Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Лабораторная работа №2

“Объекты и Классы”

По дисциплине “Объектно-ориентированное программирование”

Выполнил:

Студент группы ПРО-232Б

Степанов К.А.

Проверил:

доцент кафедры ВМиК

Котельников В. А.

Уфа-2023

**Цель работы:**

Определение и реализация тестовых классов, и написание программы, иллюстрирующей их использование.

**Задание:**

* Определения
  + классов
  + атрибутов, методов
  + классов-наследников
  + конструкторов (обязательно несколько: без параметров, с параметрами, с параметром-объектом того же класса), деструкторов
  + доступности атрибутов и методов (показать, как влияют модификаторы private, protected, public)
* Реализация
  + методов объектов
  + реализация методов сразу в определении или после определения
  + конструкторов, деструкторов
* Создание и использование (обращение к атрибутам, вызов методов)
  + статически создаваемых объектов («MyClass obj;»)
  + динамически создаваемых объектов («MyClass \*obj = new MyClass();»)
  + объектов с помощью различных конструкторов (**у каждого** создаваемого объекта должны быть: конструктор без параметров, с параметрами, с параметром-объектом того же класса – конструктор копирования)
  + инициализация полей класса в списке инициализации конструктора
  + помещение объектов в переменные различных типов (объяснять, чем отличается MyBase \* obj = new MyBase() от MyBase \* obj = new MyDeriv())
  + объектов классов-наследников (проверить и продемонстрировать, какие конструкторы классов при этом вызываются)
  + композиция объектов: атрибутом одного объекта класса A является другой объект класса B (не указатель!), инициализируемый в списке инициализации конструктора класса A; показать, когда этот объект класса B удаляется
  + композиция объектов: атрибутом одного объекта класса A является указатель на другой объект класса B, создаваемый в конструкторе класса A и уничтожаемый в деструкторе класса A; композируемые классы должны определяться отдельно (не один в другом); при композиции показать, в чем разница, если объект класса А хранит прямо объект класса В или указатель на объект класса В.
* Уничтожение
  + статически созданных объектов
  + динамически созданных объектов
  + объектов классов-наследников (проверить и продемонстрировать, какие деструкторы классов при этом вызываются)

**Ход работы:**

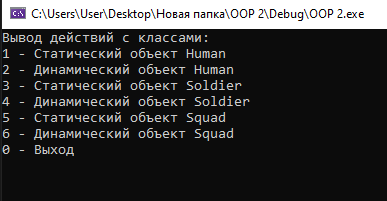


Рис. 1. Выбор действий

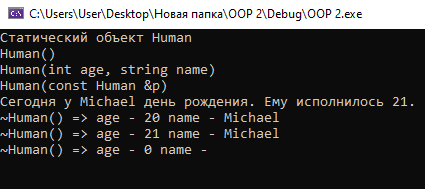


Рис. 2. Статический объект Human

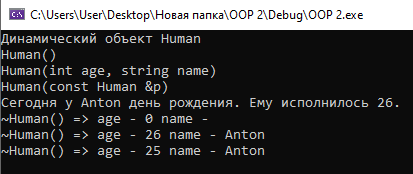


Рис. 3. Динамический объект Human

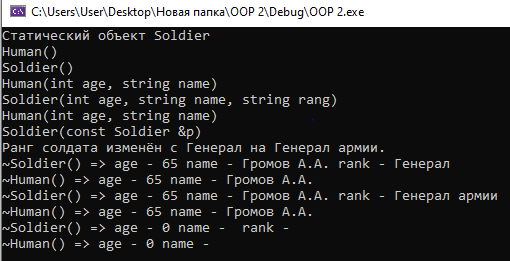


Рис. 4. Статический объект Soldier

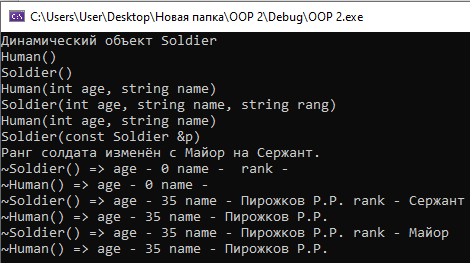


Рис. 5. Динамический объект Soldier

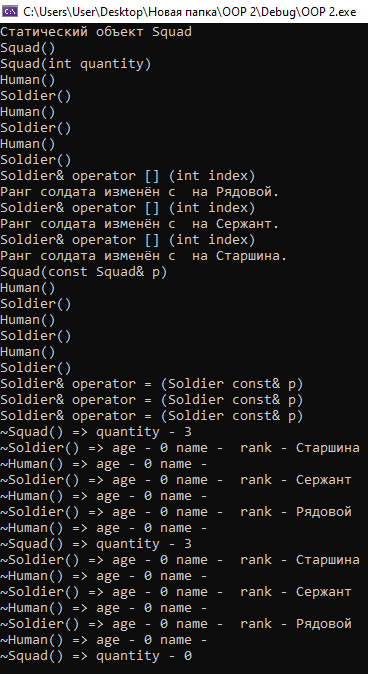


Рис. 6. Статический объект Squad

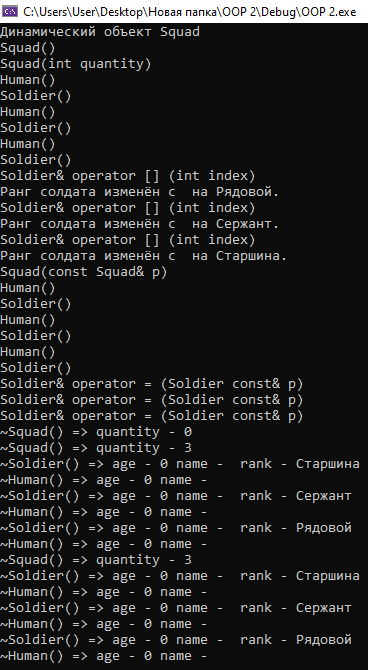


Рис. 7. Динамический объект Squad

**Вывод:**

В результате выполнения работы было освоено создание классов, создание дочерних классов, композиция, реализация методов внутри и вне описания класса, изменение атрибутов доступа через поля класса, а также удаление объектов класса.

