

Автор: Єлєсін Артем , КІТ-1196

Дата: 01.06.2020

## Лабораторна робота 7. ПОЛІМОРФІЗМ

Тема. Класи. Поліморфізм. Абстрактні класи.

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

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.

Опис класів

Базовий клас: Ccooperator

Клас, що має в собі масив класів наслідників та методи для роботи з ними: Clist

Клас, що демонструє агрегацію: Cchri

Клас, що демонструє композицію: Cweight

Клас наслідувач базового класу: CcoopK

Клас наслідувач базового класу: CcoopFamily

Опис змінних

std::string nameRect; -ім'я ректора

int ageRect; - вік ректора

int id, age, salary; - айди, вік, заробітна плата

string name; - імя

Cchpi\* rect; - адрес агрегатного класу

Cweight weight; - клас ваги

int amountChild; - кількість дітей

int weight; - вага

string mPlaceWork; - місце роботи

size\_t size; - розмір масиву

Ccooperator\*\* mass; - масив

Опис методів

static bool sortAsc(const int& a, const int& b); - перевіряє  $a > b$

static bool sortDesc(const int& a, const int& b); - перевіряє  $a < b$

Ccooperator\*\* addNewEl(Ccooperator\* el); - додає елемент

Ccooperator\*\* delEl(size\_t n); - видаляє елемент

Ccooperator\* getEl(size\_t n); - повертає елемент

Ccooperator\* creatElK(); - створює елемент типу K

Ccooperator\* creatElF(); - створює елемент типу F

int getSize(); - повертає розмір елементу

void addWhithStr(char a); - додавання через строку

void sortMass(string sprz, Fun s); - сортує масив

void EndEnd(); - видаляє масив

void showAll(); - показує всі елементи масиву

void creatMass(size\_t n); - створює масив

void readFile(string fileName); - читання з файлу

void writeToFile(string fileName); - запис до файлу

void End(); - видаляє масив залишаючи агрегатні об'єкти

## Текст програми

### Cooperator.h

```
# #pragma once
#include <sstream>
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "Cchpi.h"
#include "Cweight.h"
#include <locale>
using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::ifstream;
using std::stringstream;
using std::ofstream;
class Ccooperator
{
protected:
    int id, age, salary;
    string name;
    Cchpi* rect;
    Cweight weight;
public:

    virtual void setName(string name)final;

    virtual void setId(const int id)final;

    virtual void setAge(const int age)final;

    virtual void setSalary(const int salary)final;

    virtual void setRect(Cchpi* rect)final;

    virtual void setWeight(Cweight w)final;

    virtual void delRect()final;

    virtual int getId()const final;

    virtual int getAge()const final;
    virtual int getSalary()const final;

    virtual string getName()final;

    virtual Cchpi* getRect()final;

    virtual Cweight getWeight()final;

    virtual string getString() = 0;
```

```

        virtual void setmPlaceWork(const string a)= 0;

        virtual string getmPlaceWork()const = 0;

        virtual void setAmountChild(const int n) = 0;

        virtual int getAmountChild() const=0;

        Ccooperator();

        Ccooperator(int a, int b, int c, const string f, Cweight d, Cchpi* i);

        Ccooperator(const Ccooperator& a);

        virtual ~Ccooperator() = default;
};

```

## Cooperator.cpp

```

#include "Ccooperator.h"

void Ccooperator::setName(string name) {
    this->name = name;
}
string Ccooperator::getName() {
    return this->name;
}
void Ccooperator::setId(const int id) {
    this->id = id;
}
void Ccooperator::setAge(const int age) {
    this->age = age;
}
void Ccooperator::setRect(Cchpi* rect)
{
    this->rect = rect;
}
void Ccooperator::setWeight(Cweight w) {
    this->weight = w;
}
void Ccooperator::delRect()
{
    if(rect)
        if(rect->getAge()>=0)
            delete rect;
}
Cchpi* Ccooperator::getRect() {
    return rect;
}
Cweight Ccooperator::getWeight() {
    return weight;
}
void Ccooperator::setSalary(const int salary) {
    this->salary = salary;
}
int Ccooperator::getId()const {
    return this->id;
}

