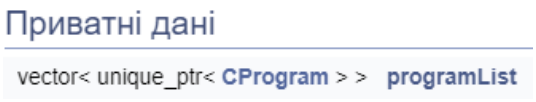
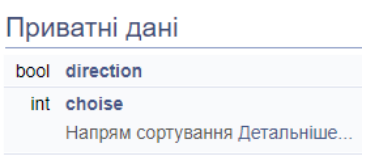


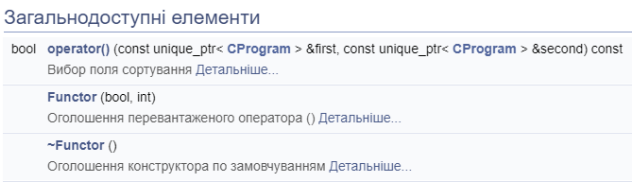
Рис. 3. Поля класу-контролеру *CList*

Рис. 4. Розроблені методи класу *CList*

Так як метод сортування було реалізовано за допомогою функтора, його поля та методи можна побачити на рис. 5а-б.



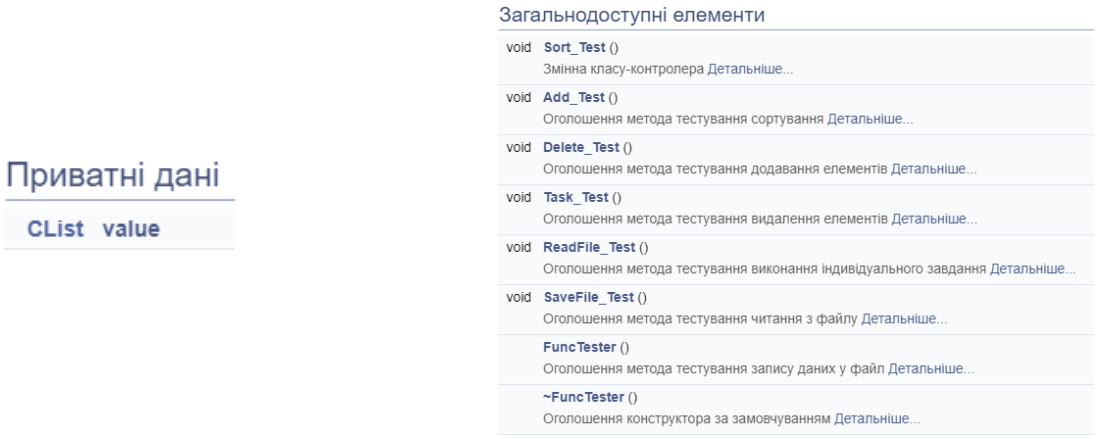
а)



б)

Рис. 5. Поля та методи класу-функтора

Також відповідно до додаткового завдання було створено клас-тестер, який слугує виконує функції тестування основних методів класу-контролера. Його поля та методи можна побачити на рис. 6а-б.



а) б)

Рис. 6. Поля та методи класу-тестеру

На рис. 7 подано структуру проекту розробленого програмного продукту.

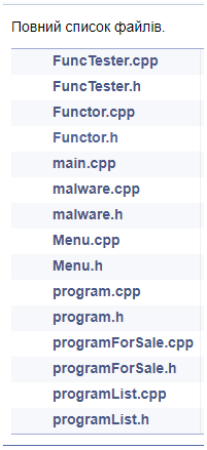


Рис. 7. Структура проекту

СХЕМИ АЛГОРИТМУ ПРОГРАМИ

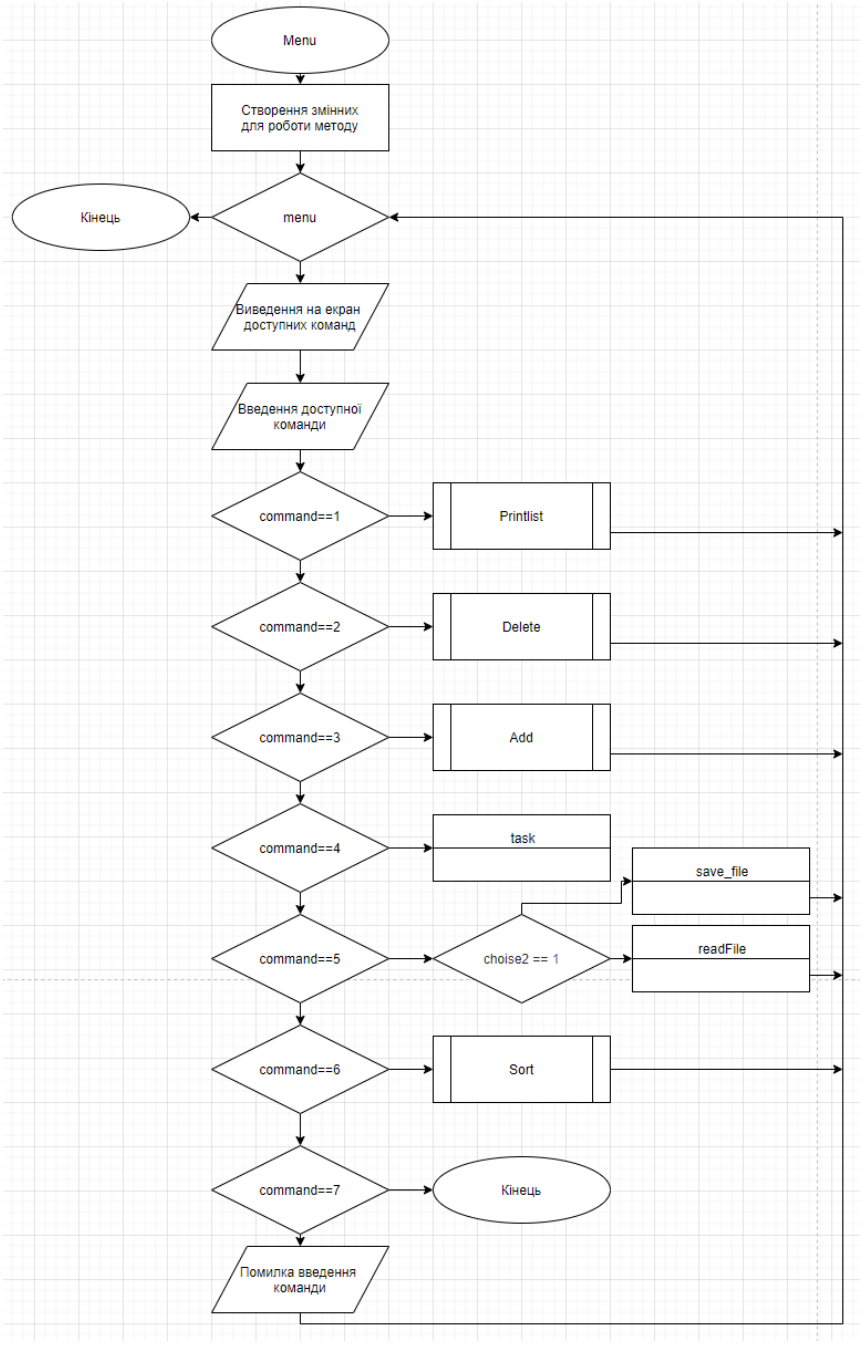
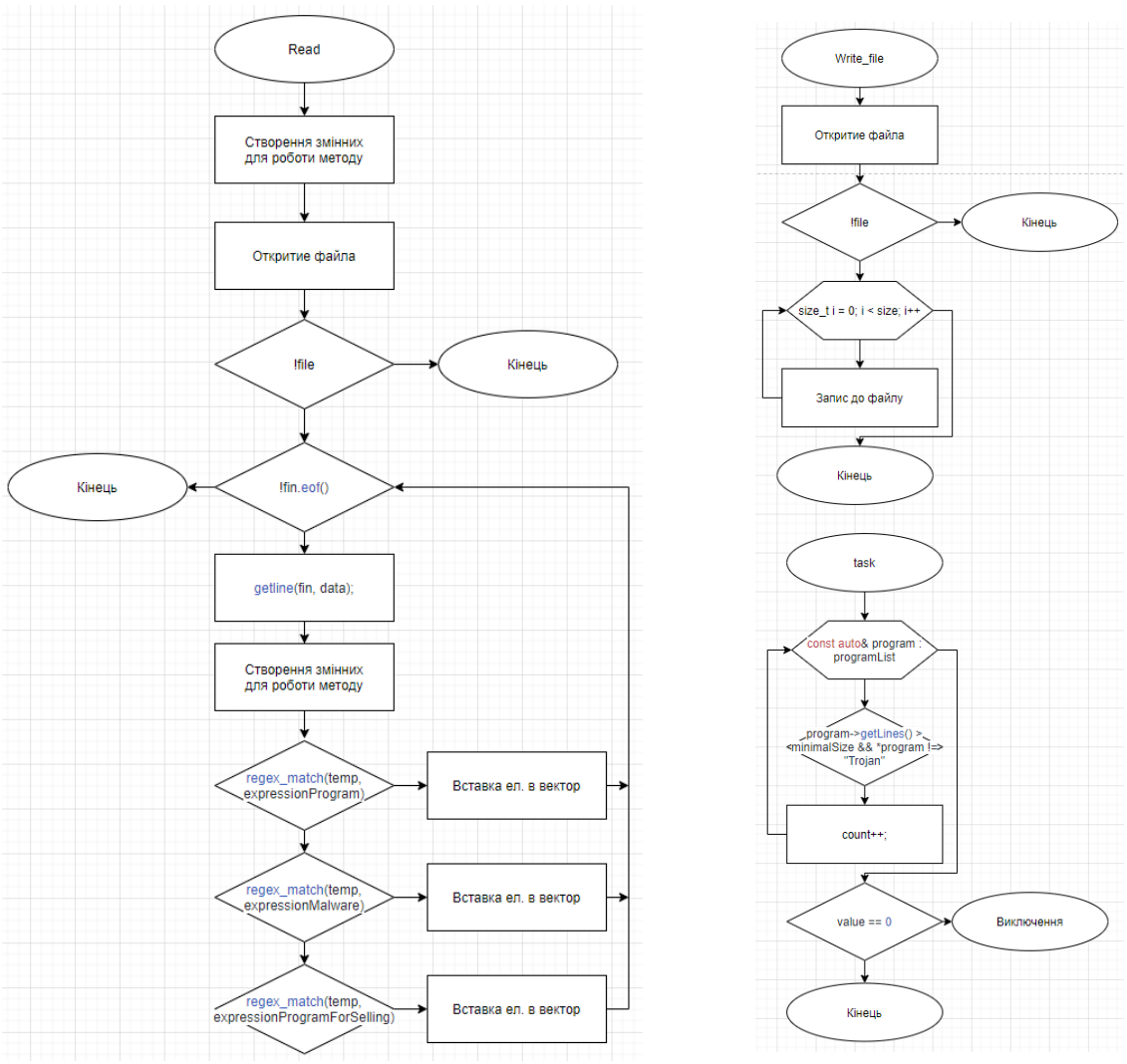


