МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка
1КІТ.102.8А. 18037-01 81 01-1 –АЗ

Розробник
Виконав:

студент групи 1КІТ-102.8А
_____/ Малюга А. В./

Перевірив:
_____/Старший викладач Молчанов Г.І./

Харків 2019

РОЗРАХУНКОВОГО ЗАВДАННЯ З ДИСЦИПЛІНИ
«ПРОГРАМУВАННЯ»

Тема роботи. Розробка інформаційно-довідкової системи.

Мета роботи. Закріпити отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

1 ВИМОГИ

1.1 Розробник

- Малюга Андрій Володимирович;
- Студент групи КІТ 102.8а

1.2 Загальне завдання

Завдання до роботи:

Кожний студент отримує індивідуальне завдання. Варіант завдання обирається за номером прізвища студента у журналі групи. При виконанні завдання з розробки інформаційно-довідкової системи необхідно виконати наступне:

1) з табл. 1, відповідно до варіанта завдання, обрати прикладну галузь;

2) дослідити літературу стосовно прикладної галузі. За результатами аналізу літератури оформити перший, аналітичний розділ пояснювальної записки обсягом 2–3 сторінки;

3) для прикладної галузі розробити розгалужену ієрархію класів, яка складається з не менш ніж трьох класів, один з яких є «батьком» для інших (класів-спадкоємців). Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;

4) розробити клас-контролер, що буде включати 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) пошук елементів за вказаним критерієм (див. «Завдання для обходу колекції» в табл. 1);

5) розробити клас, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;

6) оформити схеми алгоритмів функцій класів контролера та діалогового меню;

7) оформити документацію: пояснювальну записку (див. розділ 2 даних методичних вказівок).

Увага. Текст програми та результати роботи програми мають бути подані в додатках.

Вимоги:

− усі класи повинні мати конструктори та деструктори;

− якщо функція не змінює поля класу, вона має бути декларована як константна;

− рядки повинні бути типу string;

− при перевантаженні функції треба використовувати ключове слово override;

− програмний код усіх класів має бути 100 % doxygen документований;

− у звіті текст програми слід оформляти стилем Courier new 8 пт, інтервал – одиничний; довжина рядка не повинна перевищувати 80 символів.

Додаткові вимоги на оцінку «добре»:

− виконання основного завдання та додаткових наступних вимог:

− додати обробку помилок; при цьому функція, що генерує виключення, при її декларуванні повинна мати ключове слово throw;

− виконати перевірку вхідних даних за допомогою регулярних виразів.

Додаткові вимоги на оцінку «відмінно»:

− виконати завдання відповідно до вимог на оцінку «добре» та додаткові наступні вимоги:

− критерій для пошуку та сортування задавати у вигляді функтора;

− розробити клас-тестер, основною метою якого буде перевірка коректності роботи класу-контролера.

2 ОПИС ПРОГРАМИ

2.1 Функціональне призначення

За допомогою цієї програми можна створити масив об’єктів, додавати та видаляти об’єкти, виводити вміст масиву на екран та вивід об’єкта по індексу, пошук по імені, читання з файлу даних про об'єкти та запис значень полів об'єктів масиву у файл. Також у цій програмі реалізоване зручне меню спілкування з користувачем.