```

```

}
int Ccooperator::getAge()const {
    return this->age;
}
int Ccooperator::getSalary()const {
    return this->salary;
}
Ccooperator::Ccooperator() :id(0), age(0), salary(0), weight(0), rect(NULL), name("Ivan")
{
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id << "\n";
}
Ccooperator::Ccooperator(const Ccooperator& a) : id(a.id), age(a.age), salary(a.salary),
weight(a.weight), rect(a.rect), name(a.name) {
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id << "\n";
}
Ccooperator::Ccooperator(int a, int b, int c, const string f, Cweight d, Cchpi* i) :
id(a), age(b), salary(c), weight(d), rect(i), name(f) {
    cout << "\nБыл вызван конструктор по умолчанию в объекте с id: " << id << "\n";
}

```

## Clist.h

```

#pragma once
#include "Ccooperator.h"
#include "CcoopK.h"
#include "CcoopFamily.h"
#include <iostream>
#include <fstream>
#include <regex>
#include <ctype.h>
typedef bool (Fun)(const int&, const int&);
class Clist
{
private:
    size_t size;
    Ccooperator** mass;
public:

    static bool sortAsc(const int& a, const int& b);

    static bool sortDesc(const int& a, const int& b);
    Ccooperator** addNewEl(Ccooperator* el);
    Ccooperator** delEl(size_t n);
    Ccooperator* getEl(size_t n);
    Ccooperator* creatElK();
    Ccooperator* creatElF();
    int getSize();

    void addWhithStr(char a);

    void sortMass(string sprz, Fun s);

    void EndEnd();
    void showAll();
    void creatMass(size_t n);
    void readFile(string fileName);
}

```

```

    void writeToFile(string fileName);

    void End();

    Clist();

    Clist(int n, Ccooperator** m);

    Clist(Clist& l);

    virtual ~Clist( )= default;
};

```

## Clist.cpp

```

#include "Clist.h"

void Clist::addWhithStr(char a)
{
    Ccooperator* temp;
    if(a=='K')
        temp= new CcoopK;
    else
        if (a == 'F')
            temp= new CcoopFamily;
        else {
            return;
        }
    std::stringstream ss1;

    std::cout << "\nВведите данные с клавиатуры в таком порядке: id, age, salary,
weight,name,age rector, name rector,";
    if (a == 'K')
        std::cout << "place work\n";
    else
        std::cout << "amount child\n";
    string tid = " ", tage = " ", tsalary = " ", tweight = " ", tname = " ",trectage="
", trectname = " ", pork=" ";
    int tid1;
    string tname1 = " ";
    cin >> tid >> tage >> tsalary >> tweight >>tname >> trectage >> trectname>>pork;
    ss1 << tid;
    ss1 >> tid1;
    temp->setId(tid1);
    ss1.clear();
    ss1 << tage;
    ss1 >> tid1;
    temp->setSalary(tid1);
    ss1.clear();
    ss1 << tsalary;
    ss1 >> tid1;
    temp->setAge(tid1);
    ss1.clear();
    ss1 << tname;
    ss1 >> tname1;
    temp->setName(tname1);
}

```

```

ss1.clear();
ss1 << tweight;
ss1 >> tid1;
temp->setWeight(tid1);
ss1.clear();
ss1 << trectage;
ss1 >> tid1;
Cchpi* rect= new Cchpi;
rect->setAge(tid1);
ss1.clear();
ss1 << trectname;
ss1 >> tname1;
rect->setName(tname1);
temp->setRect(rect);
if (a=='F') {
    ss1.clear();
    ss1 << pork;
    ss1 >> tid1;
    temp->setAmountChild(tid1);
}
if (a=='K') {
    ss1.clear();
    ss1 << pork;
    ss1 >> tname1;
    temp->setmPlaceWork(tname1);
}
addNewEl(temp);
}

void Clist::sortMass(string sprz, Fun s)
{
    int prz = 0;
    Ccooperator* temp;
    if (sprz == "id") {
        do {
            prz = 0;
            for (size_t i = 1; i < size; i++) {
                if (s(mass[i - 1]->getId(), mass[i]->getId())) {
                    temp = mass[i - 1];
                    mass[i - 1] = mass[i];
                    mass[i] = temp;
                    prz = 1;
                }
            }
        } while (prz != 0);
    }
    if (sprz == "salary") {
        do {
            prz = 0;
            for (size_t i = 1; i < size; i++) {
                if (s(mass[i - 1]->getSalary(), mass[i]->getSalary())) {
                    temp = mass[i - 1];
                    mass[i - 1] = mass[i];
                    mass[i] = temp;
                    prz = 1;
                }
            }
        }
    }
}