Рис. 8. Схема алгоритму методу Menu



а) б)

Рис. 9. Схеми алгоритмів методів ReadFile, WriteFile, Task

ВИСНОВОК

У результаті розробки інформаційно-довідкової системи було виконано наступні завдання:

1. Досліджено літературу стосовно прикладної галузі та оформлено аналітичний розділ пояснювальної записки;
2. Для прикладної галузі література розроблено розгалужену ієрархію класів, що складається з трьох класів – один «батьківський», два спадкоємці. У них було перевантажено оператори введення-виведення та оператор порівняння;
3. Розроблено клас-контролер, що включає колекцію розроблених класів, та наступні методи роботи з цією колекцією:
    а) читання даних з файлу та їх запис у контейнер;
    б) запис даних з контейнера у файл;
    в) сортування елементів у контейнері за вказаними критеріями: поле та напрям сортування, які задаються користувачем з клавіатури;
    г) Виконання індивідуального завдання;
    д) Додавання елементів у контейнер;
    е) Видалення елементів з контейнеру;
4. Розроблено клас, який відображає діалогове меню для демонстрації реалізованих функцій класу контролера;
5. Оформлено схеми алгоритмів функцій класів контролера та діалогового меню;
6. Оформлено документацію;
7. Було додано обробку помилок, перевірку вхідних даних за допомогою регулярних виразів;
8. Розроблено клас-тестер, що перевіряє методи класу-контролера на коректність.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Шілдт, Герберт. Повний довідник по C++, 4-е видання .: Пер. з англ .: - М .: Изд. будинок «Вільямс», 2004. - 800 с .;
2. Дейтел Х.М. Як програмувати на C++ / Х.М. Дейтел, П.Дж. Дейтел М.: ЗАТ БІНОМ, 1999. - 1000 с.
3. Штейн Кліфорд (2019). Алгоритми. Побудова і аналіз.
4. Вандервуд, Джосаттіс - Шаблони C++. Довідник розробника. / Пер. з англ. - М .: Вільямс, 2008. - 536 с.
5. Андрій Александреску, Сучасне проектування на C++. М.: ТОВ «І.Д.Вільямс», 2002.
6. Страуструп Б. Дизайн і еволюція C++ / Б. Страуструп; пер. з англ. - М. : ДМК Прес; С.Пб: Пітер, 2007. - 445 с.
7. Остерн. Узагальнене програмування і STL: Використання інаращіваніе стандартної бібліотеки шаблонів C++ / Остерн; Пер. Санглена. - С.Пб: Невський Діалект, 2004. - 544 с.

Додаток А

Текст програми

programList.h

```
/**
* @file programList.h
* Файл оголошення класу-контролера.
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#pragma once
#include "programForSale.h"       /** Підключення файлу programForSale.h */
#include "malware.h"              /** Підключення файлу malware.h */
#include "Functor.h"              /** Підключення файлу Functor.h */

class CList {                     /** Оголошення класу-контролера */
private:
	vector <unique_ptr<CProgram>> programList;/** Контейнер для зберігання елементів */

public:
	void PrintList() const noexcept;  /** Оголошення методу виводу списку у консоль */
	int Task(int) const;/** Оголошення методу виконання індивідуального завднання */
	int AddProgram(int);/** Оголошення методу додавання нового елемента у список */
	int DeleteProgram(int);    /** Оголошення методу видалення програми зі списку */
	void Sort(Functor&) noexcept;            /** Оголошення методу сортування даних */
	int SaveFile(string) const;       /** Оголошення методу виведення даних у файл */
	int ReadFile(string);             /** Оголошення методу виводу списку у консоль */
	int GetIndex(int) const;   /** Оголошення методу отримання індекса за номером */

	CList();      /** Оголошення конструктора за замовчуванням */
	~CList();     /** Оголошення деструктора */
};
```

