

Лабораторна робота 6. СПАДКУВАННЯ

Тема. Класи. Спадкування.

Мета: отримати знання про парадигму ООП – спадкування. Навчитися застосовувати отримані знання на практиці.

1. Завдання до роботи

Загальне завдання. Оберіть завдання для створення класу-спадкоємця відповідно до номера у журналі групи:

Прикладна галузь: програмне забезпечення.

Додаткові поля у класі-спадкоємці: тип зловмисного ПО (перерахування).

2.1 Опис змінних

Базовий клас: C_Program.

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

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

Клас, що відображає ком позитивні відносини з базовим класом: CDate.

Клас-спадкоємець: CMalware.

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

2.2 Опис змінних

`int` `timeOfWork` – поле класу C_Program(час виконання програми).

`int` `size` – поле класу C_Program(розмір програми у мегабайтах).

`int` `amountOfLines` – поле класу C_Program(кількість рядків коду).

`int` `index` – поле класу C_Program(індентифікаційний номер).

`bool` `useInternet` – поле класу C_Program(використовує інтернет).

`string` `name` – поле класу C_Program(назва програми).

`CAuthor` `author` – поле класу CAuthor(автор програми).

`CDate` `date` – поле класу CDate(дата створення програми).

`int` `listSize` – поле класу CList(розмір масиву елементів класу C_Program).

`C_Program*` `list` – поле класу C_Program(масив елементів класу C_Program).

`CList` `list` – об'єкт класу CList.

`sint` `day`, `month`, `year` – поле класу CDate(дата).

`string` `type` – поле класу CMalware(тип зловмисного ПО).

`int` `size` – поле класу MalwareList(тип зловмисного ПО).

`CMalware*` `list` – поле класу MalwareList(масив елементів класу CMalware).

C_Program newProgram, getProgram – змінні для програм, необхідні для роботи програми.

int choise = 1, value = 0, stop = 1 – змінні, необхідні для роботи функції меню.

string fileName – змінна, необхідна для запису назви файлу для роботи з ним.

2.3 Опис методів

void setListSize(int) – запис даних у змінну розміру масиву елементів класу Program (метод класу CList).

int getListSize() const – отримання даних змінної розміру масиву елементів класу Program (метод класу CList).

void createList(int) – створення масиву елементів і заповнення даними (метод класу CList).

void printAll() const – виведення даних елементів у консоль (метод класу CList).

void printOneEl(int) const – виведення даних одного елементу у консоль (метод класу CList).

void addEl(C_Program&) – додавання нового елементу в масив (метод класу CList).

void deleteEl(int) – видалення елемента з масиву (метод класу CList).

int task(int) – знаходження елементів за певним критерієм (метод класу CList).

C_Program getProgramID(int) const – отримання даних елемента по індексу (метод класу CList).

C_Program programs(int) – програми для заповнення списку (метод класу CList).

int linesInFile(string) – знаходження кількості рядків у файлі (метод класу CList).

void readFile(string) – читання даних з файлу (метод класу CList).

stringstream getOneEl(int) const – отримання строки даних (метод класу CList).

void saveToFile(string) – збереження даних у файл (метод класу CList).

void showOneEl(stringstream&) const – читання даних з рядка у консоль (метод класу CList).

void enterNewEl() – введення даних з клавіатури (метод класу CList).

void regexTask() – виведення елементів, назва яких містить 2 слова (метод класу CList).

void sort(comp) – функція сортування списку елементів (метод класу CList).

~CList() – деструктор списку елементів (метод класу CList).

C_Program() – конструктор без параметра (метод класу C_Program)

C_Program(bool, int, int, int, int, string, C_Author, sint, sint, sint) – конструктор класу з параметрами (метод класу C_Program)

C_Program(const C_Program& other) – конструктор копіювання (метод класу C_Program)

~C_Program() – деструктор елемента (метод класу C_Program).

2.4 Опис функцій

`void Menu()` – функція меню.

`void Test_GetProgramID(C_List&, int&)` – тест функції знаходження та повернення об'єкту по індексу.

`void Test_AddEl(C_List&)` – тест функції додавання об'єкта до масиву об'єктів.

`void Test_DelEl(C_List&)` – тест функції видалення об'єкта з масиву об'єктів.

`void Test_Task(C_List&, int&)` – тест функції знаходження елементів за певними критеріями (індивідуальне завдання).

`void Test_Stringstream(CList&)` – тест функції отримання даних зі строки.

`void Test_ReadFile(CList&)` – тест функції читання даних з файлу.

`void Test_RegexTask(CList&)` – тест функції отримання даних програм, які містять 2 слова.

`void Test_Sort(CList& list)` – тест функції сортування списку.

3. Текст програми test.cpp

```
#include "programList.h"
#include "malwareList.h"

#pragma region Program
void Test_GetProgramID(CList&);
void Test_AddEl(CList&);
void Test_DelEl(CList&);
void Test_Task(CList&);
void Test_Stringstream(CList&);
void Test_ReadFile(CList&);
void Test_RegexTask(CList&);
void Test_Sort(CList&);
#pragma endregion Program

#pragma region Malware
void Test_GetProgramID(MalwareList&);
void Test_AddEl(MalwareList&);
void Test_DelEl(MalwareList&);
void Test_Task(MalwareList&);
void Test_Stringstream(MalwareList&);
void Test_ReadFile(MalwareList&);
void Test_RegexTask(MalwareList&);
void Test_Sort(MalwareList&);
#pragma endregion Malware

void Test_Aggregation(CAuthor*);

int main() {
    setlocale(LC_ALL, "Rus");
    int value;
    cout << "Выберите класс для тестирования:" << endl;
    cout << "1) Основной класс" << endl;
    cout << "2) Класс наследник" << endl;
    cout << "Ваш выбор: ";
    cin >> value;
    cout << endl;

    CAuthor authors;
    CAuthor* authorsList;
    authorsList = authors.createList(4);

    if (value == 1)
    {
        CList list;
```

```

        list.createList(4, authorsList);

        Test_GetProgramID(list);
        Test_AddEl(list);
        Test_DelEl(list);
        Test_Task(list);
        Test_Stringstream(list);
        Test_ReadFile(list);
        Test_RegexTask(list);
        Test_Sort(list);
    }
    else if (value == 2)
    {
        MalwareList list2;
        list2.createList(4, authorsList);

        Test_GetProgramID(list2);
        Test_AddEl(list2);
        Test_DelEl(list2);
        Test_Task(list2);
        Test_Stringstream(list2);
        Test_ReadFile(list2);
        Test_RegexTask(list2);
        Test_Sort(list2);
    }
    else { cout << "Неверный символ. Завершение работы." << endl; }

    Test_Aggregation(authorsList);

    if (_CrtDumpMemoryLeaks()) cout << "\n\nЕсть утечка памяти.\n\n";
    else cout << "\n\nУтечка памяти отсутствует.\n\n";

    return 0;
}

#pragma region ProgramTests
void Test_GetProgramID(CList& list)
{
    int expected = 5678;
    int real = list.list[2].getIndex();

    if(expected == real) cout << "Тест получения элемента по ID\t\t выполнен успешно.\n";
    else cout << "Тест получения элемента по ID\t\t не выполнен успешно.\n";
}
void Test_AddEl(CList& list)
{
    C_Program newProgram;
    int size = list.getListSize();
    list.addEl(newProgram);

    if (list.getListSize() > size) cout << "Тест добавления элемента в список\t выполнен успешно.\n";
    else cout << "Тест добавления элемента в список\t не выполнен успешно.\n";
}
void Test_DelEl(CList& list)
{
    int size = list.getListSize();
    list.deleteEl(3);

    if (size > list.getListSize()) cout << "Тест функции удаления\t\t\t выполнен успешно.\n\n";
    else cout << "Тест функции удаления\t\t\t не выполнен успешно.\n\n";
}
void Test_Task(CList& list)
{
    int expected = 1;
    int real = list.task(200);

    cout << endl;
    if(expected == real) cout << "Тест нахождения элементов по параметру\t выполнен успешно.\n";
}