```

```

        }
    } while (prz != 0);
}
if (sprz == "age") {
    do {
        prz = 0;
        for (size_t i = 1; i < size; i++) {
            if (s(mass[i - 1]->getAge(), mass[i]->getAge())) {
                temp = mass[i - 1];
                mass[i - 1] = mass[i];
                mass[i] = temp;
                prz = 1;
            }
        }
    } while (prz != 0);
}
}

void Clist::EndEnd()
{
    for (size_t i = 0; i < size; i++) {
        mass[i]->delRect();
        delete mass[i];
    }
    delete mass;
}

void Clist::showAll()
{
    for (size_t i = 0; i < size; i++)
        if (mass[i]) {
            cout << "\nNum: " << i << mass[i]->getString();
        }
}

void Clist::creatMass(size_t n)
{
    size = n;
    mass = new Ccooperator* [n];
    for (size_t i = 0; i < size; i++) {
        mass[i] = new CcoopK;
    }
}

void Clist::readFile(string fileName)
{
    if (mass != NULL)
        End();
    creatMass(0);
    ifstream file;
    file.open(fileName);
    if (!file.is_open())
    {
        cout << " Файл не открыт, давай по новой Миша\n";
        return;
    }
    std::regex regular_main(

```



```

"([0-9]+[ \f\\n\\r\\t\\v]{1})"//id
"([0-9]+[ \f\\n\\r\\t\\v]{1})"//age
"([0-9]+[ \f\\n\\r\\t\\v]{1})"//salary
"([0-9]+[ \f\\n\\r\\t\\v]{1})"//weight
"([A-ZА-Я]{1}[A-ZА-Яa-za-я]+[\\.\\,\\;\\:~-
]{0,1}[ \f\\n\\r\\t\\v]{1}){1}"//name
"([A-Z А-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}[ \f\\n\\r\\t\\v]{1})*"
"([0-9]+[ \f\\n\\r\\t\\v]{1}){1}"//ageRect
"([A-ZА-Яa-za-я0-9]+[\\.\\,\\;\\:~-]{0,1}[ \f\\n\\r\\t\\v]{1}){1}"//nameRect
"([A-Z А-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}[ \f\\n\\r\\t\\v]{1})*"
"([A-Z А-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}){1}"// Place or Child
);

```

```

std::cmatch result;
std::stringstream ss;
int integer;
string str;
string line;
string tstr = "";
Ccooperator* temp;
int maxSize = 0;
char ctype;
//bool prz;
Cweight w;
Cchpi* ch;
while (getline(file, line)) {
    //prz = 1;
    tstr = "";
    if (regex_match(line.c_str(), result, regular_main)) {
        maxSize = result.size()-1;
        if (isdigit(result[maxSize].str()[0])) {
            temp = creatElF();
            ctype = 'F';
        }
        else {
            temp = creatElK();
            ctype = 'K';
        }
    }

    ss << result[1];
    ss >> integer;
    temp->setId(integer);
    ss.clear();
    ss << result[2];
    ss >> integer;
    temp->setAge(integer);
    ss.clear();
    ss << result[3];
    ss >> integer;
    temp->setSalary(integer);
    ss.clear();
    ss << result[4];
    ss >> integer;
    w.setWeigt(integer);
    temp->setWeight(w);
    if (ctype == 'F') {
        do {

```

```

        maxSize--;
    } while (!(isdigit(result[maxSize].str()[0]));
    //if ((isdigit(result[maxSize].str()[0])) &&
(isdigit(result[maxSize - 1].str()[0])))
    //{
    //    prz = 0; // Rector
    //}
}
else {
    do {
        maxSize--;
    } while (!(isdigit(result[maxSize].str()[0]));
    //if ((isdigit(result[maxSize].str()[0]))
&& !(isdigit(result[maxSize - 1].str()[0])))
    //{
    //    prz = 0; //Rector
    //}
}
if (ctype=='F') {
    for (int i = 5; i < maxSize; i++)
    {
        ss.clear();
        ss << result[i];
        ss >> str;
        if ((str + " ") != tstr)
            tstr = tstr + str + " ";
    }
    temp->setName(tstr);
    ss.clear();
    ss << result[maxSize];
    ss >> integer;
    ch = new Cchpi;
    ch->setAge(integer);
    tstr = "";
    for (int i = maxSize+1; i < result.size()-1; i++)
    {
        ss.clear();
        ss << result[i];
        ss >> str;
        if ((str + " ") != tstr)
            tstr = tstr + str + " ";
    }
    ch->setName(tstr);
    temp->setRect(ch);
    ss.clear();
    ss << result[result.size()-1];
    ss >> integer;
    temp->setAmountChild(integer);
    addNewEl(temp);
}

if (ctype == 'K') {
    for (int i = 5; i < maxSize; i++)
    {
        ss.clear();
        ss << result[i];
        ss >> str;
    }
}