Ссылка на GitHub: <https://github.com/stkirill15/OOP-2>.

**Приложение 1**

**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <conio.h>

using namespace std;

class Human { // Класс-предок

public:

int age;

string name;

Human() : age(0), name("") {

printf("Human()\n");

}

Human(int age, string name) {

printf("Human(int age, string name)\n");

this->age = age;

this->name = name;

}

Human(const Human &p) {

printf("Human(const Human &p)\n");

age = p.age;

name = p.name;

}

~Human() {

printf("~Human() => ");

cout << "age - " << age << " name - " << name << endl;

}

void birthday() {

age++;

cout << "Сегодня у " << name << " день рождения. Ему исполнилось " << age << ".\n";

}

};

class Soldier : Human { // Класс-наследник

public:

string rank;

Soldier() : rank("") {

printf("Soldier()\n");

}

Soldier(int age, string name, string rank) : Human(age, name) {

printf("Soldier(int age, string name, string rang)\n");

this->rank = rank;

}

Soldier(const Soldier& p) :Human(p.age, p.name) {

printf("Soldier(const Soldier &p)\n");

rank = p.rank;

}

Soldier& operator = (Soldier const& p) {

printf("Soldier& operator = (Soldier const& p)\n");

age = p.age;

name = p.name;

rank = p.rank;

return \*this;

}

~Soldier() {

printf("~Soldier() => ");

cout << "age - " << age << " name - " << name << " rank - " << rank << endl;

}

void change\_rang(string rank) {

cout << "Ранг солдата изменён с " << this->rank << " на " << rank << ".\n";

this->rank = rank;

}

};

class Squad { // Класс с указателем на класс-наследник

public:

Soldier\* soldiers; // массив класса "солдат"

int quantity; // количество солдат

Squad() {

printf("Squad()\n");

quantity = 0;

soldiers = nullptr;

}

Squad(int quantity) {

printf("Squad(int quantity)\n");

this->quantity = quantity;

soldiers = new Soldier[quantity];

}

Squad(const Squad& p) {

printf("Squad(const Squad& p)\n");

quantity = p.quantity;

soldiers = new Soldier[quantity];

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

soldiers[i] = p.soldiers[i];

}

~Squad() {

printf("~Squad() => ");

cout << "quantity - " << quantity << endl;

delete[]soldiers;

}

Soldier& operator [] (int index) {

printf("Soldier& operator [] (int index)\n");

return soldiers[index];

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

while (1) {

system("cls");

printf("Вывод действий с классами:\n1 - Статический объект Human\n2 - Динамический объект Human\n3 - Статический объект Soldier\n4 - Динамический объект Soldier\n5 - Статический объект Squad\n6 - Динамический объект Squad\n0 - Выход");

char choice = \_getch();

system("cls");

if (choice == '1') { // Статический объект Human

printf("Статический объект Human\n");

Human h1; // пустой объект

Human h2(20, "Michael"); // пример со значениями

Human h3 = h2; // пример с конструктором копирования

h2.birthday();

}

else if (choice == '2') { // Динамический объект Human

printf("Динамический объект Human\n");

Human\* h1 = new Human(); // пустой объект

Human\* h2 = new Human(25, "Anton"); // объект со значениями

Human\* h3 = new Human(\*h2); // пример с конструктором копирования

h2->birthday();

delete h1;

delete h2;

delete h3;

}

else if (choice == '3') { // Статический объект Soldier

printf("Статический объект Soldier\n");

Soldier s1; // пустой объект

Soldier s2(65, "Громов А.А.", "Генерал"); // объект со значениями

Soldier s3 = s2; // пример с конструктором копирования

s2.change\_rang("Генерал армии");

}

else if (choice == '4') { // Динамический объект Soldier

printf("Динамический объект Soldier\n");

Soldier\* s1 = new Soldier();

Soldier\* s2 = new Soldier(35, "Пирожков Р.Р.", "Майор");

Soldier\* s3 = new Soldier(\*s2);

s2->change\_rang("Сержант");

delete s1;

delete s2;

delete s3;

}

else if (choice == '5') {

printf("Статический объект Squad\n");

Squad s1;

Squad s2(3);

s2[0].change\_rang("Рядовой");

s2[1].change\_rang("Сержант");

s2[2].change\_rang("Старшина");

Squad s3 = s2;

}

else if (choice == '6') {

printf("Динамический объект Squad\n");

Squad \*s1 = new Squad();

Squad \*s2 = new Squad(3);

(\*s2)[0].change\_rang("Рядовой");

(\*s2)[1].change\_rang("Сержант");

(\*s2)[2].change\_rang("Старшина");

Squad \*s3 = new Squad (\*s2);

delete s1;

delete s2;

delete s3;

}

else if (choice == '0')

break;

\_getch();

}

}