Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №1

За 3 семестр

По дисциплине «Языки программирования»

Тема: «Классы и объекты в С++»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Хацкевич М.В.

Брест 2020

**Лабораторная работа №1**

**Классы и объекты в С++**

**Цель:** получить практические навыки реализации классов на С++.

**Постановка задачи:**

1. Определить пользовательский класс в соответствии с вариантом задания.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных.
5. Определить указатель на компоненту-функцию.
6. Определить указатель на экземпляр класса.
7. Написать демонстрационную программу, в которой создаются и разрушаются объекты пользовательского класса и каждый вызов сопровождается выдачей соответствующего сообщения (какой объект какой конструктор или деструктор вызвал).
8. Показать в программе использование указателя на объект и указателя на компоненту-функцию.

Методические указания.

1. Пример определения класса.

const int LNAME=25;

class STUDENT{

char name[LNAME]; // имя

int age; // возраст

float grade; // рейтинг

public:

STUDENT(); // конструктор без параметров

STUDENT(char\*,int,float); // конструктор с параметрами

STUDENT(const STUDENT&); // конструктор копирования

~STUDENT();

char \* GetName() ;

int GetAge() const;

float GetGrade() const;

void SetName(char\*);

void SetAge(int);

void SetGrade(float);

void Set(char\*,int,float);

void Show(); };

Более профессионально определение поля name типа указатель:

char\* name. Однако в этом случае реализация компонентов-функций ус-

ложняется.

2. Пример реализации конструктора с выдачей сообщения.

STUDENT::STUDENT(char\*NAME,int AGE,float GRADE)

{

strcpy(name,NAME); age=AGE; grade=GRADE;

cout<< \nКонструктор с параметрами вызван для объекта

<<this<<endl;

}

3. Следует предусмотреть в программе все возможные способы вы-

зова конструктора копирования. Напоминаем, что конструктор копирова-

ния вызывается:

а) при использовании объекта для инициализации другого объекта

Пример.

STUDENT a(“Иванов”,19,50), b=a;

б) когда объект передается функции по значению

Пример.

void View(STUDENT a){a.Show;}

в) при построении временного объекта как возвращаемого значения

функции

Пример.

STUDENT NoName(STUDENT & student)

{STUDENT temp(student);

temp.SetName(“NoName”);

return temp;}

STUDENT c=NoName(a);

4. В программе необходимо предусмотреть размещение объектов как

в статической, так и в динамической памяти, а также создание массивов

объектов.

Примеры.

а) массив студентов размещается в статической памяти

STUDENT gruppa[3];

gruppa[0].Set(“Иванов”,19,50);

и т.д.

или

STUDENT gruppa[3]={STUDENT(“Иванов”,19,50),

STUDENT(“Петрова”,18,25.5),

STUDENT(“Сидоров”,18,45.5)};

б) массив студентов размещается в динамической памяти

STUDENT \*p;

p=new STUDENT [3];

p-> Set(“Иванов”,19,50);

и т.д.

5. Пример использования указателя на компонентную функцию

void (STUDENT::\*pf)();

pf=&STUDENT::Show;

(p[1].\*pf)();

6. Программа использует три файла:

• заголовочный h-файл с определением класса,

• cpp-файл с реализацией класса,

• сpp-файл демонстрационной программы.

Для предотвращения многократного включения файла-заголовка

следует использовать директивы препроцессора

#ifndef STUDENTH

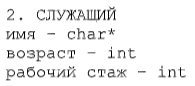
#define STUDENTH

// модуль STUDENT.H

...

#endif

**Вариант 2**



**Ход работы:**

**Определение пользовательского класса:**

class Employee { //Класс Служащий

private:

char\* e\_name; //Имя

int e\_age, e\_experience; //Возраст и Стаж

public:

Employee(); //Конструктор без параметров

Employee(const char\* name, int age, int experience); // Конструктор с параметрами

Employee(const Employee& data); //Конструктор копирования

~Employee(); //Деструктор

char\* get\_name(); //Ввод имени

int get\_age(); //Ввод возраста

int get\_experience(); //Ввод стажа

void set\_name(); //Присваивание имени

void set\_age(); //Присваивание возраста

void set\_experience(); //Присваивание стажа

void set(); //Вызов функций присваивания полей

void show(); //Вывод информации

};