```

```

        if ((str + " ") != tstr)
            tstr = tstr + str + " ";
    }
    temp->setName(tstr);
    ss.clear();
    ss << result[maxSize];
    ss >> integer;
    ch = new Cchpi;
    ch->setAge(integer);
    tstr = "";
    for (int i = maxSize + 1; i < result.size()-1; i++)
    {
        ss.clear();
        ss << result[i];
        ss >> str;
        if ((str + " ") != tstr)
            tstr = tstr + str + " ";
    }
    ch->setName(tstr);
    temp->setRect(ch);
    ss.clear();
    ss << result[result.size()-1];
    ss >> str;
    temp->setmPlaceWork(str);

    addNewEl(temp);
    }
    ss.clear();
}
result.end();

}
file.close();
}

void Clist::writeToFile(string fileName)
{
    ofstream file;
    file.open(fileName);
    if (!file.is_open())
    {
        cout << " Файл не открыт, давай по новой Миша\n";
        return;
    }
    for (size_t i = 0; i < size; i++) {
        file << mass[i]->getString();
    }

    file.close();
}

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

bool Clist::sortDesc(const int& a, const int& b)
{

```

```

        return a < b;
    }

Ccooperator** Clist::addNewEl(Ccooperator* el)
{
    Ccooperator** tempMass = new Ccooperator * [(size + 1)];
    for (size_t i = 0; i < size; i++) {
        tempMass[i] = mass[i];
        tempMass[i]->setRect(mass[i]->getRect());
    }
    tempMass[size] = el;
    if (size != 0) {
        End();
    }
    else
        EndEnd();

    size++;
    mass = tempMass;
    return mass;
}

Ccooperator** Clist::delEl(size_t n)
{
    Ccooperator** tempMass = new Ccooperator * [size - 1];
    if (n >= size) {
        cout << "Нельзя удалить несуществующий элемент" << endl;
        return mass;
    }
    for (int i = 0; i < n; i++) {
        tempMass[i] = mass[i];
        tempMass[i]->setRect(mass[i]->getRect());
    }
    for (int i = n + 1; i < size; i++) {
        tempMass[i-1] = mass[i];
        tempMass[i-1]->setRect(mass[i]->getRect());
    }
    mass[n]->delRect();
    delete mass[n];
    delete[] mass;
    size--;
    mass = tempMass;
}

Ccooperator* Clist::getEl(size_t n)
{
    if (n >= size) {
        cout << "Нельзя вернуть несуществующий элемент" << endl;
        return NULL;
    }
    return mass[n];
}

Ccooperator* Clist::creatElK()
{
    return new CcoopK;
}

Ccooperator* Clist::creatElF()

```

```

{
    return new CcoopFamily;
}

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

void Clist::End()
{
    delete [] mass;
}

Clist::Clist(): size(0), mass(NULL)
{
}

Clist::Clist(int n, Ccooperator** m):size(n), mass(m)
{
}

Clist::Clist(Clist& l) : size(l.size), mass(l.mass)
{
}

```

## Cchpi.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
class Cchpi
{
private:
    std::string nameRect;
    int ageRect;
public:
    std::string getName()const;
    int getAge() const;
    void setName(std::string name);
    void setAge(int age);
    Cchpi();
    Cchpi(std::string name, int age);
    Cchpi(const Cchpi& temp);
    ~Cchpi();
};

```

## Cchpi.cpp

```

#include "Cchpi.h"
#include <iostream>
std::string Cchpi::getName()const {
    return nameRect;
}
int Cchpi::getAge() const
{
    return ageRect;
}

```

```

}
void Cchpi::setName(std::string name) {
    this->nameRect = name;
}
void Cchpi::setAge(int age)
{
    this->ageRect = age;
}
Cchpi::Cchpi() : nameRect("E. Sokon"), ageRect(68) {};
Cchpi::Cchpi(std::string name, int age) : nameRect(name), ageRect(age) {};
Cchpi::Cchpi(const Cchpi& temp) : nameRect(temp.nameRect), ageRect(temp.ageRect) {};
Cchpi::~Cchpi() {};

```

## CcoopK.h

