Автор: Білий Вадим., КІТ-119а

Дата: 26.02.2020

Лабораторна робота №2 ПЕРЕВАНТАЖЕННЯ МЕТОДІВ

Мета роботи: отримати базові знання про класи, конструктори та деструктори. Дослідити механізм створення та видалення об’єктів. Завдання до роботи

Загальне завдання.

Поширити попередню лабораторну роботу таким чином:

1) в базовому класі необхідно додати:

- мінімум одне поле типу char*;

- конструктор за замовчуванням, копіювання та конструктор з аргументами;

- деструктор;

2) у клас-список потрібно додати метод обходу масиву для виконання індивідуального завдання.

Опис класів

Базовий клас: Ccooperator

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

Опис змінних

const char* name – ім’я.

int amount - кількість елементів

Ccooperator* fEl - 1 масив

Ccooperator* fEl1 - 2 масив

int id - Id персоны

int age - вік

int salary -заробітна плата

Опис методів

void setId(const int id); - встановлює id.

void setAge(const int age);- встановлює вік.

void setSalary(const int salary); - встановлює заробітну плату.

int getId()const; - повертає id.

int getAge()const; - повертає вік.

int getSalary()const; - повертає заробітну плату.

Ccooperator(); - конструктор.

Ccooperator(int a, int b, int c, const char* d); - конструктор с параметрами.

Ccooperator(const Ccooperator& a) – конструктор копіювання.

~Ccooperator() {} – деструктор.

int averageSalary() – середня заробітна плата.

void creatMass(int a); - створює масив.

cooperator creatEl1(); - створює елемент.

cooperator creatEl2(); - створює елемент.

void Add(cooperator); - додає елемент.

void Delete(int); - видаляє елемент.

cooperator getCooperator(int a); - повертає елемент.

void showAll(); - показує всі елементи.

cooperator findCooperator(const int a); - знаходить елемент.

int getAmount(); - повертає кількість елементів.

void End(); - видаляє всі масиви.

Текст програми

Cooperator.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <clocale>
using namespace std;
class Ccooperator
{
private:
	int id, age, salary;
	const char* name;
public:
	Ccooperator();
	Ccooperator(int a, int b, int c, const char* d);
	Ccooperator(const Ccooperator& a);
	void setId(const int id);
```

```
        void setAge(const int age);
        void setSalary(const int salary);
        void setName(char*);
        const char* getName();
        int getId()const;
        int getAge()const;
        int getSalary()const;
        ~Ccooperator() {};
};
```

Cooperator.cpp

```
#include "cooperator.h"
void Ccooperator::setName(char* name) {
        this->name = name;
}
const char* 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::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), name("Ivan") {
        printf("Был вызван конструктор по умолчанию в обекте с id: %i\n", id);

}
Ccooperator::Ccooperator(const Ccooperator& a) :id(a.id), age(a.age), salary(a.salary),
name(a.name) {
printf("Был вызван конструктор по умолчанию в обекте с id: %i\n", id);
};
Ccooperator::Ccooperator(int a , int b , int c , const char* d ) :id(a), age(b),
salary(c), name(d) {
        printf("Был вызван конструктор по умолчанию в обекте с id: %i\n", id);
};
```

List.h

```
#pragma once
#include "cooperator.h"
class CList {
private:
        int amount;
        Ccooperator* fEl;
        Ccooperator* fEl1;
public:
        int averageSalary();
```

```
        void creatMass(int a);
        Ccooperator creatEl1();
        Ccooperator creatEl2();
        void Add(Ccooperator);
        void Delete(int b);
        Ccooperator getCooperator(int a);
        void showAll();
        int getAmount();
        void End();
};
```

List.cpp

```
#include "list.h"
void CList::creatMass(int a)
{
        amount = a;
        //printf("Введите количество елементов ");
        //scanf("%i", &amount);
        fEl = new Ccooperator[amount];
        //int a;
        //printf("\nВыбирите вариант создания елементов\n1. Создать елемент вручную\n2.
Готовый елемент\nВаш выбор: ");
        //scanf("%i", &a);
        //if (a == 1)
        //for (int i = 0; i < amount; i++) {
              //fEl[i] = creatEl1();
        //}
        //if (a == 2)
              for (int i = 0; i < amount; i++) {
                    fEl[i] = creatEl2();
              }
}
Ccooperator CList::creatEl1() {
        Ccooperator El;
        int a;
        printf("Введите id сотрудника: ");
        scanf("%i", &a);
        El.setId(a);
        printf("Введите зарплату сотрудника: ");
        scanf("%i", &a);
        El.setSalary(a);
        printf("Введите возраст сотрудника: ");
        scanf("%i", &a);
        El.setAge(a);
        return El;
}
int CList::averageSalary() {
        int averageSalary=  0;
        for (int i = 0; i < amount; i++)
        {
              averageSalary = averageSalary + fEl[i].getSalary();
        }
        return averageSalary = averageSalary / amount;

};
Ccooperator CList::creatEl2() {
        Ccooperator El;
        El.setId(0);
        El.setSalary(0);
        El.setAge(0);
        return El;
}
```

