

Лабораторна робота 6. ПОЛІМОРФІЗМ

Тема. Класи. Поліморфізм. Абстрактні класи.

Мета: отримати знання про парадигму ООП – поліморфізм; навчитися застосовувати отримані знання на практиці.

1. Завдання до роботи

Загальне завдання. Модернізувати попередню лабораторну роботу шляхом:

- додавання ще одного класу-спадкоємця до базового класу. Поля обрати самостійно;
- базовий клас зробити абстрактним. Додати абстрактні поля;
- розроблені класи-списки поєднуються до одного класу таким чином, щоб він міг працювати як з базовим класом, так і з його спадкоємцями. При цьому серед полів класу-списку повинен бути лише один масив, що містить усі типи класів ієрархії. Оновити методи, що працюють з цим масивом.

2.1. Опис змінних

Базовий абстрактний клас: C_Program.

Клас, що має в собі масиви класів-спадкоємців та методи для роботи з ними: CList.

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

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

Клас-спадкоємець: CMalware.

Клас-спадкоємець: ProgramForSale.

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**` programList – поле класу C_Program(масив елементів класу C_Program).
`sint` day, month, year – поле класу CDate(дата).
`string` type – поле класу CMalware(тип зловмисного ПО).
`int` price – поле класу ProgramForSale (ціна).

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_Program**)` – тест функції знаходження та повернення об'єкту по індексу.

`C_Program** Test_AddEl(CList&, CAuthor*, C_Program**)` – тест функції додавання об'єкта до масиву об'єктів.

`C_Program** Test_DelEl(C_Program**, CList&)` – тест функції видалення об'єкта з масиву об'єктів.

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

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

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

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

`void Test_Sort(C_Program**, CList&)` – тест функції сортування списку.

`void Test_Aggregation(CAuthor*)` – тест функції агрегативних відносин класів.

3. Текст програми test.cpp

```
#include "programList.h"

void Test_GetProgramID(C_Program**);
C_Program** Test_AddEl(CList&, CAuthor*, C_Program**);
C_Program** Test_DelEl(C_Program**, CList&);
void Test_Task(CList&);
void Test_Stringstream(C_Program**, CList&);
void Test_ReadFile(CList&, C_Program**);
void Test_RegexTask(CList&);
void Test_Sort(C_Program**, CList&);
void Test_Aggregation(CAuthor*);

void funcTest();

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

    funcTest();

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

    return 0;
}

void funcTest() {
    CList list;
    CAuthor author;
    CAuthor* authorsList;
    C_Program** programList;

    authorsList = author.createList(6);
    programList = list.createList(4, authorsList);
```

```

Test_GetProgramID(programList);
programList = Test_AddEl(list, authorsList, programList);
programList = Test_DelEl(programList, list);
Test_Task(list);
Test_Stringstream(programList, list);
Test_ReadFile(list, programList);
Test_RegexTask(list);
Test_Sort(programList, list);
Test_Aggregation(authorsList);
}

void Test_GetProgramID(C_Program** list)
{
    int expected = 7896;
    int real = 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";
}

C_Program** Test_AddEl(CList& list, CAuthor* listAuthors, C_Program** programList)
{
    int value = list.getListSize();
    C_Program* newProgram = list.newProgram(listAuthors, 1);
    programList = list.addProgram(newProgram, programList);

    if (list.getListSize() > value&& programList[4]->getIndex() == 6734)
    {
        cout << "Тест добавления элемента в список\t выполнен успешно.\n";
    }
    else cout << "Тест добавления элемента в список\t не выполнен успешно.\n";

    return programList;
}

C_Program** Test_DelEl(C_Program** programList, CList& list)
{
    int size = list.getListSize();
    programList = list.delProgram(1, programList);
    int newSize = list.getListSize();

    if (size > list.getListSize() && programList[1]->getIndex() == 7896)
    {
        cout << "Тест функции удаления\t\t\t выполнен успешно.\n\n";
    }
    else cout << "Тест функции удаления\t\t\t не выполнен успешно.\n\n";

    return programList;
}

void Test_Task(CList& 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(C_Program** programList, CList& list)
{
    string nameExpected = "65";
    stringstream funcResult = programList[0]->getStr();
    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, C_Program** programList)