programForSale.h

```
/**
* @file programForSale.h
* Файл оголошення класу-спадкоємця.
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#pragma once
#include "program.h"                            /** Підключення файлу program.h */
class CProgramForSale final : public CProgram   /** Оголошення класу спадкоємця */
{
private:
	int price;          /** Ціна програми */

public:
	int getTime() const override;/** Оголошення перевантаженого гетера отримання часу
виконання програми */
	int getSize() const override;/** Оголошення перевантаженого гетера отримання розміру
програми */
	int getLines() const override;/** Оголошення перевантаженого гетера отримання
кількості рядків коду програми */
	int getIndex()const override;/** Оголошення перевантаженого гетера отримання індексу
програми */
	bool getInternet()const override; /** Оголошення перевантаженого гетера отримання
часу виконання програми */
```

```
	string getName() const override;  /** Оголошення перевантаженого гетера отримання назви програми */

	string getInfo() const override;  /** Оголошення перевантаженого метода отримання інформації програми */
	void enter(istream&) override;/** Оголошення перевантаженого метода вводу інформації програми */

	CProgramForSale();   /** Оголошення конструктора за замовчуванням */
	CProgramForSale(bool, int, int, int, int, string, int);/** Оголошення конструктора з параметрами */
	CProgramForSale(const CProgramForSale&); /** Оголошення конструктора копіювання */
	~CProgramForSale() override;      /** Оголошення перевантаженого деструктора */

	CProgramForSale& operator= (CProgramForSale&);/** Оголошення оператора присвоювання*/
	bool operator!=(const string) const override;   /** Оголошення перевантаженого оператора нерівності */
	bool operator==(const int) const override;/** Оголошення перевантаженого оператора порівняння */
};
```

program.h

```
/**
 * @file program.h
 * Підключення необхідних бібліотек та оголошення класу CProgram.
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */


#pragma once

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

#include <string>           /** Підключення бібліотеки string*/
#include <iostream>         /** Підключення бібліотеки iostream*/
#include <iomanip>          /** Підключення бібліотеки iomanip*/
#include <locale>           /** Підключення бібліотеки locale*/
#include <fstream>          /** Підключення бібліотеки fstream*/
#include <sstream>          /** Підключення бібліотеки sstream*/
#include <regex>            /** Підключення бібліотеки regex*/
#include <memory>           /** Підключення бібліотеки memory*/
#include <vector>           /** Підключення бібліотеки vector*/
#include <exception>        /** Підключення бібліотеки exception*/
#include <iterator>         /** Підключення бібліотеки 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::istream;
using std::ostream;
using std::ofstream;
using std::stringstream;
using std::istringstream;
using std::regex;
```

```
using std::regex_match;
using std::regex_search;
using std::regex_replace;
using std::cmatch;
using std::unique_ptr;
using std::vector;
using std::exception;
using std::iterator;

class CProgram {                    /** Оголошення базового класу*/
protected:
	int timeOfWork;             /** Час виконання програми*/
	int size;                   /** Розмір програми*/
	int amountOfLines;          /** Кількість рядків коду програми*/
	int index;                  /** Індекс програми*/
	bool useInternet;           /** Використовує програма Інтернет чи ні*/
	string name;                /** Назва програми*/

public:
	virtual int getTime() const;      /** Оголошення віртуального гетера отримання часу
роботи програми*/
	virtual int getSize() const;      /** Оголошення віртуального гетера отримання
розміру програми*/
	virtual int getLines() const;     /** Оголошення віртуального гетера отримання
кількості рядків програми*/
	virtual int getIndex()const;      /** Оголошення віртуального гетера отримання
індексу програми*/
	virtual bool getInternet()const;  /** Оголошення віртуального гетера отримання часу
роботи програми*/
	virtual string getName() const;   /** Оголошення віртуального гетера отримання назви
програми*/

	virtual string getInfo() const;   /** Оголошення віртуальної функції отримання
інформації програми*/
	virtual void enter(istream&);

	CProgram();                       /** Оголошення конструктора по замовчуванням*/
	CProgram(bool, int, int, int, int, string);     /** Оголошення конструктора з
параметрами*/
	CProgram(const CProgram& other);  /** Оголошення конструктора копіювання*/
	virtual ~CProgram();              /** Оголошення віртуального деструктора*/

	friend ofstream& operator<< (ofstream&,const CProgram&);     /** Оголошення
перевантаженого оператора виводу у файл*/
	friend ostream& operator<< (ostream&, const CProgram&);      /** Оголошення
перевантаженого оператора виводу у консоль*/
	friend istream& operator>> (istream&, CProgram&);            /** Оголошення
перевантаженого оператора вводу*/
	virtual bool operator!=(const string) const;                 /** Оголошення
віртуального перевантаженого оператора нерівності*/
	virtual bool operator==(const int) const;/** Оголошення віртуального перевантаженого
оператора порівняння*/
	CProgram& operator= (CProgram&);  /** Оголошення перевантаженого оператора
присвоювання*/
};
```

Menu.h

```
/**
* @file Menu.h
* Файл оголошення класу меню
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/
```

```
#pragma once
#include "programList.h"    /** Підключення файлу programList.h */
class Menu
{
public:
	void menu() const;   /** Оголошення методу роботи зі списком*/

	Menu();              /** Оголошення конструктора за замовчуванням */
	~Menu();             /** Оголошення деструктором */
};
```

malware.h

```
/**
* @file malware.h
* Файл оголошення класу спадкоємця
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#pragma once
#include "program.h"        /** Підключення файлу program.h */
class CMalware final: public CProgram           /** Оголошення класу спадкоємця */
{
private:
	string type;                                /** Підключення файлу program.h */

public:
	int getTime() const override;/** Оголошення перевантаженого гетера часу виконання
програми */
	int getSize() const override;/** Оголошення перевантаженого гетера розміру програми
*/
	int getLines() const override;/** Оголошення перевантаженого гетера кількості рядків
коду програми */
	int getIndex()const override;/** Оголошення перевантаженого гетера індексу програми
*/
	bool getInternet()const override; /** Оголошення перевантаженого гетера Інтернету
програми */
	string getName() const override;  /** Оголошення перевантаженого гетера назви
програми */

	string getInfo() const override;  /** Оголошення перевантаженого метода отримання
інформації програми */
	void enter(istream&) override;           /** Оголошення перевантаженого метода
введення інформації програми */

	CMalware();                         /** Оголошення конструктора за замовчуванням */
	CMalware(bool, int, int, int, int, string, string);   /** Оголошення конструктора з
параметрами */
	CMalware(const CMalware&);              /** Оголошення конструктора копіювання */
	~CMalware() override;               /** Оголошення перевантаженного деструктора */

	CMalware& operator= (CMalware&);  /** Оголошення перевантаженого оператора
присвоювання */
	bool operator!=(const string) const override;   /** Оголошення перевантаженого
оператора нерівності */
	bool operator==(const int) const override;      /** Оголошення перевантаженого
оператора порівняння */
};
```

Functor.h

```
/**
* @file Functor.h
* Файл оголошення класу, який виконує функції функтора
```

```
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#pragma once
#include "program.h"/** Підключення файлу program.h */

class Functor              /** Оголошення класу функтора*/
{
private:
	bool direction;      /** Напрям сортування*/
	int choise;          /** Вибор поля сортування*/

public:
	bool operator()(const unique_ptr<CProgram>& first, const unique_ptr<CProgram>&
second) const;             /** Оголошення перевантаженого оператора () */

	Functor(bool, int);  /** Оголошення конструктора по замовчуванням*/
	~Functor();          /** Оголошення деструктора*/
};
```

