

МІНІСТЕРСТВО ОСВІТИ І НАУКИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХПІ»

Кафедра «Обчислювальна техніка та програмування»

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ.15А. 81 01-1 -А3

Розробники

Виконав:

студент групи КІТ-119а

\_\_\_\_\_ / Білий В. І./

Перевірив:

\_\_\_\_\_ /аспірант Бартош М. В./

Харків 2020

ЗАТВЕРДЖЕНО

КІТ.119а.

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ.119а.

Аркушів \_34\_

Харків 2020

## ЗМІСТ

Вступ.....	4
1 Поняття «Інформаційна система».....	4
1.1 Призначення та галузь застосування.....	4
1.2 Постановка завдання до розробки.....	4
2 Розробка інформаційно-довідкової системи.....	7
3 Схеми алгоритму програми.....	10
Висновок.....	14
Список джерел інформації.....	15
Додаток А. Текст програми.....	16
Додаток Б. Результати роботи програми .....	33

## ВСТУП

### Поняття «Інформаційна система»

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

### Призначення та галузь застосування

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

### Постановка завдання до розробки

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

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

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

Режим реального часу — режим обробки інформації системою при взаємодії

з зовнішніми процесами в темпі ходу цих процесів.

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

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

До переваг масиву можна віднести:

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

До недоліків:

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

До переваг списку можна віднести:

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

До недоліків:

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

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

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

## Розробка алгоритмів програми

При розробленні структур даних було створено: базовий клас Ccooperator, який наслідують класи CсоорFamily та CсоорK. На рис. 1 показано внутрішню структуру.

```
int id, age, salary;
string name;
Cchpi* rect;
Cweight weight;          int amountChild;  string mPlaceWork;
```

а)

б)

в)

Рисунок 1 – Поля базового класу (а), а також класів-спадкоємців (б, в)

Дані про підручники будуть заноситися до списку. Для цього було розроблено клас-контролер Clist з полями показаними на рис. 2 і методами на рис. 3.

```
size_t size;
Ccooperator** mass;
```

Рисунок 2 – Поля класу-контролеру

```

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;
/* ... */
void screachEl();
/* ... */
void sredSal();

```

Рисунок 3 – Розроблені методи класу

На рис. 4 подано структуру проекту розробленого програмного продукту.