```

```

    else cout << "Тест нахождения элементов по параметру\t не выполнен успешно.\n";
}
void Test_Stringstream(CList& list)
{
    string nameExpected = "1234";
    stringstream funcResult = list.getOneEl(1);
    string nameReal;
    funcResult >> nameReal;

    if (nameExpected == nameReal) cout << "Тест функции stringstream\t\t выполнен успешно." << endl;
    else cout << "Тест функции stringstream\t\t не выполнен успешно." << endl;
}
void Test_ReadFile(CList& list)
{
    string filename = "data.txt";
    list.readFile(filename);

    string expected = "Notepad ";
    string real = list.list[0].getName();

    if (expected == real) cout << "Тест функции чтения из файла\t\t выполнен успешно.\n" << endl;
    else cout << "Тест функции чтения из файла\t\t не выполнен успешно.\n" << endl;
}
void Test_RegexTask(CList& list)
{
    cout << "Здесь должны быть программы, содержащие в названии больше 2 слов:" << endl;
    list.regexTask();
}
void Test_Sort(CList& list)
{
    int beforeSorting = list.list[0].getLines();
    list.sort(list.sortDesc);
    int afterSorting = list.list[0].getLines();
    int expected = 666;

    if (beforeSorting != afterSorting && afterSorting == expected) cout << "Тест функции
сортировки\t\t\t выполнен успешно." << endl;
    else cout << "Тест функции сортировки\t\t\t не выполнен успешно." << endl;
}
#pragma endregion ProgramTests

#pragma region MalwareTests
void Test_GetProgramID(MalwareList& list)
{
    if (list.list[2].getIndex() == 5678) cout << "Тест получения элемента по ID\t\t выполнен
успешно.\n";
    else cout << "Тест получения элемента по ID\t\t не выполнен успешно.\n";
}
void Test_AddEl(MalwareList& list)
{
    CMalware newProgram;
    int size = list.getListSize();
    list.addEl(newProgram);

    if (list.getListSize() > size) cout << "Тест добавления элемента в список\t выполнен
успешно.\n";
    else cout << "Тест добавления элемента в список\t не выполнен успешно.\n";
}
void Test_DelEl(MalwareList& list)
{
    int size = list.getListSize();
    list.deleteEl(3);

    if (size > list.getListSize()) cout << "Тест функции удаления\t\t\t выполнен успешно.\n\n";
    else cout << "Тест функции удаления\t\t\t не выполнен успешно.\n\n";
}
void Test_Task(MalwareList& list)
{

```

```

int expected = 0;
int real = list.task(200);

cout << endl;
if (expected == real) cout << "Тест нахождения элементов по параметру\t выполнен успешно.\n";
else cout << "Тест нахождения элементов по параметру\t не выполнен успешно.\n";
}
void Test_Stringstream(MalwareList& list)
{
    string nameExpected = "1234";
    stringstream funcResult = list.getOneEl(1);
    string nameReal;
    funcResult >> nameReal;

    if (nameExpected == nameReal) cout << "Тест функции stringstream\t\t выполнен успешно." << endl;
    else cout << "Тест функции stringstream\t\t не выполнен успешно." << endl;
}
void Test_ReadFile(MalwareList& list)
{
    string filename = "dataM.txt";
    list.readFile(filename);

    string expected = "Exploit";
    string real = list.list[0].getType();

    if (expected == real) cout << "Тест функции чтения из файла\t\t выполнен успешно." << endl;
    else cout << "Тест функции чтения из файла\t\t не выполнен успешно." << endl;
}
void Test_RegexTask(MalwareList& list)
{
    cout << "Здесь должны быть программы, содержащие в названии больше 2 слов:" << endl;
    list.regexTask();
}
void Test_Sort(MalwareList& list)
{
    int beforeSorting = list.list[0].getLines();
    list.sort(list.sortDesc);
    int afterSorting = list.list[0].getLines();
    int expected = 666;

    if (beforeSorting != afterSorting && afterSorting == expected) cout << "Тест функции
сортировки\t\t\t выполнен успешно." << endl;
    else cout << "Тест функции сортировки\t\t\t не выполнен успешно." << endl;
}
}
#pragma endregion MalwareTests

void Test_Aggregation(CAuthor* list)
{
    string expected = "Lambda";
    string real = (list + 1)->getAuthor();

    if (expected == real) cout << "Тест агрегации\t\t\t\t выполнен успешно." << endl;
    else cout << "Тест агрегации\t\t\t\t не выполнен успешно." << endl;
}
}

```

programList.h

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

class CList {
private:
    int listSize;

public:
    C_Program* list;

    void setListSize(int);
    int getListSize() const;

    void createList(int, CAuthor*);
    void printAll() const;
    void printOneEl(int) const;
    void addEl(C_Program&);
    void deleteEl(int);
    int task(int);
    int linesInFile(string);
    void readFile(string);
    void saveToFile(string);
    stringstream getOneEl(int) const;
    void showOneEl(stringstream&) const;
    void enterNewEl();
    void regexTask();
    void sort(comp);

    static bool sortAsc(const int&, const int&);
    static bool sortDesc(const int&, const int&);