FuncTester.h

```
/**
* @file FuncTester.h
* Файл оголошення класу-тестера
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#pragma once
#include "programList.h"          /** Підключення файлу programList.h */
class FuncTester                  /** Оголошення класу-тестера */
{
private:
	CList value;              /** Змінна класу-контролера */

public:
	void Sort_Test();         /** Оголошення метода тестування сортування */
	void Add_Test();          /** Оголошення метода тестування додавання елементів */
	void Delete_Test();       /** Оголошення метода тестування видалення елементів */
	void Task_Test();         /** Оголошення метода тестування виконання індивідуального
завдання */
	void ReadFile_Test();     /** Оголошення метода тестування читання з файлу */
	void SaveFile_Test();     /** Оголошення метода тестування запису даних у файл */

	FuncTester();             /** Оголошення конструктора за замовчуванням */
	~FuncTester();            /** Оголошення деструктора */
};
```

programList.cpp

```
/**
* @file programList.cpp
* Файл реалізації методів класу CList
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/
```

```
#include "programList.h"    /** Підключення файлу programList.h */

CList::~CList()             /** Реалізація деструктора */
{
	programList.clear(); /** Очищення пам'яті масива програм */
}
CList::CList()              /** Реалізація конструктора за замовчуванням */
{
	for (size_t i = 0; i < 5; i++)          /** Масив, який заповнює вектор елементами
*/
	{
		if (i == 0)
			programList.emplace_back(new CProgram());
		else if (i == 1)
			programList.emplace_back(new CMalware(1, 8800, 555, 35, 35634,
"BestMalware", "Exploit"));
		else if (i == 2)
			programList.emplace_back(new CProgram(0, 423, 20, 654, 53453,
"Calculator"));
		else if (i == 3)
			programList.emplace_back(new CProgramForSale(0, 345, 789, 423, 67456,
"MoneyStealer", 99999));
		else
			programList.emplace_back(new CMalware());
	}
}
void CList::PrintList() const noexcept   /** Реалізація методу виведення списку на екран */
{
	try
	{
		if (programList.size() == 0)            /** Перевірка розміру списка */
			throw exception("Список пуст.");

		int value = 1;

		cout << endl << setiosflags(ios::left);
		cout << setw(12) << "  Время" << setw(12) << "Размер";
		cout << setw(10) << "Строки" << setw(11) << "Интернет";
		cout << setw(14) << "Индекс" << setw(17) << "Название";
		cout << setw(14) << "Тип/Цена" << endl;  /** Виведення назв полів класів */

		for_each(programList.begin(), programList.end(), [&value](const
unique_ptr<CProgram>& program)    /** Цикл, який виводить список елементів */
			{
				cout << value << ". " << *program << endl;      /** Виведення
номеру та полів елемента */
				value++;            /** Збільшення зміної нумерування */
			});

		cout << endl;
	}
	catch (const std::exception& ex)
	{
		cout << ex.what() << endl;
	}
}
int CList::Task(int minimalSize) const   /** Реалізація метода виконання індивідуального
завдання */
{
	int value = 1;             /** Оголошення змінної кількості відповідних елементів */

	for (const auto& program : programList)/** Цикл який обходить усі елементи масиву */
	{
		if (program->getLines() > minimalSize && *program != "Trojan")/** Перевірка
полів елементів */
```

```
		{
			cout << value << ". " << *program << endl;	/** Виведення елемента на екран */
			value++;	/** Збільшення змінної нумерування масиву */
		}
	}

	if (value == 0)		/** Якщо елементів немає */
		throw exception("Программы с такими параметрами отсутствуют.");

	return value;	/** Повернення кількості відповідних елементів */
}
int CList::AddProgram(int value)  /** Реалізація метода додавання нової програми */
{
	if (value == 1)
	{
		CProgram* temp = new CProgram;	/** Створення елемента класу CProgram */
		programList.emplace_back(temp);	/** Вставка елемента у масив */
	}
	else if (value == 2)
	{
		CProgram* temp = new CMalware;	/** Створення елемента класу CMalware */
		programList.emplace_back(temp);	/** Вставка елемента у масив */
	}
	else if (value == 3)
	{
		CProgram* temp = new CProgramForSale;	/** Створення елемента класу CProgramForSale */
		programList.emplace_back(temp);		/** Вставка елемента у масив */
	}
	else
		throw exception("Неверная команда.");

	return programList.size();		/** Повернення розміру списку */
}
int CList::DeleteProgram(int value)		/** Реалізація метода видалення програми */
{
	if (programList.size() == 0)		/** Перевірка розміру списку */
		throw exception("Список программ пуст.");

	int number = -1;
	bool findEl = false;
	std::vector<unique_ptr<CProgram>>::iterator it; /** Створення ітератора */

	for (const auto& program : programList)  /** Цикл, який обробляє усі елементи колекції */
	{
		if (program->getIndex() == value) /** Якщо програма має потрібний індекс */
		{
			number++;			/** Збільшення змінної номеру елементу */
			findEl = true;			/** Змінення зміної знаходження елемента */
			break;				/** Зупинка роботи циклу */
		}
		else
			number++;			/** Збільшення змінної номеру елементу */
	}

	if (findEl)		/** Якщо елемент з потрібним індексом був найден */
	{
		it = programList.begin();	/** Встановлення покажчика на початок списку */
		advance(it, number);		/** Переміщення покажчика на потрібний елемент */
		programList.erase(it);		/** Видалення потрібного елементу */

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

```
	else
		throw exception("Элемент не найден.");

	return programList.size();		/** Повернення змінної розміру списку */
}
void CList::Sort(Functor& choise) noexcept	/** Реалізація метода сортування списку */
{
	std::sort(programList.begin(), programList.end(), choise);	/** Сортування списку */
}
int CList::SaveFile(string filename) const	/** Реалізація метода запису даних у файл */
{
	if (programList.size() == 0)	/** Перевірка розміру масиву */
		throw exception("Список пуст.");

	ofstream fout(filename);		/** Відкриття файлу */
	if (!fout.is_open())			/** Перевірка чи відкритий файл */
		throw exception("Невозможно открыть файл.");

	fout << setiosflags(ios::left);
	fout << setw(12) << "  Время" << setw(12) << "Размер";
	fout << setw(10) << "Строки" << setw(11) << "Интернет";
	fout << setw(14) << "Индекс" << setw(17) << "Название";
	fout << setw(14) << "Тип/Цена" << endl;  /** Виведення назв полів */

	int value = 1;
	for (const auto& program : programList)/** Цикл, який обробляє усі елементи масиву */
	{
		fout << setiosflags(ios::left) << setw(2) << value << ". ";  /** Виведення
нумерування елементів */
		fout << *program << endl;  /** Виведення полів елементів */
		value++;
	}

	fout.close();		/** Закриття файлу */
	return value - 1;	/** Повернення кількості елементів, які були збережені у файл */
}
int CList::ReadFile(string filename)	/** Реалізація метода читання даних з файлу */
{
	regex expressionProgram("([\\d]* [\\d]* [\\d]* [\\d]* [1|0]* [A-ZA-Я]+[\\wA-Яа-я,.;:-
]* [\\wA-Яа-я,.;:-]*)");	/** Регулярний вираз для класу CProgram */
	regex expressionMalware("([\\d]* [\\d]* [a-zA-ZA-Яа-я]* [\\d]* [\\d]* [1|0]* [A-ZA-
Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]*)");	/** Регулярний вираз для класу CMalware */
	regex expressionProgramForSelling("([\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [1|0]* [A-ZA-
Я]+[\\wA-Яа-я,.;:-]* [\\wA-Яа-я,.;:-]*)");	/** Регулярний вираз для класу
CProgramForSale */
	regex replaceSymbols("[;:-]");	/** Регулярний вираз для символів, які треба
замінити */

	string temp, data, replacement = "";
	int value = 0;
	ifstream fin(filename);		/** Відкриття файлу */
	istringstream ss;

	if (!fin.is_open())			/** Перевірка чи відкритий файл */
	{
		throw exception("Невозможно открыть файл для чтения.");
		return programList.size();
	}

	while (!fin.eof())			/** Цикл, який працює до кінця файлу */
	{
		getline(fin, data);		/** Отримання строки з файлу */
		temp = regex_replace(data, replaceSymbols, replacement);	/** Видалення
символів які не підходять */
		temp += " ";
```