```

class CcoopK :
    public Ccooperator
{
private:
    string mPlaceWork;
public:
    void setmPlaceWork(const string a)override;
    string getmPlaceWork()const override;
    string getString() override;
    void setAmountChild(const int n)override;
    int getAmountChild() const override;
    CcoopK();
    CcoopK(string a);
    CcoopK(const CcoopK& a);
    ~CcoopK();
};

```

## CcoopK.cpp

```

#include "CcoopK.h"

void CcoopK::setmPlaceWork(const string a)
{
    this->mPlaceWork = a;
}

string CcoopK::getmPlaceWork() const
{
    return this->mPlaceWork;
}

CcoopK::CcoopK(): mPlaceWork("Kafedra")
{

}

CcoopK::CcoopK(string a): mPlaceWork(a)
{

}

CcoopK::CcoopK(const CcoopK& a): mPlaceWork(a.getmPlaceWork())
{

```

```

}
string CcoopK::getString() {
    stringstream ss;
    ss << "\nId: " << id << "\nAge: " << age << "\nSalary: " << salary << "\nName: "
<< name;
    ss << "\nWeight: " << weight.getWeigt() << "\nPlace work: " << mPlaceWork;
    if (rect != NULL) {
        ss << "\nName Rector: " << rect->getName() << "\nAge Rector: " << rect-
>getAge();
    }
    return ss.str();
}
void CcoopK::setAmountChild(const int n)
{
}
int CcoopK::getAmountChild() const
{
    return 0;
}
CcoopK::~CcoopK()
{
}
}

```

## Cweight.h

```

#pragma once
#include <iostream>
class Cweight
{
private:
    int weight;
public:
    void setWeigt(const int m);
    int getWeigt() const;
    Cweight();
    Cweight(const int m);
    Cweight(const Cweight& m);
    virtual ~Cweight()= default;
};

```

## Cweight.cpp

```

#include "Cweight.h"
void Cweight::setWeigt(const int m) {
    weight = m;
}
int Cweight::getWeigt() const {
    return weight;
}
Cweight::Cweight() : weight(0){}
Cweight::Cweight(const int m): weight(m){}
Cweight::Cweight(const Cweight& m): weight(m.weight){}

```

## CcoopFamily.h

```

#pragma once
#include "Ccooperator.h"
class CcoopFamily :
    public Ccooperator
{
private:
    int amountChild;
public:
    void setAmountChild(const int n)override;
    int getAmountChild() const override;
    void setmPlaceWork(const string a)override;
    string getmPlaceWork()const override;
    string getString() override;
    CcoopFamily();
    CcoopFamily(int amount);
    CcoopFamily(const CcoopFamily& temp);
    ~CcoopFamily();
};

```

## CcoopFamily.cpp

```

#include "CcoopFamily.h"

void CcoopFamily::setAmountChild(const int n)
{
    this->amountChild = n;
}

int CcoopFamily::getAmountChild() const
{
    return this->amountChild;
}

void CcoopFamily::setmPlaceWork(const string a)
{
}

string CcoopFamily::getmPlaceWork() const
{
    return string();
}

string CcoopFamily::getString()
{
    stringstream ss;
    ss << "\nId: " << id << "\nAge: " << age << "\nSalary: " << salary << "\nName: " << name;
    ss << "\nWeight: " << weight.getWeigt() << "\nAmount child: " << amountChild;
    if (rect != NULL) {
        ss << "\nName Rector: " << rect->getName() << "\nAge Rector: " << rect-
>getAge();
    }
    return ss.str();
}

```



```

CcoopFamily::CcoopFamily(): amountChild(0)
{
}

CcoopFamily::CcoopFamily(int amount): amountChild(0)
{
}

CcoopFamily::CcoopFamily(const CcoopFamily& temp): amountChild(temp.amountChild)
{
}

CcoopFamily::~CcoopFamily()
{
}

```

## Test.cpp

```

#define _CRT_SECURE_NO_WARNINGS
#include "Ccooperator.h"
#include "CcoopK.h"
#include "CcoopFamily.h"
#include "Clist.h"
#include <iostream>
#include <locale>
#include <sstream>
#include <iostream>
int main() {
    setlocale(LC_ALL, "rus");
    //данные
    Clist list;

    if (list.sortAsc(0, 1))
    {
        cout << "\nТест 1.1 не пройден\n";
    };

    if (!list.sortAsc(1, 0)) {
        cout << "\nТест 1.1 не пройден\n";
    };
    if (list.sortAsc(1, 1))
    {
        cout << "\nТест 1.1 не пройден\n";
    };

    if (!list.sortDesc(0, 1))
    {
        cout << "\nТест 2.1 не пройден\n";
    };

    if (list.sortDesc(1, 0)) {
        cout << "\nТест 2.1 не пройден\n";
    };
    if (list.sortDesc(1, 1))
    {
        cout << "\nТест 2.1 не пройден\n";
    };
    list.addNewEl(new CcoopK);
}

```