    C_Program getProgramID(int) const;
    C_Program programs(int, CAuthor*);
    ~CList();
};
```

programList.cpp

```
#include "programList.h"

void CList::createList(int value, CAuthor* authors)
{
    listSize = value;
    list = new C_Program[listSize];

    for (size_t i = 0; i < listSize; i++)
        list[i] = programs(i, authors);
}

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

void CList::printAll() const
{
    cout << endl << setiosflags(ios::left);
    cout << setw(12) << "   Время" << setw(12) << "Размер";
    cout << setw(10) << "Строки" << setw(11) << "Интернет";
    cout << setw(14) << "Индекс" << setw(21) << "Название";
    cout << setw(16) << "Год выпуска" << setw(11) << "Автор";
    cout << endl;

    for (size_t i = 0; i < listSize; i++)
        printOneEl(i);

    cout << endl;
```

```

}
void CList::printOneEl(int number) const
{
    cout << setiosflags(ios::left) << setw(2) << number + 1 << " ";
    cout << setw(10) << list[number].getTime();
    cout << setw(12) << list[number].getSize();
    cout << setw(9) << list[number].getLines();
    cout << setw(12) << boolalpha << list[number].getInternet();
    cout << setw(11) << list[number].getIndex();
    cout << setw(26) << list[number].getName();
    cout << setw(14) << list[number].getYear();
    cout << setw(12) << list[number].getAuthor() << endl;
}
void CList::addEl(C_Program& newProgram)
{
    C_Program* newList = new C_Program[listSize + 1];

    for (size_t i = 0; i < listSize; i++)
        newList[i] = list[i];
    newList[listSize++] = newProgram;
    delete[] list;

    list = new C_Program[listSize];
    for (size_t i = 0; i < listSize; i++)
        list[i] = newList[i];

    delete[] newList;
    cout << "Элемент добавлен." << endl;
}
void CList::deleteEl(int index)
{
    if (listSize == 0)
    {
        cout << "список программ пуст. возвращение с выбору действий." << endl;
        return;
    }
    if (index <= 0 || index > listSize)
    {
        cout << "ошибка. неверный номер элемента. возвращение." << endl;
        return;
    }

    C_Program* newList = new C_Program[listSize - 1];

    for (size_t i = 0; i < index - 1; i++)
        newList[i] = list[i];
    for (size_t i = index - 1, j = index; j < listSize; i++, j++)
        newList[i] = list[j];
    delete[] list;

    list = new C_Program[listSize--];
    for (size_t i = 0; i < listSize; i++)
        list[i] = newList[i];
    delete[] newList;

    return;
}
int CList::task(int minimalSize)
{
    int size = 0;

    for (size_t i = 0; i < listSize; i++)
        if (list[i].getSize() > minimalSize && list[i].getInternet() == false)
        {
            printOneEl(i);
            size++;
        }

    return size;
}

```



```

}
int CList::linesInFile(string filename)
{
    int size = 0;
    string line;
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-
Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");

    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Невозможно открыть файл. Возвращение в меню." << endl;
        return 0;
    }

    while (getline(fin, line))
    {
        if (regex_match(line, regular)) size++;
        else cout << "Строка в файле не прошла проверку." << endl;
    }

    fin.close();
    return size;
}
void CList::readFile(string filename)
{
    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Неверное название файла. Повторите попытку." << endl;
        return;
    }

    string line, var;
    int size = CList::linesInFile(filename);
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-
Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");
    int i = 0, a = 0;
    bool b;

    delete[] list;
    list = new C_Program[size];
    while (getline(fin, line) && i < size)
    {
        if (regex_match(line.c_str(), regular))
        {
            int time, size, lines, index;
            sint day, month, year;
            string author;
            bool internet;
            string internet2;
            string name, name2;
            std::istringstream temp(line);

            temp >> index;
            temp >> time;
            temp >> size;
            temp >> lines;
            temp >> internet2;
            temp >> author;
            temp >> day;
            temp >> month;
            temp >> year;
            temp >> name;
            temp >> name2;

            if (internet2 == "true") internet = true;
            else internet = false;
        }
    }
}

```

```

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

        do {
            b = 0;

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

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

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

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

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

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

        } while (b == 1);

        C_Program newElement(internet, time, size, lines, index, name, author, day,
month, year);
        list[i++] = newElement;
    }

    setListSize(size);
    fin.close();
    cout << endl << "Чтение из файла завершено." << endl;
}
void CList::saveToFile(string filename)
{
    std::ofstream fout(filename);

    fout.setf(ios::left);
    fout << setw(12) << "   Время" << setw(12) << "Размер";
    fout << setw(13) << "Строки" << setw(11) << "Интернет";
    fout << setw(15) << "Индекс" << setw(24) << "Название";
    fout << setw(16) << "Год выпуска" << setw(11) << "Автор";

```

```

fout << endl;

for (size_t i = 0; i < getListSize(); i++)
{
    fout << setiosflags(ios::left) << setw(2) << i + 1 << " ";
    fout << setw(10) << list[i].getTime();
    fout << setw(12) << list[i].getSize();
    fout << setw(12) << list[i].getLines();
    fout << setw(12) << boolalpha << list[i].getInternet();
    fout << setw(15) << list[i].getIndex();
    fout << setw(26) << list[i].getName();
    fout << setw(14) << list[i].getYear();
    fout << setw(12) << list[i].getAuthor() << endl;
}

cout << "Запись в файл завершена." << endl;
fout.close();
}

stringstream CList::getOneEl(int value) const
{
    stringstream temp;

    temp << " " << list[value].getIndex() << " " << list[value].getTime() << " " <<
list[value].getSize()
    << " " << list[value].getLines() << " " << list[value].getInternet() << " " <<
list[value].getAuthor()
    << " " << list[value].getDay() << " " << list[value].getMonth() << " " <<
list[value].getYear()
    << " " << list[value].getName();

    return temp;
}

void CList::showOneEl(stringstream& line) const
{
    int time, size, lines, index;
    sint day, month, year;
    string name, name2;
    string internet2;
    bool internet;
    string author;

    line >> index;
    line >> time;
    line >> size;
    line >> lines;
    line >> internet2;
    line >> author;
    line >> day;
    line >> month;
    line >> year;
    line >> name;
    line >> name2;

    if (internet2 == "1") internet = true;
    else internet = false;

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

    cout << endl << setiosflags(ios::left);
    cout << setw(12) << "   Время" << setw(12) << "Размер";
    cout << setw(10) << "Строки" << setw(11) << "Троян";
    cout << setw(14) << "Индекс" << setw(21) << "Название";
    cout << setw(16) << "Год выпуска" << setw(11) << "Автор";
    cout << endl << "   ";

    cout << setw(10) << time;
    cout << setw(12) << size;
    cout << setw(10) << lines;

```