```
		if (regex_match(temp, expressionProgram))/** Перевірка на відповідність до класу CProgram */
		{
			istringstream ss(temp);
			CProgram* program = new CProgram();/** Створення змінної класу CProgram */

			ss >> *program;
			programList.emplace_back(program);/** Вставка елемента у масив */
			value++;		/** Збільшення змінної кількості нових елементів */
		}
		else if (regex_match(temp, expressionMalware))  /** Перевірка на відповідність до класу CProgram */
		{
			istringstream ss(temp);
			CMalware* program = new CMalware();/** Створення змінної класу CMalware */

			ss >> *program;
			programList.emplace_back(program);	/** Вставка елемента у масив */
			value++;		/** Збільшення змінної кількості нових елементів */
		}
		else if (regex_match(temp, expressionProgramForSelling))	/** Перевірка на відповідність до класу CProgram */
		{
			istringstream ss(temp);
			CProgramForSale* program = new CProgramForSale();/** Створення змінної класу CProgramForSale */

			ss >> *program;
			programList.emplace_back(program);/** Вставка елемента у масив */
			value++;		/** Збільшення змінної кількості нових елементів */
		}
	}

	fin.close();		/** Закриття файлу */
	return value;		/** Повернення змінної кількості нових елементів */
}
int CList::GetIndex(int value) const/** Реалізація метода отримання індексу елементів */
{
	return programList[value]->getIndex();   /** Повернення індексу */
}
```

programForSale.cpp

```
/**
* @file programForSale.cpp
* Файл реалізації методів класу-спадкоємця
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "programForSale.h"		/** Підключення файлу programForSale.h */

int CProgramForSale::getTime() const	/** Реалізація геттера часу роботи програми */
{
	return timeOfWork;		/** Повернення часу роботи програми */
}
int CProgramForSale::getSize() const	/** Реалізація геттера розміру програми */
{
	return size;			/** Повернення розміру програми */
}
int CProgramForSale::getLines() const	/** Реалізація геттера кількості рядків коду програми */
{
```

```
	return amountOfLines;			/** Повернення кількості рядків коду програми */
}
int CProgramForSale::getIndex() const	/** Реалізація геттера індексу програми */
{
	return index;				/** Повернення індексу програми */
}
bool CProgramForSale::getInternet()const/** Реалізація геттера інтернету */
{
	return useInternet;			/** Повернення інтернету */
}
string CProgramForSale::getName()const	/** Реалізація геттера назви програми */
{
	return name;				/** Повернення назви програми */
}

string CProgramForSale::getInfo() const	/** Реалізація методу отримання даних програми */
{
	stringstream temp;			/** Створення змінної типу stringstream */
	temp.setf(ios::left);

	temp << setw(10) << timeOfWork << setw(12) << size
		<< setw(9) << amountOfLines << setw(12) << boolalpha << useInternet
		<< setw(11) << index << setw(20) << name
		<< setw(14) << price;		/** Запис даних у об'єкт типу stringstream */

	return temp.str();			/** Повернення даних у форматі string */
}
void CProgramForSale::enter(istream& data)	/** Реалізація перевантаженого оператора
вводу */
{
	data >> index >> timeOfWork >> price >> size >> amountOfLines >> useInternet >> name;
	/** Введення даних у об'єкт типу istream */
}

CProgramForSale::CProgramForSale(bool internet, int time, int size, int lines, int index,
string name, int price) : CProgram(internet, time, size, lines, index, name), price(price)
{}	/** Реалізація конструктора з параметрами */
CProgramForSale::CProgramForSale() : CProgram(), price(2000) {}		/** Реалізація
конструктора за замовчуванням */
CProgramForSale::CProgramForSale(const CProgramForSale& other) : CProgram(other),
price(other.price) {}		/** Реалізація конструктора копіювання */
CProgramForSale::~CProgramForSale() {}		/** Реалізація деструктора */

CProgramForSale& CProgramForSale::operator=(CProgramForSale& temp)  /** Реалізація
перевантаженого оператора присвоювання */
{
	if (this == &temp)				/** Перевірка якщо змінні однакові */
		return *this;

	CProgramForSale::operator=(temp); /** Присвоювання полів базового класу */
	int price = temp.price;				/** Присвоювання полів ціни */

	return *this;
}
bool CProgramForSale::operator!=(const string type) const	/** Реалізація перевантаженого
оператора нерівності */
{
	return true;  /** Перевірка відбувається за типом зловмисного ПО (звичайна програма
не може бути зловмисним ПО) */
}
bool CProgramForSale::operator==(const int index) const		/** Реалізація перевантаженого
оператора порівняння */
{
	return this->index == index;		/** Перевірка відбувається за індексом */
}
```

program.cpp

```
/**
* @file program.cpp
* Файл реалізації методів базового класу
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "program.h"

int CProgram::getTime() const     /** Реалізація геттера часу роботи програми */
{
	return timeOfWork;          /** Повернення часу роботи програми */
}
int CProgram::getSize() const     /** Реалізація геттера розміру програми */
{
	return size;                /** Повернення розміру програми */
}
int CProgram::getLines() const    /** Реалізація геттера кількості рядків коду програми */
{
	return amountOfLines;       /** Повернення кількості рядків коду програми */
}
int CProgram::getIndex() const    /** Реалізація геттера індексу програми */
{
	return index;               /** Повернення індексу програми */
}
bool CProgram::getInternet()const /** Реалізація геттера інтернету */
{
	return useInternet;         /** Повернення інтернету */
}
string CProgram::getName()const   /** Реалізація геттера назви програми */
{
	return name;                /** Повернення назви програми */
}

void CProgram::enter(istream& data)      /** Реалізація перевантаженого оператора вводу */
{
	data >> index >> timeOfWork >> size >> amountOfLines >> useInternet >> name;      /**
Введення даних у об'єкт типу istream */
}
string CProgram::getInfo() const  /** Реалізація методу отримання даних програми */
{
	stringstream temp;          /** Створення змінної типу stringstream */
	temp.setf(ios::left);

	temp << setw(10) << timeOfWork << setw(12) << size
		<< setw(9) << amountOfLines << setw(12) << boolalpha << useInternet
		<< setw(11) << index << setw(16) << name;       /** Запис даних у об'єкт типу
stringstream */

	return temp.str();          /** Повернення даних у форматі string */
}


CProgram::CProgram(bool internet, int time, int size, int lines, int index, string name) :
useInternet(internet), timeOfWork(time), size(size), amountOfLines(lines), index(index),
name(name) {}                           /** Реалізація конструктора з параметрами */
CProgram::CProgram() : useInternet(false), timeOfWork(200), size(200), amountOfLines(200),
index(0101), name("Basic") {}           /** Реалізація конструктора за замовчуванням */
CProgram::CProgram(const CProgram& other) : useInternet(other.useInternet),
timeOfWork(other.timeOfWork), size(other.size), amountOfLines(other.amountOfLines),
index(other.index), name(other.name) {}  /** Реалізація конструктора копіювання */
CProgram::~CProgram() {}                /** Реалізація деструктора */
```