2.2 2 Важливі фрагменти програми

На рисунку №1 зображено діаграму класів

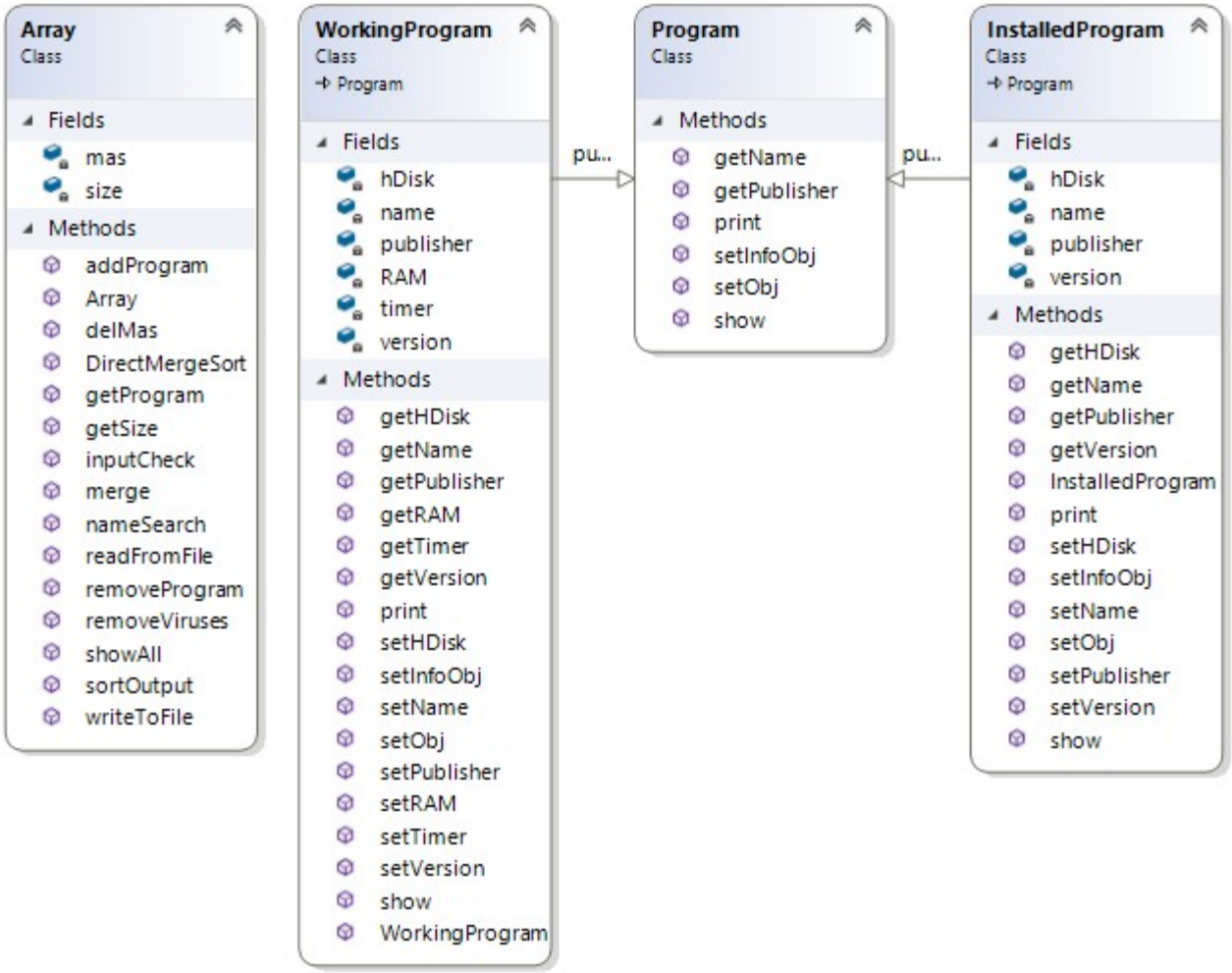

Рисунок №1 – діаграма класів

Методи класу Program:

- virtual stringstream print() - віртуальний метод, метою якого є створення рядка з інформацією про об'єкт
- virtual string getName(string name) - віртуальний метод, метою якого є повернення значення поля name

- virtual void setObj(string &info) - віртуальний метод, метою якого є зміна значень полів об'єкта
- virtual void setInfoObj(string &info) – віртуальний метод, метою якого є зчитування інформації про об'єкт з клавіатури
- virtual void show() - віртуальний метод, який виводить значення полів на екран

Методи класу InstalledProgram:

- InstalledProgram() - Конструктор за замовчуванням
- stringstream print() - метод, який створює рядок з інформацією про об'єкт і повертає її
- void setName(string name) - заповнення поля workingProgram::name (інші методи set роблять теж саме але з іншими полями)
- string getName() - читання значення поля name(інші методи get роблять теж саме але з іншими полями)
- void setObj(string &info) - метод, метою якого є зміна значень полів об'єкта
- void setInfoObj(string &info) – метод, метою якого є зчитування інформації про об'єкт з клавіатури
- void show() - віртуальний метод, який виводить значення полів на екран

Методи класу WorkingProgram:

- WorkingProgram() - Конструктор за замовчуванням
- stringstream print() - метод, який створює рядок з інформацією про об'єкт і повертає її
- void setName(string name) - заповнення поля workingProgram::name (інші методи set роблять теж саме але з іншими полями)
- string getName() - читання значення поля name(інші методи get роблять теж саме але з іншими полями)
- void setObj(string &info) - метод, метою якого є зміна значень полів об'єкта
- void setInfoObj(string &info) – метод, метою якого є зчитування інформації про об'єкт з клавіатури
- void show() - віртуальний метод, який виводить значення полів на екран

Методи класу Array:

- Array() - конструктор за замовчуванням
- void writeToFile() - метод для записузначеньполів в файл.
- void readFromFile(int &sizeMas, WorkingProgram &newObj) - метод для читання іфнормаціі про об'єкти з файлу
- void addProgram(WorkingProgram &newObj, int ind) - метод створений, щоб додати об'єкт в масив
- void showAll() - метод створений для виведення всіх елементів масиву на екран

- void removeProgram(int ind) - метод створений для видалення елемента з масиву
- void getProgram(int ind) - метод створений для виведення одного елемента за індексом з масиву
- void nameSearch(string n) - метод створений для пошуку об'єкта масиву по імені
- void removeViruses() - метод створений для видалення підозрілих програм з масиву
- void delMas() - метод створений для очищення виділеної пам'яті для масиву об'єктів
- int getSize() - метод створений для читання значення поля size
- void sortOutput() - метод який виводить на екран програми у яких назви складаються з 2 слів і більше
- bool inputCheck(string str) - метод який за допомогою регулярних виразів перевіряє імена програм і імена творців програм, що рядки повинні починатися з великої літери і не повинно бути двох і більше прогалин стоять поруч
- void merge(int b, int m, int e) - метод злиття елементів масиву
- void DirectMergeSort(int b, int e) – метод сортування злиттям

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма може буди використана для створення масиву об’єктів. Програма має методи додавання, видалення об’єктів, пошук об’єктів по імені та вивід по індексу з масиву, читання з файлу даних про об'єкти та запис значень полів об'єктів масиву у файл. Меню робить роботу з цією програмою зручною.