```

    cout << setw(11) << boolalpha << internet;
    cout << setw(10) << index;
    cout << setw(27) << name;
    cout << setw(12) << year;
    cout << setw(15) << author;
    cout << endl;
}
C_Program CList::getProgramID(int id) const
{
    C_Program newProgram;

    for (size_t i = 0; i < listSize; i++)
        if (list[i].getIndex() == id)
        {
            printOneEl(i);
            newProgram = list[i];
            return newProgram;
        }
    cout << "\nПрограммы с таким ID нету.\n" << endl;
    return newProgram;
}
C_Program CList::programs(int valueX, CAuthor* authors)
{
    C_Program standartProgram;

    if (valueX == 1)
    {
        C_Program Program1(true, 222, 222, 222, 1234, "Skype", *(authors), 2, 12, 2002);
        return Program1;

        return Program1;
    }
    else if (valueX == 2)
    {
        C_Program Program2(true, 333, 333, 666, 5678, "Standart Calculator", *(authors + 1),
13, 3, 1993);
        return Program2;
    }
    else if (valueX == 3)
    {
        C_Program Program3(false, 444, 444, 444, 9532, "Domino Super", *(authors + 2), 14, 4,
1944);
        return Program3;
    }
    else if (valueX == 4)
    {
        C_Program Program4(false, 555, 555, 555, 4356, "Text editor", *(authors + 3), 5, 5,
1995);
        return Program4;
    }
    return standartProgram;
}
void CList::enterNewEl()
{
    int time, size, lines, index;
    sint day, month, year;
    string name, name2;
    string internet2;
    bool internet;
    string author;
    string data;

    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZА-
Я]+[\\wА-Яа-я,.;:-]* [\\wА-Яа-я,.;:-]*)");

    cout << "Введите данные в линию в таком порядке:";
    cout << "Индекс Время Размер Строки Троян(true/false) Компания День Месяц Год Название" <<
endl;

```

```

cin.ignore();
getline(cin, data);

data = data + " ";

if (regex_match(data, regular))
{
    std::istringstream temp(data);

    temp >> index;
    temp >> time;
    temp >> size;
    temp >> lines;
    temp >> internet2;
    temp >> author;
    temp >> day;
    temp >> month;
    temp >> year;
    temp >> name;
    temp >> name2;

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

    if (internet2 == "true") internet = true;
    else internet = false;

    C_Program newProgram(internet, time, size, lines, index, name, author, day, month,
year);
    addEl(newProgram);
}
else
{
    cout << endl << "Было введено неправильное имя. Формат имени:" << endl;
    cout << "Первое слово в названии программы не должно начинаться с маленькой буквы." <<
endl;

    cout << "Не должно содержать символы." << endl;
    cout << "Завершение работы функции." << endl;
}

return;
}
void CList::regexTask()
{
    regex regular("(^[A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]+)");
    int listSize = getListSize();

    for (size_t i = 0; i < listSize; i++)
        if (regex_match(list[i].getName(), regular))
            printOneEl(i);

    cout << endl;
}

bool CList::sortAsc(const int& a, const int& b) { return a > b; }
bool CList::sortDesc(const int& a, const int& b) { return a < b; }
void CList::sort(comp condition)
{
    C_Program temp;
    int size = getListSize();
    bool pr;

    do {
        pr = 0;
        for (size_t i = 0; i < size-1; i++)
        {
            if (condition(list[i].getLines(), list[i+1].getLines()))
            {

```

```

        temp = list[i];
        list[i] = list[i + 1];
        list[i + 1] = temp;
        pr = 1;
    }
} while (pr == 1);
}

CList::~CList()
{
    //cout << "\nВызвался деструктор" << endl;
    delete[] list;
}

```

main.cpp

```

#include "programList.h"
#include "malwareList.h"

void menu();

int main()
{
    setlocale(LC_ALL, "Rus");
    menu();

    if (_CrtDumpMemoryLeaks()) cout << endl << "Есть утечка памяти." << endl;
    else cout << endl << "Утечка памяти отсутствует." << endl;

    return 0;
}

void menu()
{
    auto choise = 1, choise2 = 0;
    string fileName; //название файла
    auto value = 0, stop = 1;
    string::size_type n;

#pragma region Authors
    CAuthor author; //переменная поля автор
    CAuthor* listAuthors; //список авторов
    listAuthors = author.createList(4);
#pragma endregion Authors

#pragma region Program
    CList list; //список программ
    C_Program getProgram; //программа, которая вернётся при получении ID
    C_Program newProgram; //программа для добавления в список
    int size; //количество элементов больше определённого размера
    stringstream str;
    list.createList(4, listAuthors);
#pragma endregion Program

#pragma region Malware
    MalwareList list2; //список вредоносного ПО
    CMalware getProgram2;
    CMalware newProgram2;
    int size2;
    stringstream str2;
    list2.createList(4, listAuthors);
#pragma endregion Malware

    cout << endl << "Выберите команду для работы со списком: ";
    while (stop != 0)
    {
        if (list.getListSize() == 0)

```

```

{
    cout << "Список пуст. Что вы хотите сделать?" << endl;
    cout << "1) Добавить элемент вручную" << endl;
    cout << "2) Прочитать данные из файла" << endl;
    cout << "3) Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;

    switch (choise)
    {
    case 1:
        cout << "Базовый класс" << endl;
        list.enterNewEl();
        cout << "Класс наследник" << endl;
        list2.enterNewEl();
        break;

    case 2:
        cout << "Введите название файла для чтения данных для базового класса: ";
        cin >> fileName;
        cout << endl;

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

        cout << "Базовый класс" << endl;
        list.readFile(fileName);

        cout << "Введите название файла для чтения данных для базового класса: ";
        cin >> fileName;
        cout << endl;

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

        cout << "Класс наследник" << endl;
        list2.readFile(fileName);

        break;

    case 3:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;

    default:
        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
    }
}
else
{
    cout << endl;
    cout << "1) Вывод на экран" << endl;
    cout << "2) Работа с файлами" << endl;
    cout << "3) Сортировка данных" << endl;
    cout << "4) Удаление элемента" << endl;
    cout << "5) Добавление элементов" << endl;
    cout << "6) Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{

```