```
ofstream& operator<< (ofstream& output, const CProgram& program)    /** Реалізація
перевантаженого оператора запису даних у файл */
{
	output << program.getInfo();	/** Виклик фкнції отримання інформації */
	return output;			/** Повернення інформації */
}
ostream& operator<< (ostream& output, const CProgram& program)	/** Реалізація
перевантаженого оператора виводу даних у консоль */
{
	output << program.getInfo();	/** Виклик фкнції отримання інформації */
	return output;			/** Повернення інформації */
}
istream& operator>> (istream& input, CProgram& program)			/** Реалізація
перевантаженого оператора вводу даних з консолі */
{
	program.enter(input);			/** Виклик метода вводу даних */
	return input;				/** Повернення інформації */
}
bool CProgram::operator!=(const string type) const		/** Реалізація перевантаженого
оператора нерівності */
{
	return true;		/** Перевірка відбувається за типом зловмисного ПО (звичайна
програма не може бути зловмисним ПО) */
}
bool CProgram::operator==(const int index) const		/** Реалізація перевантаженого
оператора порівняння */
{
	return this->index == index;	/** Перевірка відбувається за індексом */
}
CProgram& CProgram::operator= (CProgram& temp)  /** Реалізація перевантаженого оператора
присвоювання */
{
	if (this == &temp)			/** Перевірка якщо змінні однакові */
		return *this;

	int timeOfWork = temp.timeOfWork; /** Присвоювання поля часу роботи програми */
	int size = temp.size;			/** Присвоювання поля розміру програми */
	int amountOfLines = temp.amountOfLines;  /** Присвоювання поля кількості рядків
програми */
	int index = temp.index;			/** Присвоювання поля індексу програми */
	bool useInternet = temp.useInternet;	/** Присвоювання поля інтернету */
	string name = temp.name;		/** Присвоювання поля назви програми */

	return *this;			/** Повернення програми */
}
```

Menu.cpp

```
/**
* @file Menu.cpp
* Файл реалізації методів класу Menu
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "Menu.h"		/** Підключення файлу Menu.h*/

void Menu::menu() const	/** Виклик метода діалогового меню*/
{
	/** Оголошення змінних для роботи програми*/
	int choise, choise2;
	int size = -1, value;
	bool stop = true;
	string filename;	/** Назва файлу */
```

```
	CList list;		/** Список елементів */

	while (stop != 0)	/** Цикл, який працює поки користувач не захоче вийти*/
	{
		cout << endl;
		cout << "1) Вывести список на экран" << endl;
		cout << "2) Удаление элемента" << endl;
		cout << "3) Добавление элемента" << endl;
		cout << "4) Индивидуальное задание" << endl;
		cout << "5) Работа с файлами" << endl;
		cout << "6) Сортировка" << endl;
		cout << "7) Завершение работы" << endl;
		cout << "=========================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise;		/** Введення зміної вибору дії*/
		cout << endl;

		switch (choise)
		{
		case 1:

			list.PrintList();	/** Виклик методу виведення списку у консоль */

			break;

		case 2:
			cout << "Введите ID элемента, который хотите удалить: ";
			cin >> choise;		/** Введення зміної вибору дії*/
			cout << endl;

			try
			{
				list.DeleteProgram(choise);/** Виклик методу видалення елемента
*/
			}
			catch (const std::exception& ex)
			{
				cout << ex.what() << endl;
			}

			break;

		case 3:
			cout << "Выберите программу, которую хотите добавить:" << endl;
			cout << "1. Элемент класса CProgram" << endl;
			cout << "2. Элемент класса CMalware" << endl;
			cout << "3. Элемент класса CProgramForSale" << endl;
			cout << "================================" << endl;
			cout << "Ваш выбор: ";
			cin >> value; /** Введення зміної вибору дії*/

			try
			{
				list.AddProgram(value);/** Виклик методу додавання програми */
			}
			catch (const std::exception & ex)
			{
				cout << ex.what() << endl;
			}
			cout << "Программа добавлена." << endl;

			break;
		case 4:
			cout << "Индивидуальное задание:\nПолучить список программ, размер
которых больше заданного размера(например 100 Кбайт). Из списка убрать \"трояны\"" << endl;
			cout << "Введите минимальный розмер программы: ";
```

```
                cin >> value;        /** Введення зміної вибору дії*/
                cout << endl;

                try
                {
                        list.Task(value);/** Виклик методу виконання індивідуального
завдання */
                }
                catch (const std::exception& ex)
                {
                        cout << ex.what() << endl;
                }

                break;

        case 5:
                cout << "Что делать?" << endl;
                cout << "1) Запись в файл" << endl;
                cout << "2) Чтение файла" << endl;
                cout << "3) Вернуться назад" << endl;
                cout << "==================" << endl;
                cout << "Ваш выбор: ";
                cin >> choise2;             /** Введення зміної вибору дії*/
                cout << endl;

                if (choise2 == 1)
                {
                        string::size_type n;
                        cout << "Введите название файла для записи: ";
                        cin >> filename;     /** Введення назви файлу */
                        cout << endl;

                        n = filename.find(".txt");
                        if (n > 187) filename += string(".txt");

                        try
                        {
                                list.SaveFile(filename);/** Виклик методу запису списка у
файл */
                        }
                        catch (const std::exception & ex)
                        {
                                cout << ex.what() << endl;
                        }

                        cout << "Запись завершена. " << endl;
                }
                else if (choise2 == 2)
                {
                        string::size_type n;
                        cout << "Введите название файла для чтения: ";
                        cin >> filename;     /** Введення назви файлу*/
                        cout << endl;

                        n = filename.find(".txt");
                        if (n > 187) filename += string(".txt");

                        try
                        {
                                list.ReadFile(filename);/** Виклик методу читання даних з
файлу */
                        }
                        catch (const std::exception & ex)
                        {
                                cout << ex.what() << endl;
                        }
```

```
                        cout << "Чтение файла завершено." << endl;

                }
                else if (choise2 == 3)
                        cout << "Возвращение назад. " << endl;
                else
                        cout << "Неверная команда. Повторите попытку." << endl;

                break;

        case 6:
                cout << "Сортировать по: " << endl;
                cout << "1) Возрастанию" << endl;
                cout << "2) Убыванию" << endl;
                cout << "3) Вернуться назад" << endl;
                cout << "==================" << endl;
                cout << "Ваш выбор: ";
                cin >> choise2;             /** Введення зміної напряму сортування*/
                cout << endl;

                if (choise2 == 1 || choise2 == 2)
                {
                        cout << "По какому полю сортировать: " << endl;
                        cout << "1) Название" << endl;
                        cout << "2) Индекс" << endl;
                        cout << "3) Количество строк кода" << endl;
                        cout << "4) Размер программы" << endl;
                        cout << "5) Время выполнения кода" << endl;
                        cout << "6) Использует ли программа интернет или нет" << endl;
                        cout << "============================" << endl;
                        cout << "Ваш выбор: ";
                        cin >> value;
                        cout << endl;