```

```

{
    int expected = 3;
    int real = list.linesInFile("data.txt");

    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(CList& list)
{
    int expected = 0;
    int real = list.regexTask(list);

    if (real == expected) cout << "Тест функции regex\t\t\t выполнен успешно." << endl;
    else cout << "Тест функции regex\t\t\t не выполнен успешно." << endl;
}
void Test_Sort(C_Program** programList, CList& list)
{
    int beforeSorting = programList[0]->getLines();
    list.sort(list.sortDesc);
    int afterSorting = programList[0]->getLines();
    int expected = 745;

    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;
}
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;
}

```

main.cpp

```

#include "programList.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()
{
    C_Program** programList;
    string fileName;           //название файла
    CList list;                //список программ
    CAuthor* listAuthors;      //список авторов
    CAuthor author;            //переменная поля автор
    auto chose = 1;
    auto chose2 = 0;
    auto chose3 = 0;
    auto value = 0, stop = 1;
    int result, b, size;
    string::size_type n;
    stringstream str;
}

```

```

listAuthors = author.createList(6);
programList = list.createList(4, listAuthors);

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:
                programList = list.enterNewProgram();
                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");

                list.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 список программ больше определённого размера за авторством  
Microsoft" << endl;
                cout << "3) Вывести список программ, названия которых состоят из 2 слов" <<
endl;

```

```

cout << "4) Вывести программу по ID" << endl;
cout << "5) Вернуться к выбору действий" << endl;
cout << "=====" << endl;
cout << "Ваш выбор: ";
cin >> choise2;
cout << endl;

switch (choise2)
{
case 1:
    list.printAll(programList);
    break;

case 2:
    cout << "Введите минимальный размер программ: ";
    cin >> value;
    cout << endl;

    size = list.task(value);
    break;

case 3:
    size = list.regexTask(list);

    if (size == 0)
    {
        cout << "Отсутствуют программы, содержащие 2 слова в названии." <<
endl;

    }

    break;

case 4:
    cout << "Введите id элемента, которого вы хотите получить: ";
    cin >> value;
    cout << endl;

    result = list.getProgramID(value, programList);
    if (result == -5)
    {
        cout << "Элемент с таким ID отсутствует." << endl;
        break;
    }

    list.getOneEl(result);
    break;

case 5:
    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 >> choise2;
    cout << endl;

    switch (choise2)
    {
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);
        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");

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

    break;

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

    result = list.getProgramID(value, programList);
    if (result == -5)
    {
        cout << "Элемент с таким ID отсутствует." << endl;
        break;
    }

    programList = list.delProgram(result, programList);

    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:
    cout << "1) Добавить программу вредоносного ПО" << endl;
    cout << "2) Добавить программу на продажу" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 == 1)
    {
        C_Program* newProgram = list.newProgram(listAuthors, 1);
        programList = list.addProgram(newProgram, programList);
        cout << "Программа добавлена." << endl;
        break;
    }
    else if (choise3 == 2)
    {
        C_Program* newProgram = list.newProgram(listAuthors, 2);
        programList = list.addProgram(newProgram, programList);
        cout << "Программа добавлена." << endl;
    }
    else
    {
        cout << "Неверный номер выбора." << endl;
    }

    break;

case 2:
    programList = list.enterNewProgram();
    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;
}

```

programList.h

```
#pragma once
#include "programForSale.h"
#include "malware.h"

class CList
{ private:
    int listSize;
    C_Program** programList;

public:
    int getListSize() const;

    C_Program** createList(int, CAuthor*);
    C_Program** addProgram(C_Program*, C_Program**);
    C_Program** delProgram(int, C_Program**);
    C_Program* newProgram(CAuthor*, int);
    void printAll(C_Program** program) const;
    stringstream getOneEl(int) const;
    sint task(int);
    void saveToFile(string);
    int linesInFile(string);
    C_Program** readFile(string);
    C_Program** enterNewProgram();
    int regexTask(CList&);
    int getProgramID(int, C_Program**)const;
    void sort(comp);
    static bool sortAsc(const int&, const int&);
    static bool sortDesc(const int&, const int&);

    ~CList();
};
```

programForSale.h

```
#pragma once
#include "program.h"
class ProgramForSale : public C_Program
{
private:
    int price;

public:
    int getPrice() const;
    void setPrice(int price);

    void print() const override;
    stringstream getStr() const override;
    void writeInFile(ofstream& el) override;

    ProgramForSale();
    ProgramForSale(bool, int, int, int, int, string, CAuthor, sint, sint, sint, int price);
    ProgramForSale(const ProgramForSale& other);
    ~ProgramForSale();
};
```