```
void CList::Add(Ccooperator El1) {
	fEl1 = new Ccooperator[amount + 1];
	for (int i = 0; i < amount; i++) {
		fEl1[i] = fEl[i];
	}
	fEl1[amount] = El1;
	delete[] fEl;
	amount++;
	fEl = new Ccooperator[amount];
	for (int i = 0; i < amount; i++) {
		fEl[i] = fEl1[i];
	}
	delete[] fEl1;
}
int CList::getAmount() {
	return amount;
}
void CList::Delete(int a) {
	Ccooperator* fEl1 = new Ccooperator[amount - 1];
	for (int i = 0; i < a - 1; i++) {
		fEl1[i] = fEl[i];
	}
	for (int i = a - 1, j = a; j < amount; i++, j++) {
		fEl1[i] = fEl[j];
	}
	delete[] fEl;
	amount--;
	fEl = new Ccooperator[amount];
	for (int i = 0; i < amount; i++) {
		fEl[i] = fEl1[i];
	}
	delete[] fEl1;
}
Ccooperator CList::getCooperator(const int a) {
	return fEl[a];
}
void CList::showAll() {
	for (int i = 0; i < amount; i++) {
		printf("ID: %i\n Age: %i\n Salary: %i\n Name: %s\n",
getCooperator(i).getId(), getCooperator(i).getAge(), getCooperator(i).getSalary(),
getCooperator(i).getName());
	}
}
void CList::End() {
	delete[] fEl;
}
```

Source.cpp

```
#include "cooperator.h"
#include "list.h"
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <clocale>
using namespace std;
int main() {
	setlocale(LC_ALL, "rus");

	CList List1;
	Ccooperator Obj2;
	Ccooperator Obj3;
	Ccooperator Obj4;
	Ccooperator Obj5;
	printf("ID: %i\n Age: %i\n Salary: %i\n Name: %s\n", Obj2.getId(), Obj2.getAge(),
Obj2.getSalary(), Obj2.getName());
```

```
	Obj3.setId(1);
	Obj3.setAge(19);
	Obj3.setSalary(1000);
	Obj4.setId(2);
	Obj4.setAge(28);
	Obj4.setSalary(500);
	Obj5.setId(3);
	Obj5.setAge(24);
	Obj5.setSalary(2000);
	List1.creatMass(0);
	List1.Add(Obj3);
	List1.Add(Obj4);
	List1.Add(Obj5);
	List1.showAll();
	printf("Средняя зарплата %i\n", List1.averageSalary());
	List1.End();
	Ccooperator testCop = Obj5;
	printf("ID: %i\n Age: %i\n Salary: %i\n Name: %s\n", testCop.getId(),
testCop.getAge(), testCop.getSalary(), testCop.getName());
	if (_CrtDumpMemoryLeaks())
		printf("\nMemory leack deteckted\n");
	else
		printf("\nMemory is not leack deteckted\n");
}
```

Test.cpp

```
#include "cooperator.h"
#include "list.h"
#define _CRT_SECURE_NO_WARNINGS
#define N 5
#include <iostream>
#include <clocale>
using namespace std;
int main() {
	setlocale(LC_ALL, "rus");
	Ccooperator a;
	a.setAge(0);
	a.setAge(0);
	a.setSalary(0);
	CList a1[N];
	int test[N];
	int rezult1[N];
	int rezult2[N];
	int rezult3[N];
	test[0] = 1;
	test[1] = 5;
	test[2] = 10;
	test[3] = 25;
	test[4] = 50;
	rezult1[0] = 1;
	rezult1[1] = 5;
	rezult1[2] = 10;
	rezult1[3] = 25;
	rezult1[4] = 50;
	rezult2[0] = 2;
	rezult2[1] = 6;
	rezult2[2] = 11;
	rezult2[3] = 26;
	rezult2[4] = 51;
	rezult3[0] = 1;
	rezult3[1] = 5;
	rezult3[2] = 10;
```

```
rezult3[3] = 25;
rezult3[4] = 50;
for (int i = 0; i < N; i++) {
	a1[i].creatMass(test[i]);
	if (a1[i].getAmount() == rezult1[i]) {
		printf("Тест 1.%i пройден\n", i);
	}
	else {
		printf("Тест 1.%i не пройден\n", i);
	}
}
for (int i = 0; i < N; i++) {
	a1[i].Add(a);
	if (a1[i].getAmount() == rezult2[i]) {
		printf("Тест 2.%i пройден\n", i);
	}
	else {
		printf("Тест 2.%i не пройден\n", i);
	}
}
for (int i = 0; i < N; i++) {
	a1[i].Delete(test[i]);
	if (a1[i].getAmount() == rezult3[i]) {
		printf("Тест 3.%i пройден\n", i);
	}
	else {
		printf("Тест 3.%i не пройден\n", i);
	}
}
Ccooperator Obtest1, Obtest2;
Obtest1.setSalary(-200);
Obtest2.setSalary(300);
CList TestList;
TestList.creatMass(0);
TestList.Add(Obtest1);
TestList.Add(Obtest2);
if (TestList.averageSalary() == 50) {
	printf("Тест 4 пройден\n");
}
else {
	printf("Тест 4 не пройден\n");
	}
}
```

Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з конструкторами та деструкторами.

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