                        if (value > 0 && value <= 6)
                        {
                                Functor funct(choise2 - 1, value);/** Оголошення змінної
класу Functor */
                                list.Sort(funct);    /** Виклик методу сортування */

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

                break;
        case 7:
                cout << "Завершение работы." << endl;
                stop = false;
                break;

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

        }

    }

    return;             /** Завершення роботи метода */
```

```
}

Menu::Menu() {}      /** Реалізація конструктора за замовчуванням */
Menu::~Menu() {}     /** Реалізація деструктора */
```

malware.cpp

```
/**
* @file Menu.cpp
* Файл реалізації методів класу Menu
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "Menu.h"           /** Підключення файлу Menu.h*/

void Menu::menu() const     /** Виклик метода діалогового меню*/
{
	/** Оголошення змінних для роботи програми*/
	int choise, choise2;
	int size = -1, value;
	bool stop = true;
	string filename;     /** Назва файлу */
	CList list;          /** Список елементів */

	while (stop != 0)    /** Цикл, який працює поки користувач не захоче вийти*/
	{
		cout << endl;
		cout << "1) Вывести список на экран" << endl;
		cout << "2) Удаление элемента" << endl;
		cout << "3) Добавление элемента" << endl;
		cout << "4) Индивидуальное задание" << endl;
		cout << "5) Работа с файлами" << endl;
		cout << "6) Сортировка" << endl;
		cout << "7) Завершение работы" << endl;
		cout << "=========================" << endl;
		cout << "Ваш выбор: ";
		cin >> choise;       /** Введення зміної вибору дії*/
		cout << endl;

		switch (choise)
		{
		case 1:

			list.PrintList();    /** Виклик методу виведення списку у консоль */

			break;

		case 2:
			cout << "Введите ID элемента, который хотите удалить: ";
			cin >> choise;       /** Введення зміної вибору дії*/
			cout << endl;

			try
			{
				list.DeleteProgram(choise);/** Виклик методу видалення елемента
*/
			}
			catch (const std::exception& ex)
			{
				cout << ex.what() << endl;
			}
```

```
                        break;

                case 3:
                        cout << "Выберите программу, которую хотите добавить:" << endl;
                        cout << "1. Элемент класса CProgram" << endl;
                        cout << "2. Элемент класса CMalware" << endl;
                        cout << "3. Элемент класса CProgramForSale" << endl;
                        cout << "==================================" << endl;
                        cout << "Ваш выбор: ";
                        cin >> value; /** Введення зміної вибору дії*/

                        try
                        {
                                list.AddProgram(value);/** Виклик методу додавання програми */
                        }
                        catch (const std::exception & ex)
                        {
                                cout << ex.what() << endl;
                        }
                        cout << "Программа добавлена." << endl;

                        break;
                case 4:
                        cout << "Индивидуальное задание:\nПолучить список программ, размер
которых больше заданного размера(например 100 Кбайт). Из списка убрать \"трояны\"" << endl;
                        cout << "Введите минимальный розмер программы: ";
                        cin >> value;        /** Введення зміної вибору дії*/
                        cout << endl;

                        try
                        {
                                list.Task(value);    /** Виклик методу виконання індивідуального
завдання */
                        }
                        catch (const std::exception& ex)
                        {
                                cout << ex.what() << endl;
                        }

                        break;

                case 5:
                        cout << "Что делать?" << endl;
                        cout << "1) Запись в файл" << endl;
                        cout << "2) Чтение файла" << endl;
                        cout << "3) Вернуться назад" << endl;
                        cout << "==================" << endl;
                        cout << "Ваш выбор: ";
                        cin >> choise2;             /** Введення зміної вибору дії*/
                        cout << endl;

                        if (choise2 == 1)
                        {
                                string::size_type n;
                                cout << "Введите название файла для записи: ";
                                cin >> filename;     /** Введення назви файлу */
                                cout << endl;

                                n = filename.find(".txt");
                                if (n > 187) filename += string(".txt");

                                try
                                {
                                        list.SaveFile(filename);    /** Виклик методу запису списка
у файл */
                                }
```

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

                        cout << "Запись завершена. " << endl;
                }
                else if (choise2 == 2)
                {
                        string::size_type n;
                        cout << "Введите название файла для чтения: ";
                        cin >> filename;     /** Введення назви файлу*/
                        cout << endl;

                        n = filename.find(".txt");
                        if (n > 187) filename += string(".txt");

                        try
                        {
                                list.ReadFile(filename);    /** Виклик методу читання даних
з файлу */
                        }
                        catch (const std::exception & ex)
                        {
                                cout << ex.what() << endl;
                        }

                        cout << "Чтение файла завершено." << endl;

                }
                else if (choise2 == 3)
                        cout << "Возвращение назад. " << endl;
                else
                        cout << "Неверная команда. Повторите попытку." << endl;

                break;

        case 6:
                cout << "Сортировать по: " << endl;
                cout << "1) Возрастанию" << endl;
                cout << "2) Убыванию" << endl;
                cout << "3) Вернуться назад" << endl;
                cout << "==================" << endl;
                cout << "Ваш выбор: ";
                cin >> choise2;             /** Введення зміної напряму сортування*/
                cout << endl;

                if (choise2 == 1 || choise2 == 2)
                {
                        cout << "По какому полю сортировать: " << endl;
                        cout << "1) Название" << endl;
                        cout << "2) Индекс" << endl;
                        cout << "3) Количество строк кода" << endl;
                        cout << "4) Размер программы" << endl;
                        cout << "5) Время выполнения кода" << endl;
                        cout << "6) Использует ли программа интернет или нет" << endl;
                        cout << "============================" << endl;
                        cout << "Ваш выбор: ";
                        cin >> value;
                        cout << endl;

                        if (value > 0 && value <= 6)
                        {
                                Functor funct(choise2 - 1, value);/** Оголошення змінної
класу Functor */
                                list.Sort(funct);    /** Виклик методу сортування */
```

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

                break;
        case 7:
                cout << "Завершение работы." << endl;
                stop = false;
                break;

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

        }

    }

    return;       /** Завершення роботи метода */
}

Menu::Menu() {}      /** Реалізація конструктора за замовчуванням */
Menu::~Menu() {}     /** Реалізація деструктора */
```

main.cpp

```
/**
 * @mainpage
 * <b> Розрахункове завдання. <br/></b>
 * <b><i>Індивідуальне завдання: Отримати список програм, розмір яких більше заданого
розміру (наприклад 100 Кбайт). Зі списку виключити "трояни"</i></b><br/>
 * @author Momot Roman
 * @date 2020.05.26
 * @version 1.0
 */

/**
* @file main.cpp
* Файл реалізації методів класу Menu
*/

#include "programList.h"    /** Підключення файлу programList.h */
#include "Menu.h"           /** Підключення файлу Menu.h */
#include "FuncTester.h"     /** Підключення файлу FuncTester.h */

void main()
{
    setlocale(LC_ALL, "Rus");  /** Русифікація консолі */
    Menu menu;                 /** Створення змінної класу Menu */

    menu.menu();               /** Виклик методу меню */

    cout << endl << "Тестирование функций с помощью класса-тестировщика" << endl;
    FuncTester tester;         /** Створення змінної класу FuncTester */
    tester.ReadFile_Test();    /** Виклик методу тестування методу читання файлу */
    tester.Add_Test();         /** Виклик методу тестування методу додавання елементів */
    tester.Delete_Test();/** Виклик методу тестування методу видалення елементів */
```