**Реализация конструктора и деструктора:**

Employee::Employee() :

e\_age(0), e\_experience(0)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, "\0");

cout << "\nКонструктор без параметров вызван для объекта " << this << endl;

}

Employee::Employee(const char\* name, int age, int experience) :

e\_age(age), e\_experience(experience)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, name);

cout << "\nКонструктор с параметрами вызван для объекта " << this << endl;

}

Employee::Employee(const Employee& data):

e\_age(data.e\_age), e\_experience(data.e\_experience)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, data.e\_name);

cout << "\nКонструктор копирования вызван для объекта " << this << endl;

}

Employee::~Employee()

{

cout << "\nДеструктор вызван для объекта " << this << endl;

}

**Использование указателя на объект:**

Employee\* ptr; //Указатель на объект

ptr = &e5[0]; //Присваивание указателю адреса другого объекта

ptr->show(); //Вывод значений полей объекта

**Использование указателя на функцию:**

void(Employee:: \* pf)(); //Указатель на компонентную функцию

pf = &Employee::show; //Присваивание адреса

(e5[2].\*pf)(); //Вывод значений полей объекта

**Листинг основной программы:**

**Модуль Employee.h**

#ifndef EMPLOYEEH

#define EMPLOYEEH

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>

#define NAME\_SIZE 20 //Размер имени

using namespace std;

class Employee { //Класс Служащий

private:

char\* e\_name; //Имя

int e\_age, e\_experience; //Возраст и Стаж

public:

Employee(); //Конструктор без параметров

Employee(const char\* name, int age, int experience); // Конструктор с параметрами

Employee(const Employee& data); //Конструктор копирования

~Employee(); //Деструктор

char\* get\_name(); //Ввод имени

int get\_age(); //Ввод возраста

int get\_experience(); //Ввод стажа

void set\_name(); //Присваивание имени

void set\_age(); //Присваивание возраста

void set\_experience(); //Присваивание стажа

void set(); //Вызов функций присваивания полей

void show(); //Вывод информации

};

#endif

**Модуль Employee.cpp**

#include "Employee.h"

Employee::Employee() :

e\_age(0), e\_experience(0)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, "\0");

cout << "\nКонструктор без параметров вызван для объекта " << this << endl;

}

Employee::Employee(const char\* name, int age, int experience) :

e\_age(age), e\_experience(experience)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, name);

cout << "\nКонструктор с параметрами вызван для объекта " << this << endl;

}

Employee::Employee(const Employee& data):

e\_age(data.e\_age), e\_experience(data.e\_experience)

{

e\_name = new char[NAME\_SIZE];

strcpy(e\_name, data.e\_name);

cout << "\nКонструктор копирования вызван для объекта " << this << endl;

}

Employee::~Employee()

{

cout << "\nДеструктор вызван для объекта " << this << endl;

}

void Employee::set()

{

set\_name();

set\_age();

set\_experience();

}

void Employee::show()

{

cout << "\nИмя - " << e\_name << endl

<< "Возраст - " << e\_age << endl

<< "Рабочий стаж - " << e\_experience << endl;

}

char\* Employee::get\_name()

{

char\* get = new char [NAME\_SIZE];

cin.get();

cout << "\nВведите имя - ";

cin.getline(get, NAME\_SIZE);

return get;

}

int Employee::get\_age()

{

int get;

cout << "Введите возраст - ";

cin >> get;

while (get > 150 || get < 0) {

cout << "Некорректные данные. Повторите ввод.\n";

cout << "Введите возраст - ";

cin >> get;

}

return get;

}

int Employee::get\_experience()

{

int get;

cout << "Введите стаж - ";

cin >> get;

while (get > e\_age || get < 0) {

cout << "Некорректные данные. Повторите ввод\n";

cout << "Введите стаж - ";

cin >> get;

}

return get;

}

void Employee::set\_name()

{

strcpy(e\_name, get\_name());

}

void Employee::set\_age()

{

e\_age = get\_age();

}

void Employee::set\_experience()

{

e\_experience = get\_experience();

}

**Модуль main.cpp**

#include <iostream>

#include <Windows.h>

#include "Employee.h"

using namespace std;

Employee cpy(Employee& data) //Конструктор копирования при построении временного объекта как возвращаемого значения функции

{

Employee temp(data);

temp.set\_name();

return temp; //Вызов деструктора для объекта temp

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

//Статистический массив объектов, конструктор без параметров

Employee e1[3];

e1[0].show();