Меню спілкування з користувачем зображено на рисунку №2

```
Quantity of objects in array: 0
What function do you want to cause  the list?
(0)exit from program
(1)output array on display
(2)name search
(3)to delete objest from array
(4)to add new object to array
(5)index output on display
(6)to delete suspicious programs from array
(7)to read inforamtion of objects from file
(8)to write array to file
(9)output objects from array with more than one word in name
(10)to sort array
```

Рисунок №2 – меню спілкування з користувачем

Результат виводу об’єктів масиву на екран зображено на рисунку №3

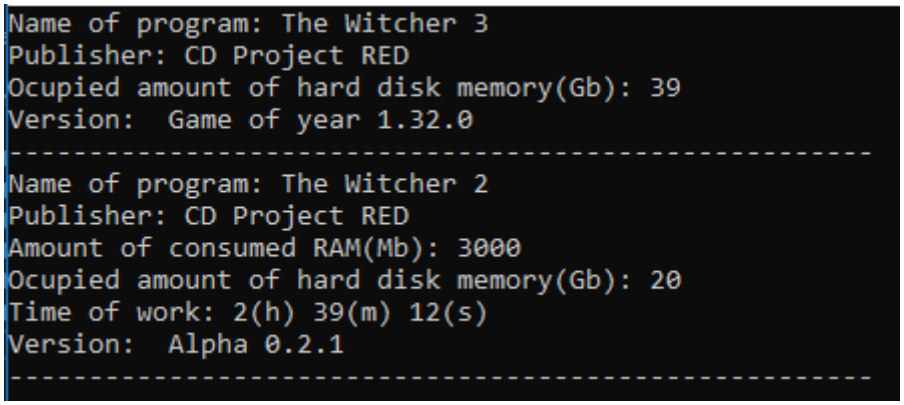

Рисунок №3 - результат виводу об’єктів масиву на екран

ВИСНОВОК

В ході виконання поставленої задачі були закріплені отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

Код програми:

File “Program.h”

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

class Program {
public:
	virtual string print() = 0;
	virtual void setObj(string &info) = 0;
	virtual void setInfoObj(string &info) = 0;
	virtual const void show() = 0;
	virtual string getName() = 0;
	virtual string getPublisher() = 0;
	virtual float getHDisk() = 0;

	};
```

File “InstalledProgram.h”

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


class InstalledProgram : public Program{
private:
	string name;
	string publisher;
	float hDisk;
	Version version;
public:
	InstalledProgram();
	string print();
	void setInfoObj(string &info);
	void setObj(string &info);
	const void show();

	void setName(string name);
	void setPublisher(string publisher);
	void setHDisk(float hDisk);
	void setVersion(Version version);

	string getName();
	string getPublisher();
	float getHDisk();
	Version getVersion();

	InstalledProgram& operator= (const InstalledProgram &obj);
	friend bool operator< (const InstalledProgram &obj1, const InstalledProgram &obj2);
	friend istream& operator>> (istream &in, InstalledProgram &obj);
	friend ostream& operator<< (ostream &in, const InstalledProgram &obj);

	};
```