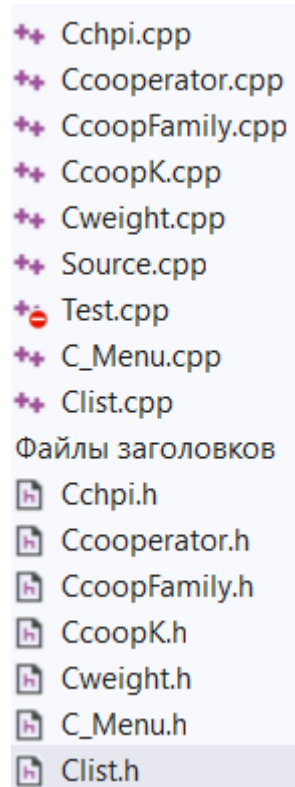


Рисунок 4 – Структура проекту  
СХЕМИ АЛГОРИТМУ ПРОГРАМИ



Рисунок 1 – Схема алгоритму методу menu класу C\_Menu

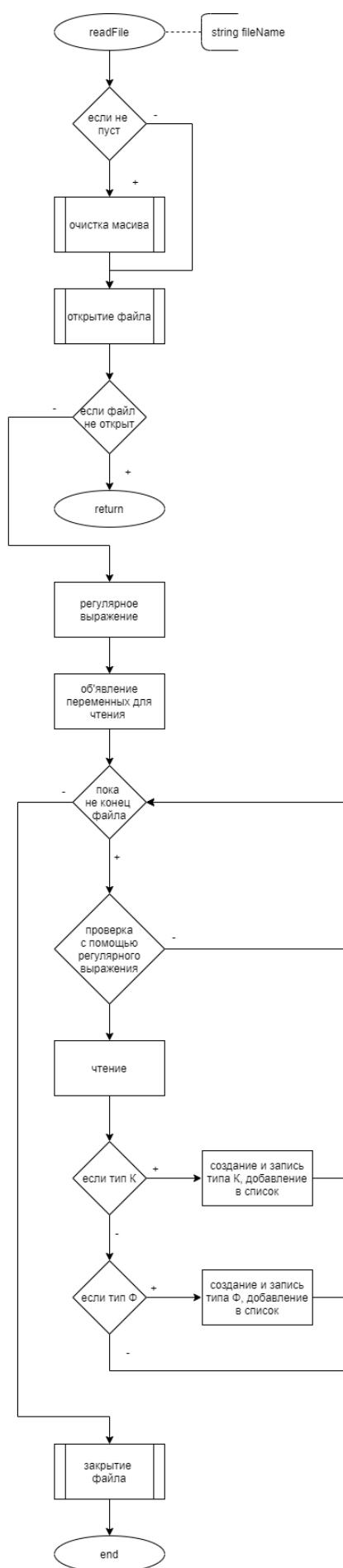


Рисунок 2 – Схема алгоритму методу читання з файлу класу-контролеру

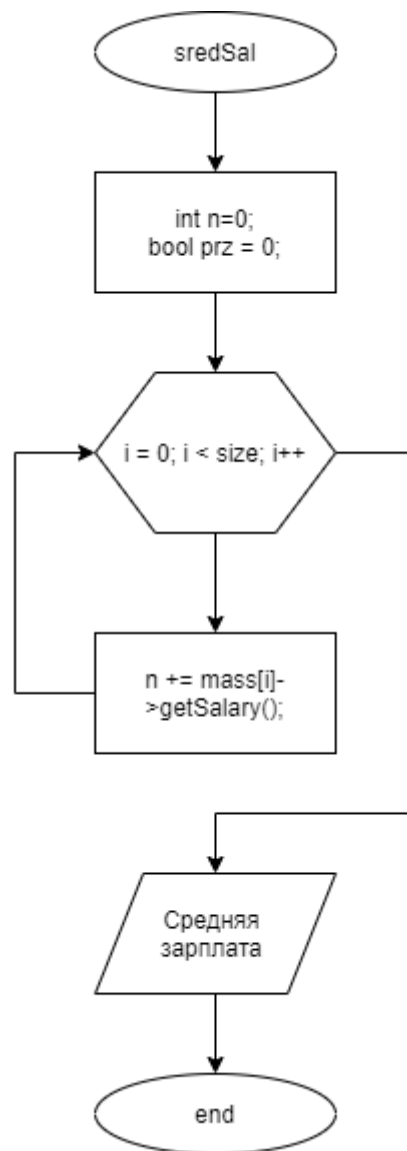


Рисунок 3 – Схема алгоритму методу запису до файлу класу-контролеру

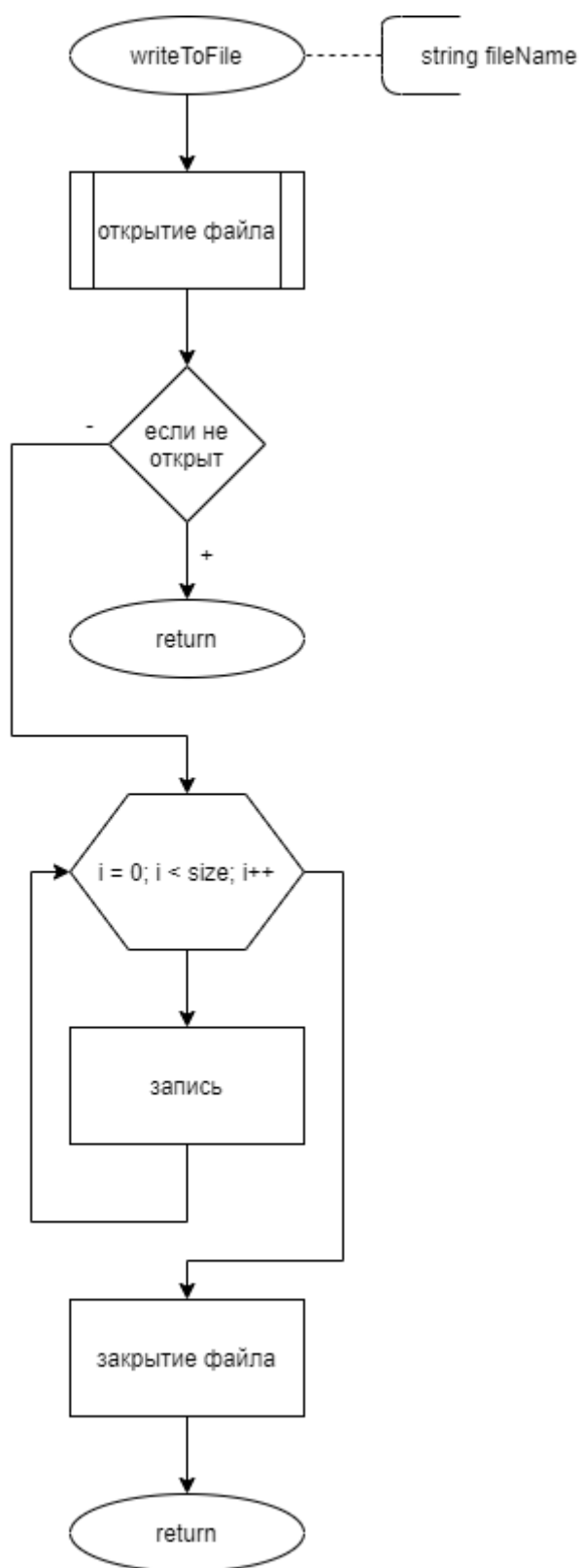


Рисунок 4 – Схема алгоритму методу task класу-контролеру

## ВИСНОВОК

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

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

## СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Дейтел Х.М. Как программировать на Си++ / Х.М. Дейтел, П.Дж. Дейтел М. : ЗАО БИНОМ, 1999. – 1000 с.
2. Штейн Клифорд (2019). Алгоритмы. Построение и анализ.
3. Вандервуд, Джосаттис - Шаблоны С++. Справочник разработчика. / Пер. с англ. - М.: Вильямс, 2008. – 536 с.
4. Андрей Александреску, Современное проектирование на С++.М.:ООО «И.Д.Вильямс», 2002.
5. Страуструп Б. Дизайн и эволюция С++ / Б. Страуструп; пер. с англ. – М. : ДМК Пресс; С.Пб: Питер, 2007. – 445 с.
6. Остерн Обобщенное программирование и STL: Использование и наращивание стандартной библиотеки шаблонов С++ / Остерн; Пер. сангл. – С.Пб: Невский Диалект, 2004. – 544 с.

## Додаток А

### Текст програми

#### Cchpi.h

```

/*!
    \brief Данный класс является полем класса Ccooperator
    Демонстрирует агрегацию

*/
#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();
};

```

#### Ccooperator.h

```

/*!
    \brief Данный класс является полем класса Ccooperator
    Демонстрирует композицию

*/
#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;
/*!
    \brief Данный класс является абстрактным
    Является основным классом проекта.
    Имеет поля "Айди", "Возраст", "Зарплата", "Имя", "Ректор", "Вес"

*/
class Ccooperator
{
protected:
    int id, age, salary;
    string name;
    Cchpi* rect;
};