//Конструктор с параметрами и копирования

Employee e2("Иван", 32, 5), e3=e2;

e2.show();

e3.show();

//Конструктор копирования (передача объекта по значению)

Employee e4(e3);

e4.show();

//Динамический массив объектов

Employee\* e5 = new Employee[3];

for (int i = 0; i < 3; i++)

e5[i].set();

for (int i = 0; i < 3; i++)

e5[i].show();

Employee\* ptr; //Указатель на объект

ptr = &e5[0]; //Присваивание указателю адреса другого объекта

ptr->show(); //Вывод значений полей объекта

//Конструктор копирования при построении временного объекта как возвращаемого значения функции

Employee e6 = cpy(e5[1]);

e6.show();

void(Employee:: \* pf)(); //Указатель на компонентную функцию

pf = &Employee::show; //Присваивание адреса

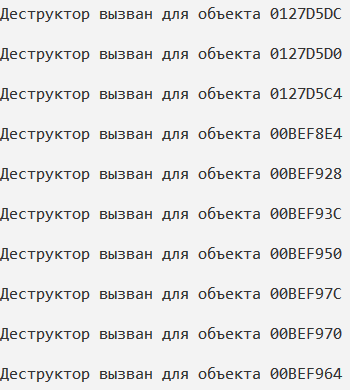
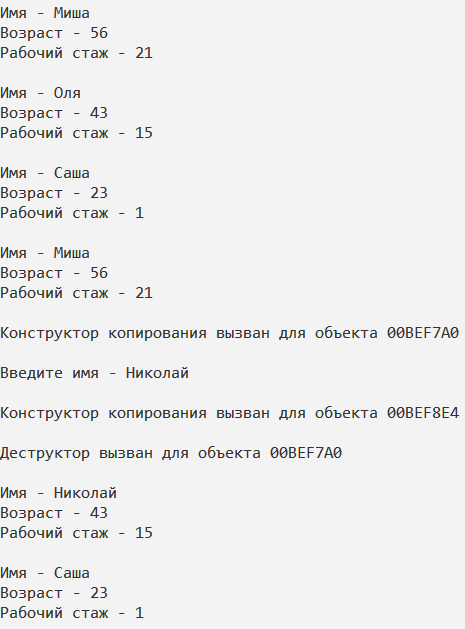
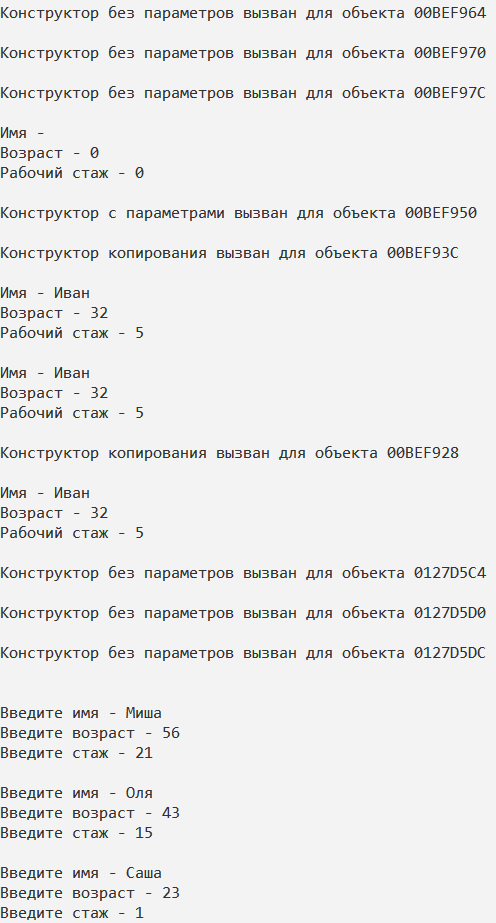
(e5[2].\*pf)(); //Вывод значений полей объекта

delete[] e5; //Вызов деструктора для массива объектов e5

return 0; //Вызов деструктора для объектов e6, e4, e3, e2, массива объектов e1

}

**Пример работы программы:**



**Вывод:** В ходе лабораторной работы получил практические навыки реализации классов на С++. Написал программу, в которой создаются и разрушаются объекты, определенного пользователем класса. Выполнил исследование вызовов конструкторов и деструкторов.