File “InstalledProgram.cpp”

```cpp
#include "Main.h"

//setter
void InstalledProgram::setName(string name) {
	this->name = name;
}

void InstalledProgram::setPublisher(string publisher) {
	this->publisher = publisher;
}


void InstalledProgram::setHDisk(float memoryGb) {
	this->hDisk = memoryGb;
}

void InstalledProgram::setVersion(Version version) {
	this->version = version;
}

//getter
string InstalledProgram::getName() {
	return name;
}

string InstalledProgram::getPublisher() {
	return publisher;
}

float InstalledProgram::getHDisk() {
	return hDisk;
}

Version InstalledProgram::getVersion() {
	return version;
}



InstalledProgram::InstalledProgram() : name(""), publisher(""), hDisk(0) {
	version.name = "";
	version.arr[0] = 0;
	version.arr[1] = 0;
	version.arr[2] = 0;
}

string InstalledProgram::print() {
	stringstream lineObj;
	lineObj << name << "|";
	lineObj << publisher << "|";
	lineObj << hDisk << " ";
	lineObj << version.name << "|";
	lineObj << version.arr[0] << " " << version.arr[1] << " " << version.arr[2];
	string infoObj;
	getline(lineObj, infoObj);
	return infoObj;
}

void InstalledProgram::setObj(string &info) {
	stringstream timeLine;

	timeLine << info;
	getline(timeLine, name, '|');
	getline(timeLine, publisher, '|');
	timeLine >> hDisk;
```

```
	getline(timeLine, version.name, '|');
	timeLine >> version.arr[0];
	timeLine >> version.arr[1];
	timeLine >> version.arr[2];
}

void InstalledProgram::setInfoObj(string &info) {
	string name;
	string publisher;
	float RAM, hDisk;
	Time *timer = new Time;
	stringstream infoObj;
	Version version;
	Array ops;
	while (true) {
		cout << "Enter name of program:" << endl;
		getline(cin, name);
		if (ops.inputCheck(name) == true) {
			break;
		}
	}
	while (true) {
		cout << "Enter name of publisher(if you don't know, enter 'Unknown'):" << endl;
		getline(cin, publisher);
		if (ops.inputCheck(publisher) == true) {
			break;
		}
	}

	infoObj << name << "|";
	infoObj << publisher << "|";

	cout << "Enter ocupied amount of hard disk memory(Gg):" << endl;
	cin >> hDisk;
	infoObj << hDisk << " ";

	cin.ignore();
	cout << "Enter version:" << endl;
	cout << "Name of version - ";
	getline(cin, version.name);
	infoObj << version.name << '|';

	cout << "First number - ";
	cin >> version.arr[0];
	infoObj << version.arr[0] << " ";

	cout << "Second number - ";
	cin >> version.arr[1];
	infoObj << version.arr[1] << " ";

	cout << "Third number - ";
	cin >> version.arr[2];
	infoObj << version.arr[2];

	getline(infoObj, info);
	delete timer;
}

const void InstalledProgram::show() {
	cout << "Name of program: " << name << endl;
	cout << "Publisher: " << publisher << endl;
	cout << "Ocupied amount of hard disk memory(Gb): " << hDisk << endl;
	cout << "Version: " << version.name << ' ' << version.arr[0] << '.' << version.arr[1] << '.' << version.arr[2] << endl;
	cout << "-----------------------------------------------------" << endl;
}
```

```
istream& operator >> (istream &in, InstalledProgram &obj) {
	in.ignore();
	cout << "Enter name of program:" << endl;
	getline(in, obj.name);

	cout << "Enter name of publisher(if you don't know, enter 'Unknown'):" << endl;
	getline(in, obj.publisher);

	cout << "Enter ocupied amount of hard disk memory(Gg):" << endl;
	in >> obj.hDisk;

	in.ignore();
	cout << "Enter version:" << endl;
	cout << "Name of version - ";
	getline(in, obj.version.name);
	cout << "First number - ";
	in >> obj.version.arr[0];
	cout << "Second number - ";
	in >> obj.version.arr[1];
	cout << "Third number - ";
	in >> obj.version.arr[2];
	return in;
}

ostream& operator<< (ostream &out, const InstalledProgram &obj) {
	out << "Name of program: " << obj.name << endl;
	out << "Publisher: " << obj.publisher << endl;
	out << "Ocupied amount of hard disk memory(Gb): " << obj.hDisk << endl;
	out << "Version: " << obj.version.name << ' ' << obj.version.arr[0] << '.' << obj.version.arr[1] << '.' << obj.version.arr[2]
<< endl;
	return out;
}

bool operator< (const InstalledProgram &obj1, const InstalledProgram &obj2) {
	return obj1.hDisk < obj2.hDisk;
}

InstalledProgram& InstalledProgram::operator= (const InstalledProgram &obj) {
	this->name = obj.name;
	this->publisher = obj.publisher;
	this->hDisk = obj.hDisk;
	this->version.name = obj.version.name;
	this->version.arr[0] = obj.version.arr[0];
	this->version.arr[1] = obj.version.arr[1];
	this->version.arr[2] = obj.version.arr[2];
	return *this;

	}
```

File “WorkingProgram.h

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



class WorkingProgram : public Program {
private:
	string name;
	string publisher;
	float RAM;
```

```
        float hDisk;
        Time timer;
        Version version;
public:
        WorkingProgram();

        string print();
        void setInfoObj(string &info);
        void setObj(string &info);
        const void show();

        void setName(string name);
        void setPublisher(string publisher);
        void setRAM(float RAM);
        void setHDisk(float hDisk);
        void setTimer(Time timer);
        void setVersion(Version version);

        string getName();
        string getPublisher();
        float getRAM();
        float getHDisk();
        Time getTimer();
        Version getVersion();

        void operator = (WorkingProgram &obj);
        bool operator == (WorkingProgram &obj);
        friend istream& operator>> (istream &in, WorkingProgram &obj);
        friend ostream& operator<< (ostream &in, WorkingProgram &obj);
};
```

File “WorkingProgram.cpp”

```
#include "Main.h"

//setter
void WorkingProgram::setName(string name) {
        this->name = name;
}

void WorkingProgram::setPublisher(string publisher) {
        this->publisher = publisher;
}

void WorkingProgram::setRAM(float opMemoryMb) {
        this->RAM = opMemoryMb;
}

void WorkingProgram::setHDisk(float memoryGb) {
        this->hDisk = memoryGb;
}

void WorkingProgram::setTimer(Time timeWorkMin) {
        this->timer = timeWorkMin;
}

void WorkingProgram::setVersion(Version version) {
        this->version = version;
}