program.h

```
#pragma once

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

#define _CRT_SECURE_NO_WARNINGS
#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::ofstream;
using std::regex_match;
using std::regex_search;
using std::cmatch;
using std::setiosflags;
using std::ios;

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

class C_Program {
protected:
    int timeOfWork;           //average time of program execution
    int size;                 //size of program
    int amountOfLines;        //number of lines in code
    int index;                //index
    bool useInternet;          //use internet
    string name;               //name of program
    CAuthor author;            //creator of program
    CDate date;                //date of creating program
public:

    virtual int getTime() const;
    virtual int getSize() const;
    virtual int getLines() const;
    virtual int getIndex()const;
    virtual bool getInternet()const;
    virtual string getName() const;
    virtual sint getDay()const;
    virtual sint getMonth()const;
    virtual sint getYear()const;
    virtual string getAuthor()const;

    virtual void print() const = 0;
    virtual stringstream getStr() const = 0;
    virtual void writeInFile(ofstream&) = 0;

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

```
virtual ~C_Program();  
};
```

malware.h

```
#pragma once  
#include "program.h"  
class C_Malware : public C_Program  
{  
private:  
    string type;  
  
public:  
    string getType() const;  
    void setType(string type);  
  
    void print() const override;  
    stringstream getStr() const override;  
    void writeInFile(ofstream& e1) override;  
  
    C_Malware(bool, int, int, int, int, string, C_Author, sint, sint, sint, string);  
    C_Malware();  
    C_Malware(const C_Malware& other);  
    ~C_Malware() override;  
};
```

date.h

```
#pragma once  
typedef short sint;  
  
class C_Date  
{  
private:  
    sint day;  
    sint month;  
    sint year;  
public:  
  
    sint getDay() const;  
    sint getMonth() const;  
    sint getYear() const;  
  
    C_Date();  
    C_Date(sint, sint, sint);  
    C_Date(const C_Date& other);  
    ~C_Date();  
};
```

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();
};
```

programList.cpp

```
#include "programList.h"

int CList::getListSize() const
{
    return listSize;
}

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

    for (size_t i = 0; i < value; i++)
    {
        if (i == 0)
        {
            *(programList + i) = new CMalware();
        }
        else if (i == 1)
        {
            *(programList + i) = new ProgramForSale(true, 121, 222, 532, 1234, "Skype",
*(authors + 1), 2, 12, 2002, 3000);
        }
        else if (i == 2)
        {
            *(programList + i) = new CMalware(true, 1445, 532, 745, 7896, "Spider",
*(authors + 2), 6, 9, 1995, "Keylogger");
        }
        else if (i == 3)
        {
            *(programList + i) = new ProgramForSale(false, 333, 444, 555, 6745,
"Calculator", *(authors + 3), 1, 11, 2001, 200);
        }
        if (i == 4)
        {
            *(programList + i) = new ProgramForSale();
        }
    }
}
```

```

        return programList;
    }
    C_Program** CList::addProgram(C_Program* program, C_Program** list)
    {
        C_Program** newList = new C_Program * [listSize + 1];

        for (size_t i = 0; i < listSize; i++)
        {
            *(newList + i) = *(list + i);
        }
        newList[listSize++] = program;
        programList = newList;

        delete list;
        cout << "Элемент добавлен." << endl;

        return programList;
    }
    C_Program** CList::delProgram(int value, C_Program** list)
    {
        if (listSize == 0)
        {
            cout << "список программ пуст. возвращение с выбору действий." << endl;
            return NULL;
        }
        if (value <= 0 || value > listSize)
        {
            cout << "ошибка. неверный номер элемента. возвращение." << endl;
            return NULL;
        }

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

        for (size_t i = 0; i < value; i++)
            *(newList + i) = *(list + i);
        for (size_t i = value, j = value+1; j < listSize; i++, j++)
            *(newList + i) = *(list + j);

        delete* (programList + value);
        listSize--;
        programList = newList;
        delete list;
        return programList;
    }
    C_Program* CList::newProgram(CAuthor* list, int value)
    {
        if (value == 1)
        {
            C_Program* Program = new C_Malware(true, 444, 555, 666, 6734, "Stealer", *(list + 4),
30, 11, 1967, "Trojan");
            return Program;
        }
        else
        {
            C_Program* Program = new ProgramForSale(false, 34, 69, 534, 5378, "Randomizer", *(list
+ 5), 15, 15, 2015, 1000000);
            return Program;
        }
    }
    void CList::printAll(C_Program** program) const
    {
        auto i = 0;
        if (listSize == 0)
        {
            cout << "Список пуст." << endl << endl;
            return;
        }
        else if (listSize < i || i < 0)