```



```

        Cweight weight;
    public:
        /*!
Сеттер имени
*/
        virtual void setName(string name)final;
        /*!
Сеттер айди
*/
        virtual void setId(const int id)final;
        /*!
Сеттер возраста
*/
        virtual void setAge(const int age)final;
        /*!
Сеттер зарплаты
*/
        virtual void setSalary(const int salary)final;
        /*!
Сеттер ректора
*/
        virtual void setRect(Cchpi* rect)final;

        /*!
Сеттер веса
*/
        virtual void setWeight(Cweight w)final;
        /*!
Очищает Cchpi
*/
        virtual void delRect()final;
        /*!
Гетер айди
*/
        virtual int getId()const final;
        /*!
Гетер возраста
*/
        virtual int getAge()const final;
        /*!
Гетер зарплаты
*/
        virtual int getSalary()const final;
        /*!
Гетер имени
*/
        virtual string getName()final;
        /*!
Гетер ректора
*/
        virtual Cchpi* getRect()final;
        /*!
Гетер веса
*/
        virtual Cweight getWeight()final;
        /*!
Возвращает все данные объекта строкой
*/
        virtual string getString() = 0;
        /*!
Сеттер места работы
*/
        virtual void setmPlaceWork(const string a)= 0;
        /*!
Гетер места работы
*/

```

```

        virtual string getmPlaceWork()const = 0;
        /*!
Сеттер количества детей
*/
        virtual void setAmountChild(const int n) = 0;
        /*!
Гетер количества детей
*/
        virtual int getAmountChild() const=0;
        /*!
Конструктор по умолчанию
*/
        Ccooperator();
        /*!
Конструктор с параметрами
*/
        Ccooperator(int a, int b, int c, const string f, Cweight d, Cchpi* i);
        /*!
Конструктор копирования
*/
        Ccooperator(const Ccooperator& a);
        /*!
Деструктор
*/
        virtual ~Ccooperator() = default;
};

```

## CcoopFamily.h

```

#pragma once
#include "Ccooperator.h"
/*!
\brief Данный класс является классом наследником класса Ccooperator
Включает новое поле "Количество детей".

*/
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();
};

```

## CcoopK.h

```

#pragma once
#include "Ccooperator.h"
/*!
\brief Данный класс является наследником класса Ccooperator
Имеет дополнительное поле "Место работы"

*/
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();
};

```

## Cweight.h

```

#pragma once
#include <iostream>
/*!
    \brief Данный класс является полем класса Ccooperator
    Демонстрирует композицию
*/
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;
};

```

## C\_Menu.h

```

#pragma once
#include "Clist.h"
class C_Menu
{
public:
    void menu();
};

```

## 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&);
/*!
    \brief Данный класс является списком
    Работает с классами CcoopK и CcoopFamily

```

```

*/
class Clist
{
private:
    size_t size;
    Ccooperator** mass;
public:
    /*!
    Проверяет a>b;

    */
    static bool sortAsc(const int& a, const int& b);
    /*!
    Проверяет a>b;

    */
    static bool sortDesc(const int& a, const int& b);
    /*!
    Добавляет новый элемент
    \param[in] el Ссылка на элемент
    */
    Ccooperator** addNewEl(Ccooperator* el);
    /*!
    Удаляет элемент
    \param[in] n номер элемента
    */
    Ccooperator** delEl(size_t n);
    /*!
    Возвращает элемент
    \param[in] n номер элемента
    */
    Ccooperator* getEl(size_t n);
    /*!
    Создает элемент типа CcoopK
    */
    Ccooperator* creatElK();
    /*!
    Создает элемент типа CcoopFamily
    */
    Ccooperator* creatElF();
    /*!
    Возвращает количество элементов
    */
    int getSize();
    /*!
    Добавляет элемент в массив и заполняет его данными с клавиатуры
    \param[in] a передает тип элемента
    */
    void addWhithStr(char a);
    /*!
    Сортирует массив по заданным параметрам
    \param[in] sprz по какому полю сортировать
    \param[in] s функция сортировки
    */
    void sortMass(string sprz, Fun s);
    /*!
    Очищает все данные массива
    */
    void EndEnd();
    /*!
    Выводит на экран все элементы