//getter
```

```
string WorkingProgram::getName() {
	return name;
}

string WorkingProgram::getPublisher() {
	return publisher;
}

float WorkingProgram::getRAM() {
	return RAM;
}

float WorkingProgram::getHDisk() {
	return hDisk;
}

Time WorkingProgram::getTimer() {
	return timer;
}

Version WorkingProgram::getVersion() {
	return version;
}



WorkingProgram::WorkingProgram() : name(""), publisher(""), RAM(0), hDisk(0) {
	timer.hours = 0;
	timer.minutes = 0;
	timer.seconds = 0;
	version.name = "";
	version.arr[0] = 0;
	version.arr[1] = 0;
	version.arr[2] = 0;
}

string WorkingProgram::print() {
	stringstream lineObj;
	lineObj << name << "|";
	lineObj << publisher << "|";
	lineObj << RAM << " ";
	lineObj << hDisk << " ";
	lineObj << timer.hours << " ";
	lineObj << timer.minutes << " ";
	lineObj << timer.seconds;
	lineObj << version.name << "|";
	lineObj << version.arr[0] << " " << version.arr[1] << " " << version.arr[2];
	string infoObj;
	getline(lineObj, infoObj);
	return infoObj;
}

void WorkingProgram::setObj(string &info) {
	stringstream timeLine;

	timeLine << info;
	getline(timeLine, name, '|');
	getline(timeLine, publisher, '|');
	timeLine >> RAM;
	timeLine >> hDisk;
	timeLine >> timer.hours;
	timeLine >> timer.minutes;
	timeLine >> timer.seconds;
	getline(timeLine, version.name, '|');
	timeLine >> version.arr[0];
	timeLine >> version.arr[1];
```

```
	timeLine >> version.arr[2];
}

void WorkingProgram::setInfoObj(string &info) {
	string name;
	string publisher;
	float RAM, hDisk;
	Time timer;
	stringstream infoObj;
	Version version;
	Array ops;
	while (true) {
		cout << "Enter name of program:" << endl;
		getline(cin, name);
		if (ops.inputCheck(name) == true) {
			break;
		}
	}
	while (true) {
		cout << "Enter name of publisher(if you don't know, enter 'Unknown'):" << endl;
		getline(cin, publisher);
		if (ops.inputCheck(publisher) == true) {
			break;
		}
	}

	infoObj << name << "|";
	infoObj << publisher << "|";
	cout << "Enter amount of consumed RAM(Mb):" << endl;
	cin >> RAM;
	infoObj << RAM << " ";

	cout << "Enter ocupied amount of hard disk memory(Gg):" << endl;
	cin >> hDisk;
	infoObj << hDisk << " ";

	cout << "Enter time of work:" << endl;

	cout << "Hours - ";
	cin >> timer.hours;
	infoObj << timer.hours << " ";

	while (true) {
		cout << "(0-59)Minutes - ";
		cin >> timer.minutes;
		infoObj << timer.minutes << " ";
		if (timer.minutes < 0 || timer.minutes >= 60) {
			cout << "You must enter from 0 to 59 minutes, try again" << endl;
		}
		else {
			break;
		}
	}
	while (true) {
		cout << "(0-59)Seconds - ";
		cin >> timer.seconds;
		infoObj << timer.seconds << " ";
		if (timer.seconds < 0 || timer.seconds >= 60) {
			cout << "You must enter from 0 to 59 minutes, try again" << endl;
		}
		else {
			break;
		}
	}

	cin.ignore();
```

```
	cout << "Enter version:" << endl;
	cout << "Name of version - ";
	getline(cin, version.name);
	infoObj << version.name << '|';

	cout << "First number - ";
	cin >> version.arr[0];
	infoObj << version.arr[0] << " ";

	cout << "Second number - ";
	cin >> version.arr[1];
	infoObj << version.arr[1] << " ";

	cout << "Third number - ";
	cin >> version.arr[2];
	infoObj << version.arr[2];

	getline(infoObj, info);
}

const void WorkingProgram::show() {
	cout << "Name of program: " << name << endl;
	cout << "Publisher: " << publisher << endl;
	cout << "Amount of consumed RAM(Mb): " << RAM << endl;
	cout << "Ocupied amount of hard disk memory(Gb): " << hDisk << endl;
	cout << "Time of work: " << timer.hours << "(h) " << timer.minutes << "(m) " << timer.seconds << "(s)" << endl;
	cout << "Version: " << version.name << ' ' << version.arr[0] << '.' << version.arr[1] << '.' << version.arr[2] << endl;
	cout << "-----------------------------------------------------" << endl;
}




void WorkingProgram::operator = (WorkingProgram& obj) {
	this->name = obj.name;
	this->publisher = obj.publisher;
	this->RAM = obj.RAM;
	this->hDisk = obj.hDisk;
	this->timer = obj.timer;
	this->version = obj.version;
}

bool WorkingProgram::operator == (WorkingProgram& obj) {
	if (this->name != obj.name) {
		return false;
	}
	if (this->publisher != obj.publisher) {
		return false;
	}
	if (this->RAM != obj.RAM) {
		return false;
	}
	if (this->hDisk != obj.hDisk) {
		return false;
	}
	if (this->timer.hours != obj.timer.hours) {
		return false;
	}
	if (this->timer.minutes != obj.timer.minutes) {
		return false;
	}
	if (this->timer.seconds != obj.timer.seconds) {
		return false;
	}
	if (this->version.name != obj.version.name) {
		return false;
```