```

```

{
    cout << "Неверный номер элемента." << endl << endl;
}

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/Цена" << endl;

for (; i < listSize; i++)
{
    cout << setiosflags(ios::left) << setw(2) << i + 1 << " ";
    program[i]->print();
}

cout << endl;
}

stringstream CList::getOneEl(int value) const
{
    stringstream temp = programList[value]->getStr();

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

    temp >> index;
    temp >> data;
    temp >> time;
    temp >> size;
    temp >> lines;
    temp >> boolalpha >> internet2;
    temp >> author;
    temp >> day;
    temp >> month;
    temp >> year;
    temp >> name;
    temp >> 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/Цена" << 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) << data;
    cout << endl;
}

```

```

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

        for (size_t i = 0; i < listSize; i++)
        {
            if (programList[i]->getSize() > minimalSize&& programList[i]->getName() ==
"Microsoft")
            {
                programList[i]->print();
                size++;
            }
        }

        if (size == 0)
        {
            cout << "Программы с такими параметрами отсутствуют." << endl;
        }

        return size;
    }
    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(14) << "Автор";
        fout << setw(12) << "Тип/Цена" << endl;

        for (size_t i = 0; i < getListSize(); i++)
        {
            fout << setiosflags(ios::left) << setw(2) << i + 1 << " ";
            programList[i]->writeInFile(fout);
        }

        cout << "Запись в файл завершена." << endl;
        fout.close();
    }
    int CList::linesInFile(string filename)
    {
        int size = 0;
        string line;
        regex expressionMalware("([\\d]* [\\d]* [\\wA-Яa-я]* [\\d]* [\\d]* [true|false]* [\\w]*
[\\d]* [\\d]* [\\d]* [A-ZА-Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");
        regex expressionProgramForSelling("([\\d]* [\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]*
[\\d]* [\\d]* [\\d]* [A-ZА-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, expressionMalware) || regex_match(line,
expressionProgramForSelling))
                size++;
            else cout << "Строка в файле не прошла проверку." << endl;
        }

        fin.close();
        return size;
    }

```



```

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

    string line, var;
    int size = CList::linesInFile(filename);
    regex expressionMalware("([\\d]* [\\d]* [A-Za-zA-Яa-я]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");
    regex expressionProgramForSelling("([\\d]* [\\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;

    int time, size2, lines, index;
    sint day, month, year;
    string name, name2;
    string internet2;
    bool internet;
    string author;
    string type;
    int price;

    delete[] programList;
    programList = new C_Program * [size];
    while (getline(fin, line))
    {
        if (regex_match(line.c_str(), expressionMalware))
        {
            std::istringstream temp(line);

            temp >> index;
            temp >> time;
            temp >> type;
            temp >> size2;
            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);

programList[i] = new CMalware(internet, time, size2, lines, index, name,
author, day, month, year, type);
i++;
}
if (regex_match(line.c_str(), expressionProgramForSelling))
{
    std::istringstream temp(line);

    temp >> index;
    temp >> time;
    temp >> price;
    temp >> size2;
    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);

    programList[i] = new ProgramForSale(internet, time, size2, lines, index, name,
author, day, month, year, price);
    i++;
}

}

listSize = size;
fin.close();
cout << endl << "Чтение из файла завершено." << endl;

return programList;
}
C_Program** CList::enterNewProgram()
{
    int time, size, lines, index;
    sint day, month, year;
    string name, name2;
    string internet2;
    bool internet;
    string author;
    string data;

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

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

```

```

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

data = data + " ";

if (regex_match(data, expressionMalware))
{
    string type;
    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);

    C_Program* Program = new CMalware(internet, time, size, lines, index, name, author,
day, month, year, type);
    addProgram(Program, programList);
}
else if (regex_match(data, expressionProgramForSelling))
{
    int price;
    std::istringstream temp(data);

    temp >> index;
    temp >> time;
    temp >> price;
    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);

    C_Program* Program = new ProgramForSale(internet, time, size, lines, index, name,
author, day, month, year, price);
    addProgram(Program, programList);
}
else
{
    cout << endl << "Было введено неправильное имя." << endl;
    cout << "Завершение работы функции. " << endl;
}

return programList;