```
	tester.SaveFile_Test();/** Виклик методу тестування методу виведення даних у файл */
	tester.Sort_Test();  /** Виклик методу тестування методу сортування */
	tester.Task_Test();  /** Виклик методу тестування методу виконання індивідуального
завдання */
	tester.~FuncTester();

	if (_CrtDumpMemoryLeaks()) /** Перевірка витоків пам'яті */
		cout << endl << "Есть утечка памяти." << endl;		/** Виведення
повідомленя якщо витоки є */
	else
		cout << endl << "Утечка памяти отсутствует." << endl; /** Виведення
повідомленя якщо витоків немає */

	return;			/** Завершення роботи програми */
}
```

Functor.cpp

```
/**
* @file Functor.cpp
* Файл реалізації методів класу Functor
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "Functor.h"		/** Підключення файлу Functor.h */

bool Functor::operator()(const unique_ptr<CProgram>& first, const unique_ptr<CProgram>&
second) const /** Реалізація перегруження оператору () */
{
	if (choise == 1)			/** Якщо треба сортувати по назві */
	{
		if (direction == true)	/** Вибір напряму сортування */
			return first->getName() > second->getName();
		else
			return first->getName() < second->getName();
	}
	else if (choise == 2)		/** Якщо треба сортувати по індексу */
	{
		if (direction == true)	/** Вибір напряму сортування */
			return first->getIndex() > second->getIndex();
		else
			return first->getIndex() < second->getIndex();
	}
	else if (choise == 3)		/** Якщо треба сортувати по кількості рядків коду */
	{
		if (direction == true)	/** Вибір напряму сортування */
			return first->getLines() > second->getLines();
		else
			return first->getLines() < second->getLines();
	}
	else if (choise == 4)		/** Якщо треба сортувати по розміру програми */
	{
		if (direction == true)	/** Вибір напряму сортування */
			return first->getSize() > second->getSize();
		else
			return first->getSize() < second->getSize();
	}
	else if (choise == 5)/** Якщо треба сортувати по часу виконання роботи програми */
	{
		if (direction == true)	/** Вибір напряму сортування */
			return first->getTime() > second->getTime();
		else
			return first->getTime() < second->getTime();
	}
```

```
	else if (choise == 6)			/** Якщо треба сортувати по Інтернету */
	{
		if (direction == true)		/** Вибір напряму сортування */
			return first->getInternet() > second->getInternet();
		else
			return first->getInternet() < second->getInternet();
	}
}

Functor::Functor(bool direction, int choise) :direction(direction), choise(choise) {}	/**
Реалізація конструктора з параметрами */
Functor::~Functor() {}					/** Реалізація деструктора */
```

FuncTester.cpp

```
/**
* @file FuncTester.cpp
* Файл реалізації методів класу-тестера
* @author Momot Roman
* @version 1.0
* @date 2020.05.26
*/

#include "FuncTester.h"						/** Підключення файлу
FuncTester.h */

void FuncTester::Add_Test()		/** Реалізація тестування метода додавання програми */
{
	if (value.AddProgram(2) == 10)			/** Виклик метода додавання програми */
		cout << "Тест функции добавления программы\t выполнен успешно." << endl;
	else
		cout << "Тест функции добавления программы\t не выполнен успешно." << endl;
}
void FuncTester::Delete_Test()	/** Реалізація тестування метода видалення програми */
{
	if (value.DeleteProgram(65) == 9)		/** Виклик метода видалення програми */
		cout << "Тест функции удаления программы\t\t выполнен успешно." << endl;
	else
		cout << "Тест функции удаления программы\t\t не выполнен успешно." << endl;
}
void FuncTester::ReadFile_Test()  /** Реалізація тестування метода читання даних з файлу */
{
	if(value.ReadFile("data.txt") == 4)		/** Виклик метода читання даних з файлу */
		cout << "Тест функции чтения файла\t\t выполнен успешно." << endl;
	else
		cout << "Тест функции чтения из файла\t\t не выполнен успешно." << endl;
}
void FuncTester::SaveFile_Test()  /** Реалізація тестування метода виведення даних у файл */
{
	if (value.SaveFile("Test.txt") == 9)		/** Виклик метода виведення даних у файл */
		cout << "Тест функции сохранения в файл\t\t выполнен успешно." << endl;
	else
		cout << "Тест функции сохранения в файл\t\t не выполнен успешно." << endl;
}
void FuncTester::Sort_Test()			/** Реалізація тестування метода сортування */
{
	Functor funct(0, 2);				/** Створення елемента класу Functor */
	int beforeSort = value.GetIndex(0);
	value.Sort(funct);					/** Виклик метода сортування */
	int afterSort = value.GetIndex(0);

	if (beforeSort != afterSort && afterSort < value.GetIndex(2))
		cout << "Тест функции сортировки списка\t\t выполнен успешно." << endl <<
endl;
	else
```

```
		cout << "Тест функции сортировки списка\t\t не выполнен успешно." << endl <<
endl;
}
void FuncTester::Task_Test()/** Реалізація тестування метода реалізації індивідуального
завдання */
{
	if (value.Task(100) == 7)  /** Виклик метода реалізації індивідуального завдання */
		cout << endl << "Тест функции индивидуального задания\t выполнен успешно." <<
endl;
	else
		cout << endl << "Тест функции индивидуального задания\t не выполнен успешно."
<< endl;
}

FuncTester::FuncTester() {}	/** Реалізація конструктора за замовчуванням */
FuncTester::~FuncTester() {}	/** Реалізація деструктора */
```

Додаток Б

Результати роботи програми

```
1) Вывести список на экран
2) Удаление элемента
3) Добавление элемента
4) Индивидуальное задание
5) Работа с файлами
6) Сортировка
7) Завершение работы
=========================
Ваш выбор: 1


  Время     Размер      Строки    Интернет   Индекс        Название         Тип/Цена
1. 200       200         200      false       65         Basic
2. 8800      555         35       true        35634      BestMalware         Exploit
3. 423       20          654      false       53453      Calculator
4. 345       789         423      false       67456      MoneyStealer        99999
5. 200       200         200      false       65         Basic               Trojan


1) Вывести список на экран
2) Удаление элемента
3) Добавление элемента
4) Индивидуальное задание
5) Работа с файлами
6) Сортировка
7) Завершение работы
=========================
Ваш выбор: 7

Завершение работы.

Тестирование функций с помощью класса-тестировщика
Тест функции чтения файла                выполнен успешно.
Тест функции добавления программы        выполнен успешно.
Удаление выполнено.
Тест функции удаления программы          выполнен успешно.
Тест функции сохранения в файл           выполнен успешно.
Тест функции сортировки списка           выполнен успешно.

1. 666       666         666      true        666        Photoshop           8800555
2. 125       234         1236     true        5322       Minecraft
3. 4523      756         4213     false       8579       TheBattleship       Adware
4. 222       333         444      false       14212      Notepad             Exploit
5. 423       20          654      false       53453      Calculator
6. 345       789         423      false       67456      MoneyStealer        99999

Тест функции индивидуального задания     выполнен успешно.

Утечка памяти отсутствует.
```

Рис. 10. Результати роботи програми