    */
    void showAll();
    /*!
    Создает массив

```

```

\param[in] n размер массива
*/
    void creatMass(size_t n);
    /*!
Считывает данные с файла
\param[in] fileName имя файла
*/
    void readFile(string fileName);
    /*!
Записывает данные в файл
\param[in] fileName имя файла
*/
    void writeToFile(string fileName);
    /*!
Удаляет данные массива кроме класса Cchpi
*/
    void End();
    /*!
Конструктор по умолчанию
*/
    Clist();
    /*!
Конструктор с параметрами
*/
    Clist(int n, Ccooperator** m);
    /*!
Конструктор копирования
*/
    Clist(Clist& l);
    /*!
Деструктор
*/
    virtual ~Clist( )= default;
    /*!
Пошук элементу
*/
    void screachEl();
    /*!
Средне значения зарплаты
*/
    void sredSal();
};

```

## 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() {};

```

## Ccooperator.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";
}

```

## 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()
{
}

```

### 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.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){}

```

## 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";
}

```

## C\_Menu.cpp

```

#include "C_Menu.h"

void C_Menu::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 << "7 - поиск элемента по id." << "\n";
        std::cout << "8 - средняя зарплата." << "\n";
        std::cout << "0 - завершить работу программы." << "\n";
        std::cin >> n;
        switch (n)
        {
            case 0:
                list.EndEnd();
                break;
            case 1:

```

```

        std::cout << "Выберите элемент какого типа вы желаете
добавить.\n1 - Элемент типа CsoorK\n2 - Элемент типа CsoorFamily\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;
    case 7:
        list.screachEl();
        break;
    case 8:
        list.sredSal();
        break;
    default:
        break;
    }
} while (n != 0);

}

```

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-Z A-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}[ \\f\\n\\r\\t\\v]{1}){1}"//name
        "([A-Z A-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-Z A-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}[ \\f\\n\\r\\t\\v]{1}){1}"//nameRect
        "([A-Z A-Я a-z а-я 0-9]*[\\.\\,\\;\\:~-]{0,1}[ \\f\\n\\r\\t\\v]{1})*"
        "([A-Z A-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)
{
}

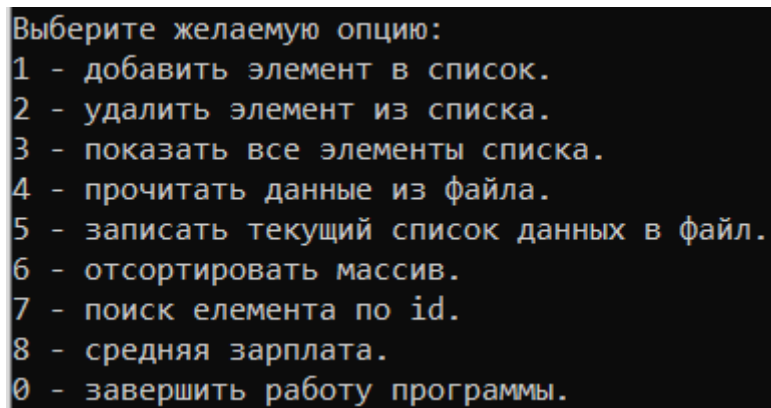
void Clist::screachEl()
{
    int n;
    bool prz=0;
    cout << endl << "Введите id элемента который ищем: ";
    cin >> n;
    cout << endl;
    for (size_t i = 0; i < size; i++) {
        if (mass[i]->getId() == n)
            prz = 1;
    }
    if (prz) {
        cout << endl << "Элемент найден" << endl;
    }
    else {
        cout << endl << "Элемент не найден" << endl;
    }
}

void Clist::sredSal()
{
    int n=0;
    bool prz = 0;
    for (size_t i = 0; i < size; i++) {
        n += mass[i]->getSalary();
    }
    cout << endl << "Средняя зарплата: " << n/size << endl;
}

```

## Додаток Б

### Результати роботи програми



```
Выберите желаемую опцию:  
1 - добавить элемент в список.  
2 - удалить элемент из списка.  
3 - показать все элементы списка.  
4 - прочитать данные из файла.  
5 - записать текущий список данных в файл.  
6 - отсортировать массив.  
7 - поиск элемента по id.  
8 - средняя зарплата.  
0 - завершить работу программы.
```

Рисунок 1 – Результаты работы програми