```

```

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

    for (size_t i = 0; i < listSize; i++)
    {
        if (regex_match(program.programList[i]->getName(), regular))
        {
            program.programList[i]->print();
            value++;
        }
    }
    cout << endl;

    return value;
}
int CList::getProgramID(int id, C_Program** programList) const
{
    int size = 0;

    for (size_t i = 0; i < listSize; i++)
        if (programList[i]->getIndex() == id)
        {
            return i;
        }

    if (size == 0)
    {
        return size = -5;
        cout << "\nПрограммы с таким ID нету.\n" << endl;
    }
}
void CList::sort(comp condition)
{
    C_Program* temp;
    bool pr;

    do {
        pr = 0;
        for (size_t i = 0; i < listSize - 1; i++)
        {
            if (condition(programList[i]->getLines(), programList[i + 1]->getLines()))
            {
                temp = *(programList + i);
                *(programList + i) = *(programList + i + 1);
                *(programList + i + 1) = temp;
                pr = 1;
            }
        }
    } while (pr);
}
bool CList::sortAsc(const int& a, const int& b) { return a > b; }
bool CList::sortDesc(const int& a, const int& b) { return a < b; }

CList::~CList()
{
    for (size_t i = 0; i < listSize; i++)
    {
        delete programList[i];
    }

    delete[] programList;
}

```

programForSale.cpp

```
#include "programForSale.h"

void ProgramForSale::setPrice(int price)
{
    this->price = price;
}
int ProgramForSale::getPrice() const
{
    return price;
}

void ProgramForSale::print() const
{
    cout << setw(10) << timeOfWork;
    cout << setw(12) << size;
    cout << setw(9) << amountOfLines;
    cout << setw(12) << boolalpha << useInternet;
    cout << setw(11) << index;
    cout << setw(26) << name;
    cout << setw(14) << date.getYear();
    cout << setw(12) << author.getAuthor();
    cout << setw(14) << price << endl;
}
stringstream ProgramForSale::getStr() const
{
    stringstream temp;

    temp << " " << index << " " << price << " " << timeOfWork << " " << size << " " <<
amountOfLines
    << " " << useInternet << " " << author.getAuthor() << " " << date.getDay()
    << " " << date.getMonth() << " " << date.getYear() << " " << name;

    return temp;
}
void ProgramForSale::writeInFile(ofstream& el)
{
    el << setw(10) << timeOfWork << setw(12) << size << setw(12) << amountOfLines
    << setw(12) << useInternet << setw(15) << index << setw(26) << name
    << setw(14) << date.getYear() << setw(12) << author.getAuthor()
    << setw(12) << price << endl;
}

ProgramForSale::ProgramForSale() : C_Program(), price(2000) {}
ProgramForSale::ProgramForSale(bool internet, int time, int size, int lines, int index, string name,
CAuthor author, sint day, sint month, sint year, int price) : C_Program(internet, time, size, lines,
index, name, author, day, month, year), price(price) {}
ProgramForSale::ProgramForSale(const ProgramForSale& other) : C_Program(other), price(other.price)
{}
ProgramForSale::~~ProgramForSale() {}
```

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();
}

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";
}
```

malware.cpp

```
#include "malware.h"

using std::string;

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

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

void CMalware::print() const
{
    cout << setw(10) << timeOfWork;
    cout << setw(12) << size;
    cout << setw(9) << amountOfLines;
    cout << setw(12) << boolalpha << useInternet;
    cout << setw(11) << index;
    cout << setw(26) << name;
    cout << setw(14) << date.getYear();
    cout << setw(12) << author.getAuthor();
    cout << setw(14) << type << endl;
}

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

    temp << " " << index << " " << type << " " << timeOfWork << " " << size << " " <<
amountOfLines
    << " " << useInternet << " " << author.getAuthor() << " " << date.getDay()
    << " " << date.getMonth() << " " << date.getYear() << " " << name;

    return temp;
}

void CMalware::writeInFile(ofstream& el)
{
    el << setw(10) << timeOfWork << setw(12) << size << setw(12) << amountOfLines
    << setw(12) << useInternet << setw(15) << index << setw(26) << name
    << setw(14) << date.getYear() << setw(12) << author.getAuthor()
    << setw(12) << type << endl;
}

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) {}
CMalware::CMalware() : C_Program(), type("Exploit") {}
CMalware::CMalware(const CMalware& other) : C_Program(other), type(other.type) {}
CMalware::~CMalware() {}
```


date.cpp

```
#include "date.h"

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() {}
```

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

Утечка памяти отсутствует.

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з поліморфізмом.

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