```

case 1:
    cout << "Выберите команду:" << endl;
    cout << "1) Вывести весь список на экран" << endl;
    cout << "2) Вывести один элемент на экран по номеру" << endl;
    cout << "3) Вывести список программ меньше определённого размера и не трояны"
<< endl;

endl;

    cout << "4) Вывести список программ, названия которых состоят из 2 слов" <<

    cout << "5) Получить строку с данными" << endl;
    cout << "6) Вывести программу по ID" << endl;
    cout << "7) Вернуться к выбору действий" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    switch (choise2)
    {
    case 1:
        cout << "Базовый класс" << endl;
        list.printAll();
        cout << "Класс наследник" << endl;
        list2.printAll();

        break;

    case 2:
        cout << "Введите номер элемента, который надо вывести: ";
        cin >> value;
        cout << endl;

        if (value <= 0 || value > list.getListSize())
        {
            cout << "Неверный номер элемента. Повторите попытку." << endl;
            break;
        }

        cout << "Базовый класс" << endl;
        list.printOneEl(value - 1);
        cout << "Класс наследник" << endl;
        list2.printOneEl(value - 1);

        break;

    case 3:
        cout << "Введите минимальный размер программ: ";
        cin >> value;
        cout << endl;

        cout << "Базовый класс" << endl;
        size = list.task(value);
        cout << "Класс наследник" << endl;
        size2 = list2.task(value);
        break;

    case 4:
        cout << "Базовый класс" << endl;
        list.regexTask();
        cout << "Класс наследник" << endl;
        list2.regexTask();
        break;

    case 5:
        cout << "Введите номер элемента, который вы хотите получить: ";
        cin >> value;
        cout << endl;

        cout << "Базовый класс" << endl;
        str = list.getOneEl(value - 1);

```



```

        list.showOneEl(str);
        cout << "Класс наследник" << endl;
        str2 = list2.getOneEl(value - 1);
        list2.showOneEl(str2);
        break;

    case 6:
        cout << "Введите id элемента, которого вы хотите получить: ";
        cin >> value;
        cout << endl;

        cout << "Базовый класс" << endl;
        getProgram = list.getProgramID(value);
        cout << "Класс наследник" << endl;
        getProgram2 = list2.getProgramID(value);
        break;

    case 7:
        break;

    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
    }
    break;
case 2:
    cout << "Выберите команду:" << endl;
    cout << "1) Сохранить данные в файл" << endl;
    cout << "2) Читать данные из файла" << endl;
    cout << "3) Вернуться к выбору" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choice2;
    cout << endl;

    switch (choice2)
    {
    case 1:
        cout << "Введите название файла для записи данных программ: ";
        cin >> fileName;
        cout << endl;

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

        list.saveToFile(fileName);

        cout << "Введите название файла для записи данных вредоносного ПО: ";
        cin >> fileName;
        cout << endl;

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

        list2.saveToFile(fileName);
        break;

    case 2:
        cout << "Введите название файла для чтения данных программ: ";
        cin >> fileName;
        cout << endl;

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

        list.readFile(fileName);

        cout << "Введите название файла для чтения данных вредоносного ПО: ";
        cin >> fileName;
        cout << endl;

```

```

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

        list2.readFile(fileName);
        break;
    case 3:
        break;
    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
    }
    break;
case 3:
    cout << "Сортировать количество строк по:" << endl;
    cout << "1) Возрастанию" << endl;
    cout << "2) Убыванию" << endl;
    cout << "Ваш выбор: ";
    cin >> value;
    cout << endl;

    if (value == 1){
        list.sort(list.sortAsc);
        list2.sort(list.sortAsc);
    } else if (value == 2) {
        list.sort(list.sortDesc);
        list2.sort(list.sortDesc);
    } else cout << "Ошибка. Неверный номер команды." << endl;

    break;

case 4:
    cout << "Введите номер элемента, который хотите удалить: ";
    cin >> value;
    cout << endl;

    list.deleteEl(value);
    list2.deleteEl(value);
    break;

case 5:
    cout << "Выберите команду:" << endl;
    cout << "1) Добавить стандартную программу" << endl;
    cout << "2) Ввести данные с клавиатуры" << endl;
    cout << "3) Вернуться к выбору" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choice2;
    cout << endl;

    switch (choice2)
    {
    case 1:
        list.addEl(newProgram);
        list2.addEl(newProgram2);
        break;

    case 2:
        list.enterNewEl();
        list2.enterNewEl();
        break;

    case 3:
        break;

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

```

```

        break;

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

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

author.deleteList();
return;
}

```

program.cpp

```

#include "program.h"

int C_Program::getTime() const
{
    return timeOfWork;
}

int C_Program::getSize() const
{
    return size;
}

int C_Program::getLines() const
{
    return amountOfLines;
}

int C_Program::getIndex() const
{
    return index;
}

bool C_Program::getInternet()const
{
    return useInternet;
}

string C_Program::getName()const
{
    return name;
}

sint C_Program::getDay()const
{
    return date.getDay();
}

sint C_Program::getMonth()const
{
    return date.getMonth();
}

sint C_Program::getYear()const
{
    return date.getYear();
}

string C_Program::getAuthor() const
{
    return author.getAuthor();
}

void C_Program::setTime(const int valueTime)
{
    timeOfWork = valueTime;
}

```

```

void C_Program::setSize(const int valueSize)
{
    size = valueSize;
}
void C_Program::setLines(const int valueLines)
{
    amountOfLines = valueLines;
}
void C_Program::setInternet(const bool internetStatus)
{
    useInternet = internetStatus;
}
void C_Program::setIndex(const int valueIndex)
{
    index = valueIndex;
}
void C_Program::setName(const string valueName)
{
    name = valueName;
}
void C_Program::setDay(const sint valueDay)
{
    date.setDay(valueDay);
}
void C_Program::setMonth(const sint valueMonth)
{
    date.setMonth(valueMonth);
}
void C_Program::setYear(const sint valueYear)
{
    date.setYear(valueYear);
}
void C_Program::setAuthor(const string valueAuthor)
{
    author.setAuthor(valueAuthor);
}

C_Program::C_Program(bool internet, int time, int size, int lines, int index, string name, CAuthor
author, sint day, sint month, sint year): useInternet(internet), timeOfWork(time), size(size),
amountOfLines(lines), index(index), name(name), author(author), date(day, month, year)
{
    //cout << "\nВызвался конструктор с параметрами";
}
C_Program::C_Program() : useInternet(false), timeOfWork(0), size(0), amountOfLines(0), index(0101),
name("Basic")
{
    //cout << "\nВызвался конструктор по умолчанию.";
}
C_Program::C_Program(const C_Program& other) : useInternet(other.useInternet),
timeOfWork(other.timeOfWork), size(other.size), amountOfLines(other.amountOfLines),
index(other.index), name(other.name), author(other.author), date(other.date)
{
    //cout << "\nВызвался конструктор копирования.";
}
C_Program::~C_Program()
{
    //cout << "\nВызвался деструктор";
}

```

program.h