```
	}
	for (int i = 0; i < 3; i++) {
		if (this->version.arr[i] != obj.version.arr[i]) {
			return false;
		}
	}
	return true;
}

istream& operator >> (istream &in, WorkingProgram &obj) {
	in.ignore();
	cout << "Enter name of program:" << endl;
	getline(in, obj.name);

	cout << "Enter name of publisher(if you don't know, enter 'Unknown'):" << endl;
	getline(in, obj.publisher);

	cout << "Enter amount of consumed RAM(Mb):" << endl;
	in >> obj.RAM;

	cout << "Enter ocupied amount of hard disk memory(Gg):" << endl;
	in >> obj.hDisk;

	cout << "Enter time of work:" << endl;
	cout << "Hours - ";
	in >> obj.timer.hours;

	while (true) {
		cout << "(0-59)Minutes - ";
		in >> obj.timer.minutes;
		if (obj.timer.minutes < 0 || obj.timer.minutes > 60) {
			cout << "You must enter from 0 to 59 minutes, try again" << endl;
		}
		else {
			break;
		}
	}
	while (true) {
		cout << "(0-59)Seconds - ";
		in >> obj.timer.seconds;
		if (obj.timer.seconds < 0 || obj.timer.seconds > 60) {
			cout << "You must enter from 0 to 59 minutes, try again" << endl;
		}
		else {
			break;
		}
	}
	in.ignore();
	cout << "Enter version:" << endl;
	cout << "Name of version - ";
	getline(in, obj.version.name);
	cout << "First number - ";
	in >> obj.version.arr[0];
	cout << "Second number - ";
	in >> obj.version.arr[1];
	cout << "Third number - ";
	in >> obj.version.arr[2];
	return in;
}

ostream& operator<< (ostream &out, WorkingProgram &obj) {
	out << "Name of program: " << obj.name << endl;
	out << "Publisher: " << obj.publisher << endl;
	out << "Amount of consumed RAM(Mb): " << obj.RAM << endl;
	out << "Ocupied amount of hard disk memory(Gb): " << obj.hDisk << endl;
```

```
	out << "Time of work: " << obj.timer.hours << "(h) " << obj.timer.minutes << "(m) " << obj.timer.seconds << "(s)" << endl;
	out << "Version: " << obj.version.name << ' ' << obj.version.arr[0] << '.' << obj.version.arr[1] << '.' << obj.version.arr[2] << endl;
	out << endl;
	return out;
}
```

File “Menu.h”

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

class Menu {
private:
	Array ops;
public:
	void menu();
};
```

File “Menu.cpp”

```
#include "Main.h"

void Menu::menu() {
	int num;
	int ind;
	int sizeMas = 0;
	string n;
	string infoObj;
	Program *pObj;
	string endFile;
	Array ops;
	ifstream object;
	string endOfFile;
	float memoryGb;
	while (true) {
		cout << "Quantity of objects in array: " << sizeMas << endl;
		cout << "What function do you want to cause  the list?" << endl;
		cout << "(0)exit from program" << endl;
		cout << "(1)output array on display" << endl;
		cout << "(2)name search" << endl;
		cout << "(3)to delete objest from array" << endl;
		cout << "(4)to add new object to array" << endl;
		cout << "(5)index output on display" << endl;
		cout << "(6)to delete suspicious programs from array" << endl;
		cout << "(7)to read inforamtion of objects from file" << endl;
		cout << "(8)to write array to file" << endl;
		cout << "(9)output objects from array with more than one word in name" << endl;
		cout << "(10)to sort array" << endl;
		cout << "(11)show programs, that take up more memory of a given size" << endl;
		cin >> num;
		system("cls");

		switch (num) {
		case 1:
```

```
			ops.showAll();
			cout << endl;
			system("pause");
			system("cls");
			break;
case 2:
			cin.ignore();
			cout << "('0' exit from search)Enter name of program from array: ";
			getline(cin, n);

			if (n == "0") {
				system("cls");
				break;
			}
			else {
				ops.nameSearch(n);
				cout << endl;
				system("pause");
				system("cls");
				break;
			}
			break;
case 3:
			cout << "Enter number of object for delete it: ";
			cin >> ind;
			sizeMas = ops.getSize();
			if (ind <= sizeMas + 1 && ind >= 1) {
				ops.removeProgram(ind);
				sizeMas = ops.getSize();
				system("cls");
				break;
			}
			else {
				cout << "There is no object in the array with this index" << endl;

				system("pause");
				system("cls");
				break;
			}
case 4:
			cout << "Enter type of program ((1)working programm, (other numbers)installed program) - ";
			cin >> ind;
			if (ind == 1) {
				cout << "Enter number of object for add it to array: ";
				cin >> ind;
				sizeMas = ops.getSize();

				cin.ignore();
				if (ind <= sizeMas + 1 && ind >= 1) {
					Program *newObj1 = new WorkingProgram;
					newObj1->setInfoObj(infoObj);
					newObj1->setObj(infoObj);
					pObj = newObj1;
					ops.addProgram(pObj, ind);
					sizeMas = ops.getSize();
					system("cls");
					break;
				}
				else {
					cout << "There is no object in the array with this index" << endl;

					system("pause");
					system("cls");
					break;
				}
			}
```

```
		else {
			cout << "Enter number of object for add it to array: ";
			cin >> ind;
			sizeMas = ops.getSize();

			cin.ignore();
			if (ind <= sizeMas + 1 && ind >= 1) {
				Program *newObj2 = new InstalledProgram;
				newObj2->setInfoObj(infoObj);
				newObj2->setObj(infoObj);
				pObj = newObj2;
				ops.addProgram(pObj, ind);

				sizeMas = ops.getSize();
				system("cls");
				break;
			}
			else {
				cout << "There is no object in the array with this index" << endl;

				system("pause");
				system("cls");
				break;
			}
		}
		infoObj = "";
		break;
case 5:
		cout << "Enter number of object from array: ";
		cin >> ind;
		sizeMas = ops.getSize();
		if (ind <= sizeMas && ind >= 1) {
			ops.getProgram(ind);

			system("pause");
			system("cls");
			break;
		}
		else {
			cout << "There is no object in the array with this index" << endl;

			system("pause");
			system("cls");
			break;
		}
case 6:
		ops.removeViruses();
		sizeMas = ops.getSize();
		system("cls");
		break;
case 7:
		ops.readFromFile(sizeMas, ops);
		sizeMas = ops.getSize();
		system("cls");
		break;
case 8:
		ops.writeToFile();
		system("cls");
		break;
case 9:
		ops.sortOutput();
		system("pause");
		system("cls");
		break;
case 10:
		ops.DirectMergeSort(0, sizeMas - 1);
```

```
                        system("cls");
                        break;
                case 11:
                        cout << "Enter size of memory: ";
                        cin >> memoryGb;
                        ops.findProgram(memoryGb);

                        system("pause");
                        system("cls");
                        break;
                case 0:
                        ops.delMas();
                        return;
                }
        }
}
```