```

    if (list.getSize() != 1) {
        cout << "\nТест 3.1 не пройден\n";
    }; list.addNewEl(new CcoopFamily);
    if (list.getSize() != 2) {
        cout << "\nТест 3.2 не пройден\n";
    };
    list.delEl(0);
    if (list.getSize() != 1) {
        cout << "\nТест 4 не пройден\n";
    };
    list.EndEnd();
    if (_CrtDumpMemoryLeaks())
        cout << "\nMemory leak detected\n";
    else
        cout << "\nMemory is not leak detected\n";
}

```

## Source.cpp

```

#define _CRT_SECURE_NO_WARNINGS
#include "Ccooperator.h"
#include "CcoopK.h"
#include "CcoopFamily.h"
#include "Clist.h"
#include <iostream>
#include <locale>
#include <sstream>
#include <iostream>
typedef bool (Fun)(const int&, const int&);
void menu();
int main() {
    setlocale(LC_ALL, "rus");
    menu();
    if (_CrtDumpMemoryLeaks())
        cout << "\nMemory leak detected\n";
    else
        cout << "\nMemory is not leak detected\n";
}
void menu() {
    int n;
    string str;
    Clist list;
    do {
        n = 0;
        std::cout << "\nВыберите желаемую опцию:" << "\n";
        std::cout << "1 - добавить элемент в список." << "\n";
        std::cout << "2 - удалить элемент из списка." << "\n";
        std::cout << "3 - показать все элементы списка." << "\n";
        std::cout << "4 - прочитать данные из файла. " << "\n";
        std::cout << "5 - записать текущий список данных в файл. " << "\n";
        std::cout << "6 - отсортировать массив." << "\n";
        std::cout << "0 - завершить работу программы." << "\n";
        std::cin >> n;
    } while (n < 0 || n > 6);
}

```

```

switch (n)
{
case 0:
    list.EndEnd();
    break;
case 1:
    std::cout << "Выберите элемент какого типа вы желаете добавить.\n1 -
Элемент типа Scoop\n2 - Элемент типа ScoopFamily\n";
    std::cin >> n;
    if (n == 1)
        list.addWhithStr('K');
    if (n == 2)
        list.addWhithStr('F');
    break;
case 2:
    std::cout << "\nВыберите номер элемента, который хотите удалить: ";
    std::cin >> n;
    list.delEl(n);
    n = 20;
    break;
case 3:
    list.showAll();
    break;
case 4:
    std::cout << "\nВведите имя файла: ";
    std::cin >> str;
    list.readFile(str);
    break;
case 5:
    std::cout << "\nВведите имя файла: ";
    std::cin >> str;
    list.writeToFile(str);
    break;
case 6:
    std::cout << "\nВыберите по какому параметру вы хотите отсортировать
массив.\n1 - Id\n2 - Age\n3 - Salary\n";
    std::cin >> n;
    std::cout << "\nОтсортировать по возрастанию? (y/n)\n";
    std::cin >> str;
    if (n == 1 && str == "y")
        list.sortMass("id", list.sortAsc);
    if (n == 2 && str == "y")
        list.sortMass("age", list.sortAsc);
    if (n == 3 && str == "y")
        list.sortMass("salary", list.sortAsc);
    if (n == 1 && str == "n")
        list.sortMass("id", list.sortDesc);
    if (n == 2 && str == "n")
        list.sortMass("age", list.sortDesc);
    if (n == 3 && str == "n")
        list.sortMass("salary", list.sortDesc);
    break;
default:
    break;
}
} while (n!=0);

```

}

### Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з класами спадкоємцями та абстрактними класами.

Було розроблено програму, що використовуючи поліморфізм працює з ієрархією, яка знаходиться в одному масиві.

Поліморфізм, це властивість системи використовувати об'єкти з однаковим інтерфейсом без інформації про тип та внутрішню структуру об'єкта.

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