Лабораторная работа №11

*«Программирование с использованием различных свойств наследования»*

**Цель работы:** сформировать знания и умения по использованию принципа наследования в ЯП Си++, приобрести практические навыки создания программ на языке Си++ с использованием иерархии классов.

**Задание 2.** Откомпилировать и выполнить примеры программ, приведенных в теоретическом материале.



Рисунок 1 – Результат выполнения программы

**Задание 3.** Создать иерархию классов для работы с клиентами:

– Базовый класс почтовый адрес. Свойства: город, улица, номер дома, номер квартиры/офиса.

Методы: ввод, редактирование, вывод.

– Два производных класса:

Физическое лицо. Свойства: фамилия, имя, отчество, паспортные данные.

Методы: ввод, редактирование, вывод.

Юридическое лицо: форма собственности, название, расчетный счет, УНН.

Методы: ввод, редактирование, вывод.

**Решение:**

#include <iostream>

#include <string>

using namespace std;

class pochAdress {

protected:

string city;

string street;

unsigned int num;

unsigned int kv;

public:

pochAdress(string c, string s, unsigned int n, unsigned int k)

{

city = c;

street = s;

num = n;

kv = k;

}

~pochAdress() {}

void input()

{

cout << "Введите город: "; cin >> city;

cout << "Введите улицу: "; cin >> street;

cout << "Введите номер дома: "; cin >> num;

cout << "Введите номер квартиры: "; cin >> kv;

}

void edit();

void output()

{

cout << endl << city << " " << street << " " << num << " " << kv << endl << endl;

}

};

class fPers : public pochAdress {

string surname;

string name;

string otchestvo;

string ID;

public:

fPers(string sr, string nm, string o, string i, string c, string s, unsigned int n, unsigned int k) : pochAdress(c, s, n, k)

{

surname = sr;

name = nm;

otchestvo = o;

ID = i;

}

~fPers() {}

void input()

{

cout << "Введите фамилию: "; cin >> surname;

cout << "Введите имя: "; cin >> name;

cout << "Введите отчество: "; cin >> otchestvo;

cout << "Введите номер паспорта: "; cin >> ID;

}

void edit();

void output()

{

cout << endl << city << " " << street << " " << num << " " << kv << " " << surname << " " << name << " " << otchestvo << " " << ID << endl << endl;

}

};

class uPers : public pochAdress {

string form;

string names;

unsigned long int rasScht;

string UNN;

public:

uPers(string f, string ne, unsigned long int rs, string unn, string c, string s, unsigned int n, unsigned int k) : pochAdress(c, s, n, k)

{

form = f;

names = ne;

rasScht = rs;

UNN = unn;

}

~uPers() {}

void input()

{

cout << "Введите форму собственности: "; cin >> form;

cout << "Введите название: "; cin >> names;

cout << "Введите номер расчетного счета: "; cin >> rasScht;

cout << "Введите номер УНН: "; cin >> UNN;

}

void edit();

void output()

{

cout << endl << city << " " << street << " " << num << " " << kv << " " << form << " " << names << " " << rasScht << " " << UNN << endl << endl;

}

};

int main()

{

setlocale(LC\_ALL, "Rus");

pochAdress ad("aa", "asd", 4, 5);

ad.input();

ad.output();

fPers fiz("Gabets", "Staniclav", "Olegovich", "0124949GG030HG1", "Minsk", "Mayakovskogo", 45, 22);

fiz.input();

fiz.output();

uPers ured("Предприятие", "ООО Белгазпром", 123467990, "TYUUFGEHV124GHHJF", "Minsk", "Mayakovskogo", 123, 64);

ured.input();

ured.output();

return 0;

}

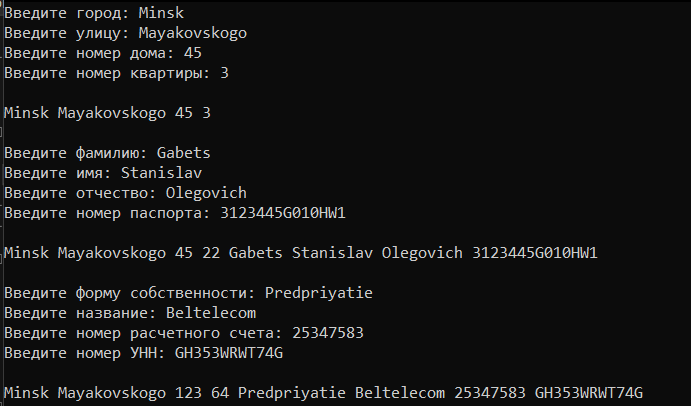


Рисунок 2 – Результат выполнения программы

**Контрольные вопросы:**

1. **Что подразумевает под собой механизм наследования?**

Наследование — это механизм создания нового класса на основе уже существующего. При этом к существующему классу могут быть добавлены новые элементы (данные и функции), либо существующие функции могут быть изменены.

1. **Как и на что влияют режимы доступа при наследовании?**

Права доступа при наследовании ведут себя по-разному в зависимости от типа наследования.

Для спецификации доступа в С++ есть три уровня: private, protected и public.

Само наследование также может быть private, protected и public.

По умолчанию классы наследуются, как private, а структуры - как public.

Важно понимать, что независимо от типа наследования, ни один из наследников не сможет получить доступ к private-данным базового класса.

private наследование

Данные protected и public базового класса доступны из методов производного класса, но недоступны извне, то есть они становятся private.

protected наследование

Данные, которые в CBase были protected и public становятся protected.

Применение protected наследования в реальном коде практически не встречается, однако он встречается в тестах и на собеседованиях.

public наследование

Публичное наследование наиболее употребительно. При этом protected и public данные из базового класса остаются, соответственно protected и public в производном классе.

1. **Как передать параметры конструктора базового класса в конструктор производного класса?**

При вызове конструктора сначала вызывается конструктор базового класса, а потом выполняется конструктор производного класса. Деструкторы выполняются в обратном порядке. Если необходимо передать параметры из конструктора базового класса в конструктор класса потомка, то используется конструкция:

констр\_произв\_класса(пар-ры): констр\_базового\_класса(пар-ры)

{

тело конструктора

}

1. **Что произойдет если в базовом и производном классах совпадут имена методов?**

Метод будет переопределен для этого класса.