File “Array.h”

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

class Array {
private:
        int size;
        Program **mas;
public:
        Array();
        void writeToFile();
        void readFromFile(int &sizeMas, Array &ops);
        void addProgram(Program *newObj, int ind);
        const void showAll();
        void removeProgram(int ind);
        const void getProgram(int ind);
        void nameSearch(string n);
        void removeViruses();
        void delMas();
        int getSize();
        void merge(int b, int m, int e);
        void DirectMergeSort(int b, int e);
        bool inputCheck(string str);
        const void sortOutput();
        const void findProgram(float memoryGB);
};
```

File “Array.cpp”

```
#include "Main.h"

void Array::addProgram(Program *newObj, int ind) {
        ind = ind - 1;

        Program **timeMas = new Program*[size + 1];

        timeMas[ind] = newObj;

        int i = 0;
        while (i < ind) {
                timeMas[i] = mas[i];
```

```
                i++;
        }
        while (i < size) {
                timeMas[i + 1] = mas[i];
                i++;
        }

        if (size != 0) {
                delete[] mas;
        }

        mas = timeMas;
        size++;
}

void Array::removeProgram(int ind) {
        if (size == 0) {
                cout << "Array is empty" << endl;
                system("pause");
                return;
        }
        ind = ind - 1;
        Program **timeMas = new Program*[size - 1];

        int i = 0;
        while (i < ind) {
                timeMas[i] = mas[i];
                i++;
        }

        if (size > 1) {
                while (i < size - 1) {
                        timeMas[i] = mas[i + 1];
                        i++;
                }
        }


        if (size != 0) {
                delete mas[ind];
                delete[] mas;
        }

        mas = timeMas;
        size--;
}

const void Array::showAll() {
        if (size == 0) {
                cout << "Array is empty" << endl;
                return;
        }
        for (int i = 0; i < size; i++) {
                mas[i]->show();
        }
}

const void Array::getProgram(int ind) {
        if (size == 0) {
                cout << "Array is empty" << endl;
                system("pause");
                return;
        }
        mas[ind - 1]->show();
}
```

```
void Array::nameSearch(string n) {
	if (size == 0) {
		cout << "Array is empty" << endl;
		system("pause");
		return;
	}

	string na;

	for (int i = 0; i < size; i++) {
		na = mas[i]->getName();
		if (na == n) {
			mas[i]->show();
		}
	}
}

int Array::getSize() {
	return size;
}

void Array::delMas() {
	for (int i = 0; i < size; i++) {
		delete mas[i];
	}
	delete[] mas;
}

Array::Array() :size(0), mas(NULL) {
}


void Array::removeViruses() {
	if (size == 0) {
		cout << "Array is empty" << endl;
		system("pause");
		system("cls");
		return;
	}

	string p = "Unknown";
	for (int i = 0; i < size; i++) {
		if (p == mas[i]->getPublisher()) {
			removeProgram(i + 1);
		}
	}
}

void Array::readFromFile(int &sizeMas, Array &ops) {
	string publisher, name;
	float RAM, hDisk;
	Time timer;
	string type;
	Version version;
	stringstream infoObj;
	string info;
	ifstream objects("maliuha07.txt");
	if (!objects.is_open()) {
		cout << "File was not opened" << endl;
		system("pause");
		return;
	}

	for (int i = 0; i < sizeMas + 1; i++) {
		getline(objects, type);
		if (type == "") {
```