```
#pragma once

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

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

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;
using std::boolalpha;
using std::regex;
using std::ifstream;
using std::regex_match;
using std::regex_search;
using std::cmatch;
using std::setiosflags;
using std::ios;

#include "author.h"
#include "date.h"

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

class C_Program {
private:
    int timeOfWork;           //average time of program execution
    int size;                 //size of program
    int amountOfLines;        //number of lines in code
    int index;                //index
    bool trojan;              //trojan(yes or no)
    string name;              //name of program
    CAuthor author;           //creator of program
    CDate date;               //date of creating program
public:
    int getTime() const;
    int getSize() const;
    int getLines() const;
    int getIndex()const;
    bool getTrojan()const;
    string getName() const;
    sint getDay()const;
    sint getMonth()const;
    sint getYear()const;
    string getAuthor()const;

    void setTime(const int);
    void setSize(const int);
    void setLines(const int);
    void setIndex(const int);
    void setTrojan(const bool);
    void setName(const string);
    void setDay(const sint);
    void setMonth(const sint);
    void setYear(const sint);
```

```

    void setAuthor(const string);

    C_Program();
    C_Program(bool, int, int, int, int, string, CAuthor, sint, sint, sint);
    C_Program(const C_Program& other);
    ~C_Program();
};

```

author.h

```

#pragma once
#include <iostream>
#include <string>

using std::string;

class CAuthor {
private:
    string companyName;
    int listSize;
    CAuthor* list;

public:
    void setAuthor(const string);
    string getAuthor()const;

    CAuthor* createList(int size);
    void deletelist();

    CAuthor authors(int value);

    CAuthor();
    CAuthor(string author);
    CAuthor(const CAuthor& other);
    ~CAuthor();
};

```

author.cpp

```

#include "author.h"

string CAuthor::getAuthor() const
{
    return companyName;
}

void CAuthor::setAuthor(string name)
{
    companyName = name;
}

CAuthor* CAuthor::createList(int size)
{
    list = new CAuthor[size];
    listSize = size;

    for (size_t i = 0; i < size; i++)
    {
        list[i] = authors(i);
    }

    return list;
}

void CAuthor::deletelist()
{

```

```

        delete[] list;
    }

CAuthor CAuthor::authors(int value)
{
    if (value == 0)
    {
        CAuthor author("Oracle");
        return author;
    }
    else if (value == 1)
    {
        CAuthor author2("Lambda");
        return author2;
    }
    else if (value == 2)
    {
        CAuthor author3("AMD");
        return author3;
    }
    else if (value == 3)
    {
        CAuthor author4("Logitech");
        return author4;
    }
}

CAuthor::CAuthor() : companyName("IBM") {}
CAuthor::CAuthor(string author) : companyName(author) {}
CAuthor::CAuthor(const CAuthor& other) : companyName(other.companyName) {}
CAuthor::~CAuthor() {}

```

date.h

```

#pragma once
typedef short sint;

class CDate
{
private:
    sint day;
    sint month;
    sint year;
public:
    void setDay(sint day);
    void setMonth(sint month);
    void setYear(sint year);

    sint getDay() const;
    sint getMonth() const;
    sint getYear() const;

    CDate();
    CDate(sint, sint, sint);
    CDate(const CDate& other);
    ~CDate();
};

```

date.cpp

```
#include "date.h"

void CDate::setDay(sint day)
{
    this->day = day;
}
void CDate::setMonth(sint month)
{
    this->month = month;
}
void CDate::setYear(sint year)
{
    this->year = year;
}

sint CDate::getDay() const
{
    return day;
}
sint CDate::getMonth() const
{
    return month;
}
sint CDate::getYear() const
{
    return year;
}

CDate::CDate() : day(1), month(1), year(1999) {}
CDate::CDate(sint day, sint month, sint year) : day(day), month(month), year(year) {}
CDate::CDate(const CDate& other) : day(other.day), month(other.month), year(other.year) {}
CDate::~CDate() {}
```

malwareList.h

```
#pragma once
#include "malware.h"
class MalwareList
{
private:
    int size;

public:
    CMalware* list;
    void setListSize(int);
    int getListSize() const;

    void createList(int, CAuthor*);
    void printAll() const;
    void printOneEl(int) const;
    void addEl(CMalware&);
    void deleteEl(int);
    int task(int);
    int linesInFile(string);
    void readFile(string);
    void saveToFile(string);
    stringstream getOneEl(int) const;
    void showOneEl(stringstream&) const;
    void enterNewEl();
    int regexTask();
    void sort(comp);

    static bool sortAsc(const int&, const int&);
```



```

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

        CMalware getProgramID(int) const;
        CMalware programs(int, CAuthor*);
        ~MalwareList();
};

```

malware.cpp

```

#include "malware.h"

using std::string;

string CMalware::getType() const
{
    return type;
}

void CMalware::setType(string type)
{
    this->type = type;
}

CMalware::CMalware(bool internet, int time, int size, int lines, int index, string name, CAuthor
author, sint day, sint month, sint year, string type) : C_Program(internet, time, size, lines,
index, name, author, day, month, year), type(type)
{
    //cout << "\nВызвался конструктор с параметрами";
}
CMalware::CMalware() : C_Program(), type("Exploit")
{
    //cout << "\nВызвался конструктор по умолчанию.";
}
CMalware::CMalware(const CMalware& other) : C_Program(other), type(other.type)
{
    //cout << "\nВызвался конструктор копирования.";
}
CMalware::~CMalware()
{
    //cout << "\nВызвался деструктор";
}

```

malware.h

```

#include "program.h"
class CMalware : public C_Program
{
private:
    string type;

public:
    string getType() const;
    void setType(string type);

    CMalware(bool, int, int, int, int, string, CAuthor, sint, sint, sint, string);
    CMalware();
    CMalware(const CMalware& other);
    ~CMalware();
};

```

malwareList.cpp