```
		getline(objects, type);
	}
	if (type == "WorkingProgram") {
		getline(objects, name);
		if (name == "") {
			getline(objects, name);
		}
		infoObj << name << "|";

		getline(objects, publisher);
		infoObj << publisher << "|";

		objects >> RAM;
		infoObj << RAM << " ";

		objects >> hDisk;
		infoObj << hDisk << " ";

		objects >> timer.hours;
		infoObj << timer.hours << " ";

		objects >> timer.minutes;
		infoObj << timer.minutes << " ";

		objects >> timer.seconds;
		infoObj << timer.seconds << " ";

		getline(objects, version.name);
		if (version.name == "") {
			getline(objects, version.name);
		}
		infoObj << version.name << "|";

		for (int i = 0; i < 3; i++) {
			objects >> version.arr[i];
			infoObj << version.arr[i] << " ";
		}

		getline(infoObj, info);

		Program *obj = new WorkingProgram;
		obj->setObj(info);
		ops.addProgram(obj, sizeMas + 1);

		sizeMas++;
		type = "";
		infoObj.str("");
		infoObj.clear();
	}
	if (type == "InstalledProgram") {
		getline(objects, name);
		if (name == "") {
			getline(objects, name);
		}
		infoObj << name << "|";


		getline(objects, publisher);
		infoObj << publisher << "|";

		objects >> hDisk;
		infoObj << hDisk << " ";

		getline(objects, version.name);
		if (version.name == "") {
			getline(objects, version.name);
```

```
                        }
                        infoObj << version.name << "|";

                        for (int i = 0; i < 3; i++) {
                                objects >> version.arr[i];
                                infoObj << version.arr[i] << " ";
                        }

                        getline(infoObj, info);

                        Program *obj2 = new InstalledProgram;
                        obj2->setObj(info);
                        ops.addProgram(obj2, sizeMas + 1);

                        sizeMas++;
                        type = "";
                        info = "";
                        infoObj.str("");
                        infoObj.clear();
                }
        }
        objects.close();
}

void Array::writeToFile() {
        std::streambuf *coutbuf = std::cout.rdbuf();
        std::ofstream out("maliuha07w.txt");
        std::cout.rdbuf(out.rdbuf());
        for (int i = 0; i < size; i++) {
                mas[i]->show();

        }
        std::cout.rdbuf(coutbuf);
}

bool Array::inputCheck(string str) {
        regex regular("^[A-Z][A-Za-z ,.1234567890]*");
        regex regular2(" {2,}");
        if (!regex_match(str, regular) || regex_search(str, regular2)) {
                cout << "This line does not meet the requirements: " << endl;
                cout << str << endl;
                return false;
        }
        else {
                return true;
        }
}

const void Array::sortOutput() {
        regex regular("[A-Za-z]*");
        for (int i = 0; i < size; i++) {
                if (!regex_match(mas[i]->getName(), regular)) {
                        mas[i]->show();
                }
        }
}

void Array::merge(int b, int m, int e) {
        int pos1 = b;
        int pos2 = m + 1;
        int pos3 = 0;
        Program **temp = new Program*[e - b + 1];
        string name1, name2;

        while (pos1 <= m && pos2 <= e)
        {
```

```
                name1 = mas[pos1]->getName();
                name2 = mas[pos2]->getName();
                if (name1[0] < name2[0]) {
                        temp[pos3++] = mas[pos1++];
                }
                else {
                        temp[pos3++] = mas[pos2++];
                }
        }

        while (pos2 <= e) {
                temp[pos3++] = mas[pos2++];
        }
        while (pos1 <= m) {
                temp[pos3++] = mas[pos1++];
        }
        for (pos3 = 0; pos3 < e - b + 1; pos3++) {
                mas[b + pos3] = temp[pos3];
        }
        delete[] temp;
}


void Array::DirectMergeSort(int b, int e)
{
        long m;
        if (b < e)
        {
                m = (b + e) / 2;
                DirectMergeSort(b, m);
                DirectMergeSort(m + 1, e);
                merge(b, m, e);
        }
}


const void Array::findProgram(float memoryGB) {
        if (size == 0) {
                cout << "Array is empty" << endl;
                return;
        }

        float timeMemory = 0;

        for (int i = 0; i < size; i++) {
                timeMemory = mas[i]->getHDisk();
                if (memoryGB < timeMemory) {
                        mas[i]->show();
                }
        }

}
```

File “Main.h”

```
#pragma once

#include <cstdlib>
#include <sstream>
#include <string>
#include <iostream>
#include <fstream>
#include <regex>
```

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


struct Time {
	int hours;
	int minutes;
	int seconds;
};

struct Version {
	string name;
	int arr[3];
};


#include "Program.h"
#include "InstalledProgram.h"
#include "WorkingProgram.h"
#include "Array.h"
#include "Menu.h"
```

File “main.cpp”

```
#include "Main.h"

int main() {
	Menu use;
	use.menu();



	/*_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
	_CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDERR);
	_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
	_CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDERR);
	_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
	_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDERR);*/
	return _CrtDumpMemoryLeaks();
}
```