```
#include "malwareList.h"

void MalwareList::createList(int value, CAuthor* authors)
{
    size = value;
    list = new CMalware[value];

    for (size_t i = 0; i < value; i++)
        list[i] = programs(i, authors);
}

void MalwareList::setListSize(int size) { this->size = size; }
int MalwareList::getListSize() const { return size; }
void MalwareList::printAll() const
{
    cout << endl << setiosflags(ios::left);
    cout << setw(12) << "Время" << setw(12) << "Размер";
    cout << setw(10) << "Строки" << setw(11) << "Интернет";
    cout << setw(14) << "Индекс" << setw(21) << "Название";
    cout << setw(16) << "Год выпуска" << setw(14) << "Автор";
    cout << setw(14) << "Тип" << endl;

    for (size_t i = 0; i < size; i++)
        printOneEl(i);

    cout << endl;
}

void MalwareList::printOneEl(int number) const
{
    cout << setiosflags(ios::left) << setw(2) << number + 1 << " ";
    cout << setw(10) << list[number].getTime();
    cout << setw(12) << list[number].getSize();
    cout << setw(9) << list[number].getLines();
    cout << setw(12) << boolalpha << list[number].getInternet();
    cout << setw(11) << list[number].getIndex();
    cout << setw(26) << list[number].getName();
    cout << setw(14) << list[number].getYear();
    cout << setw(12) << list[number].getAuthor();
    cout << setw(14) << list[number].getType() << endl;
}

void MalwareList::addEl(CMalware& newProgram)
{
    CMalware* newList = new CMalware[size + 1];

    for (size_t i = 0; i < size; i++)
        newList[i] = list[i];
    newList[size++] = newProgram;
    delete[] list;

    list = new CMalware[size];
    for (size_t i = 0; i < size; i++)
        list[i] = newList[i];

    delete[] newList;
    cout << "Элемент добавлен." << endl;
}

void MalwareList::deleteEl(int index)
{
    if (size == 0)
    {
        cout << "список программ пуст. возвращение с выбору действий." << endl;
        return;
    }
    if (index <= 0 || index > size)
    {
        cout << "ошибка. неверный номер элемента. возвращение." << endl;
    }
}
```

```

        return;
    }

    CMalware* newList = new CMalware[size - 1];

    for (size_t i = 0; i < index - 1; i++)
        newList[i] = list[i];
    for (size_t i = index - 1, j = index; j < size; i++, j++)
        newList[i] = list[j];
    delete[] list;

    list = new CMalware[size--];
    for (size_t i = 0; i < size; i++)
        list[i] = newList[i];
    delete[] newList;

    return;
}

int MalwareList::task(int minimalSize)
{
    int size2 = 0;

    for (size_t i = 0; i < size; i++)
        if (list[i].getSize() > minimalSize && list[i].getType() == "Trojan")
        {
            printOneEl(i);
            size2++;
        }

    return size2;
}

int MalwareList::linesInFile(string filename)
{
    int size = 0;
    string line;
    regex regular("([\\d]* [\\d]* [\\wA-Яa-я]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");

    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Невозможно открыть файл. Возвращение в меню." << endl;
        return 0;
    }

    while (getline(fin, line))
    {
        if (regex_match(line, regular)) size++;
        else cout << "Строка в файле не прошла проверку." << endl;
    }

    fin.close();
    return size;
}

void MalwareList::readFile(string filename)
{
    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Неверное название файла. Повторите попытку." << endl;
        return;
    }

    string line, var;
    int size = MalwareList::linesInFile(filename);
    regex regular("([\\d]* [\\d]* [\\wA-Яa-я]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");
    int i = 0, a = 0;

```

```

bool b;

delete[] list;
list = new CMalware[size];
while (getline(fin, line) && i < size)
{
    if (regex_match(line.c_str(), regular))
    {
        int time, size, lines, index;
        sint day, month, year;
        string name, name2;
        string internet2;
        bool internet;
        string author;
        string type;

        std::istringstream temp(line);

        temp >> index;
        temp >> time;
        temp >> type;
        temp >> size;
        temp >> lines;
        temp >> internet2;
        temp >> author;
        temp >> day;
        temp >> month;
        temp >> year;
        temp >> name;
        temp >> name2;

        if (internet2 == "true") internet = true;
        else internet = false;

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

        do {
            b = 0;

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

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

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

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

            a = name.find(";");

```

```

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

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

    CMalware newElement(internet, time, size, lines, index, name, author, day,
month, year, type);
    list[i++] = newElement;
}

setListSize(size);
fin.close();
cout << endl << "Чтение из файла завершено." << endl;
}
void MalwareList::saveToFile(string filename)
{
    std::ofstream fout(filename);

    fout.setf(ios::left);
    fout << setw(12) << "   Время" << setw(12) << "Размер";
    fout << setw(13) << "Строки" << setw(11) << "Интернет";
    fout << setw(15) << "Индекс" << setw(24) << "Название";
    fout << setw(16) << "Год выпуска" << setw(14) << "Автор";
    fout << setw(12) << "Тип" << endl;

    for (size_t i = 0; i < getListSize(); i++)
    {
        fout << setiosflags(ios::left) << setw(2) << i + 1 << " ";
        fout << setw(10) << list[i].getTime();
        fout << setw(12) << list[i].getSize();
        fout << setw(12) << list[i].getLines();
        fout << setw(12) << boolalpha << list[i].getInternet();
        fout << setw(15) << list[i].getIndex();
        fout << setw(26) << list[i].getName();
        fout << setw(14) << list[i].getYear();
        fout << setw(12) << list[i].getAuthor();
        fout << setw(12) << list[i].getType() << endl;
    }

    cout << "Запись в файл завершена." << endl;
    fout.close();
}
stringstream MalwareList::getOneEl(int value) const
{
    stringstream temp;

    temp << " " << list[value].getIndex() << " " << list[value].getType() << " " <<
list[value].getTime() << " " << list[value].getSize()
        << " " << list[value].getLines() << " " << list[value].getInternet() << " " <<
list[value].getAuthor()
        << " " << list[value].getDay() << " " << list[value].getMonth() << " " <<
list[value].getYear()
        << " " << list[value].getName();

    return temp;
}
void MalwareList::showOneEl(stringstream& line) const
{

```

```

int time, size, lines, index;
sint day, month, year;
string name, name2;
string internet2;
bool internet;
string author;
string type;

line >> index;
line >> type;
line >> time;
line >> size;
line >> lines;
line >> boolalpha >> internet2;
line >> author;
line >> day;
line >> month;
line >> year;
line >> name;
line >> name2;

if (internet2 == "1") internet = true;
else internet = false;

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

cout << endl << setiosflags(ios::left);
cout << setw(12) << "   Время" << setw(12) << "Размер";
cout << setw(10) << "Строки" << setw(11) << "Интернет";
cout << setw(14) << "Индекс" << setw(21) << "Название";
cout << setw(16) << "Год выпуска" << setw(15) << "Автор";
cout << setw(13) << "Тип" << endl << "   ";

cout << setw(10) << time;
cout << setw(12) << size;
cout << setw(10) << lines;
cout << setw(10) << internet;
cout << setw(14) << index;
cout << setw(25) << name;
cout << setw(12) << year;
cout << setw(15) << author;
cout << setw(14) << type;
cout << endl;
}
CMalware MalwareList::getProgramID(int id) const
{
    CMalware newProgram;

    for (size_t i = 0; i < size; i++)
        if (list[i].getIndex() == id)
        {
            printOneEl(i);
            newProgram = list[i];
            return newProgram;
        }
    cout << "\nПрограммы с таким ID нету.\n" << endl;
    return newProgram;
}
CMalware MalwareList::programs(int valueX, CAuthor* authors)
{
    CMalware standartMalware;

    if (valueX == 1)
    {
        CMalware Malware1(true, 222, 222, 222, 1234, "Skype", *(authors), 2, 12, 2002,
"Keylogger");
        return Malware1;
    }
}

```

```

        return Malware1;
    }
    else if (valueX == 2)
    {
        CMalware Malware2(true, 333, 333, 666, 5678, "Standart Calculator", *(authors + 1),
13, 3, 1993, "Trojan");
        return Malware2;
    }
    else if (valueX == 3)
    {
        CMalware Malware3(false, 444, 444, 444, 9532, "Domino Super", *(authors + 2), 14, 4,
1944, "Virus");
        return Malware3;
    }
    else if (valueX == 4)
    {
        CMalware Malware4(false, 555, 555, 555, 4356, "Text editor", *(authors + 3), 5, 5,
1995, "Bootkit");
        return Malware4;
    }
    return standartMalware;
}
void MalwareList::enterNewEl()
{
    int time, size, lines, index;
    sint day, month, year;
    string name, name2;
    string internet2;
    bool internet;
    string author;
    string type;
    string data;

    regex regular("([\\d]* [\\d]* [\\wA-Яa-я]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]*
[\\d]* [A-ZA-Я]+[\\wA-Яa-я.,;:-]* [\\wA-Яa-я.,;:-]*)");

    cout << "Введите данные в линию в таком порядке:";
    cout << "Индекс Время Тип Размер Строки Интернет(true/false) Компания День Месяц Год
Название" << endl;

    cin.sync();
    getline(cin, data);

    data = data + " ";

    if (regex_match(data, regular))
    {
        std::istringstream temp(data);

        temp >> index;
        temp >> time;
        temp >> type;
        temp >> size;
        temp >> lines;
        temp >> internet2;
        temp >> author;
        temp >> day;
        temp >> month;
        temp >> year;
        temp >> name;
        temp >> name2;

        if (internet2 == "true") internet = true;
        else internet = false;

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

```

```

        CMalware newMalware(internet, time, size, lines, index, name, author, day, month,
year, type);
        addEl(newMalware);
    }
    else
    {
        cout << endl << "Было введено неправильное имя. Формат имени:" << endl;
        cout << "Первое слово в названии программы не должно начинаться с маленькой буквы." <<
endl;

        cout << "Не должно содержать символы." << endl;
        cout << "Завершение работы функции." << endl;
    }

    return;
}
int MalwareList::regexTask()
{
    int value = 0;
    regex regular("(^[A-ZА-Я]+[\\wА-Яа-я,.;:-]* [\\wА-Яа-я,.;:-]+)");
    int listSize = getListSize();

    for (size_t i = 0; i < listSize; i++)
    {
        if (regex_match(list[i].getName(), regular))
        {
            printOneEl(i);
            value++;
        }
    }
    cout << endl;

    return value;
}

bool MalwareList::sortAsc(const int& a, const int& b) { return a > b; }
bool MalwareList::sortDesc(const int& a, const int& b) { return a < b; }
void MalwareList::sort(comp condition)
{
    CMalware temp;
    int size = getListSize();
    bool pr;

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

MalwareList::~MalwareList()
{
    //cout << "\nВызвался деструктор" << endl;
    delete[] list;
}

```


4. Результаты работы программы

Выберите команду для работы со списком:

- 1) Вывод на экран
- 2) Работа с файлами
- 3) Сортировка данных
- 4) Удаление элемента
- 5) Добавление элементов
- 6) Завершение работы

=====

Ваш выбор: 1

Выберите команду:

- 1) Вывести весь список на экран
- 2) Вывести один элемент на экран по номеру
- 3) Вывести список программ меньше определённого размера и не трояны
- 4) Вывести список программ, названия которых состоят из 2 слов
- 5) Получить строку с данными
- 6) Вывести программу по ID
- 7) Вернуться к выбору действий

=====

Ваш выбор: 1

Базовый класс

	Время	Размер	Строки	Интернет	Индекс	Название	Год выпуска	Автор
1)0	0	0	0	false	65	Basic	1999	IBM
2)222	222	222	222	true	1234	Skype	2002	Oracle
3)333	333	666	666	true	5678	Standart Calculator	1993	Lambda
4)444	444	444	444	false	9532	Domino Super	1944	AMD

Класс наследник

	Время	Размер	Строки	Интернет	Индекс	Название	Год выпуска	Автор	Тип
1)0	0	0	0	false	65	Basic	1999	IBM	Exploit
2)222	222	222	222	true	1234	Skype	2002	Oracle	Keylogger
3)333	333	666	666	true	5678	Standart Calculator	1993	Lambda	Trojan
4)444	444	444	444	false	9532	Domino Super	1944	AMD	Virus

- 1) Вывод на экран
- 2) Работа с файлами
- 3) Сортировка данных
- 4) Удаление элемента
- 5) Добавление элементов
- 6) Завершение работы

=====

Ваш выбор: 6

Завершение работы.

Утечка памяти отсутствует.

Выберите класс для тестирования:

- 1) Основной класс
- 2) Класс наследник

Ваш выбор: 2

Тест получения элемента по ID	выполнен успешно.
Элемент добавлен.	
Тест добавления элемента в список	выполнен успешно.
Тест функции удаления	выполнен успешно.

Тест нахождения элементов по параметру	выполнен успешно.
Тест функции stringstream	выполнен успешно.
Строка в файле не прошла проверку.	

Чтение из файла завершено.

Тест функции чтения из файла	выполнен успешно.
Здесь должны быть программы, содержащие в названии больше 2 слов:	

2)666	666	666	true	666	Photo-shop; CPlus54	2012	Abra	Backdoor
--------	-----	-----	------	-----	---------------------	------	------	----------

Тест функции сортировки	выполнен успешно.
Тест агрегации	выполнен успешно.

Утечка памяти отсутствует.

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з спадкуванням класів.

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