ПРАКТИЧЕСКАЯ РАБОТА №7
АБСТРАКТНЫЕ КЛАССЫ.
МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ

ЦЕЛЬ ПРАКТИЧЕСКОЙРАБОТЫ:

Целью данной практической работы является знакомство с абстрактными классами и механизмом виртуальных функций, а также множественным наследованием в языке программирования C++.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Механизм абстрактных классов служит для представления общих понятий, которые фактически используются лишь для порождения более конкретных понятий. Абстрактный класс можно также употреблять как определение интерфейса, в котором производные классы обеспечивают разнообразие реализаций.

Абстрактный класс – это класс, который может использоваться лишь в качестве базового класса для некоторого другого класса. Класс является абстрактным, если он содержит хотя бы одну чистую виртуальную функцию.

Виртуальная функция называется чистой, если в объявлении функции внутри объявления класса задан чистый спецификатор = 0.

```
1) class Shape
2) { public:
3)     virtual void draw() = 0;
4) //чисто виртуальная функция
5) ...
6) };
```

Абстрактный класс нельзя употреблять в качестве типа объектов, типа параметров функций, типа возвращаемого функцией значения или как тип при явном приведении типа. Можно, однако, объявлять указатели и ссылки на абстрактный класс.

```
1) Shape  s;          // Ошибка: объект абстрактного класса
2) Shape *s;          // Всё правильно
3) Shape  f();        // Ошибка
4) void   f(Shape s);  // Ошибка
5) Shape& f(Shape &s); // Всё правильно
```

Чистые виртуальные функции наследуются и остаются чистыми виртуальными функциями, таким образом, производный класс, в котором чистая виртуальная функция не переопределена, остаётся абстрактным классом.

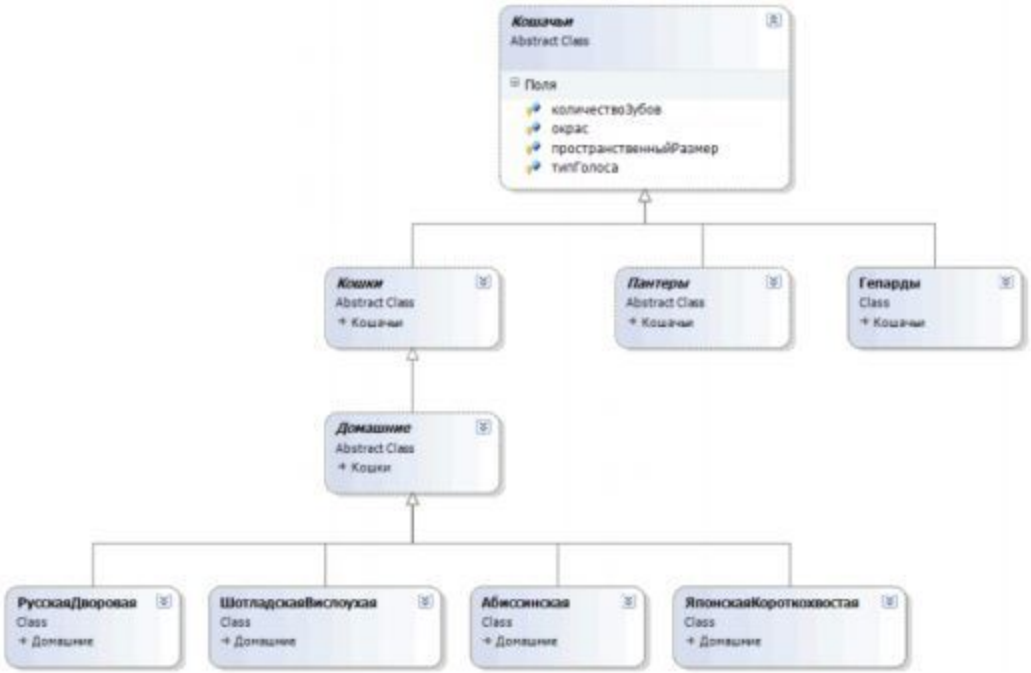

Рис. 1 «Пример наследования от абстрактных классов»

Множественное наследование

Один класс может наследовать атрибуты двух и более классов одновременно. Для этого используется список базовых классов, в котором каждый из базовых классов отделен от других запятой. Общая форма множественного наследования имеет вид:

```
class имя_порожденного_класса: список базовых классов
{
…
};
```

В следующем примере класс Z наследует оба класса X и Y:

```
#include <iostream.h>
class X {
protected:
int a;
public:
void make_a(int i) { a = i; }
};
class Y {
protected:
 int b;
 public:
 void make_b(int i) { b = i; }
 };
 // Z наследует как от X, так и от Y
 class Z: public X, public Y {
 public:
 int make_ab() { return a*b; }
 };
 int main()
 {
 Z i;
 i.make_a(10);
 i.make_b(12);
 cout << i .make_ab();
 return 0;
 }
```

Поскольку класс Z наследует оба класса X и Y, то он имеет доступ к публичным и защищенным членам обоих классов X и Y.

В предыдущем примере ни один из классов не содержал конструкторов. Однако ситуация становится более сложной, когда базовый класс содержит конструктор. Например, изменим предыдущий пример таким образом, чтобы классы X, Y и Z содержали конструкторы:

```
1)#include <iostream.h>
2)class X {
3)protected:
4)int a;
5)public:
6)X() {
7)a = 10;
8)cout << "Initializing X\n";
9)}
10) };
11) class Y {
12) protected:
13) int b;
14) public:
15) Y() {
16) cout << "Initializing Y\n";
17) b = 20;
18) }
19) };
20) // Z наследует как от X, так и от Y
21) class Z: public X, public Y {
22) public:
23) Z() { cout << "Initializing Z\n"; }
24) int make_ab() { return a*b; }
25) };
26) int main()
27) {
28) Z i;
29) cout << i.make_ab();
30) return 0;
31) }
```

Программа выдаст на экран следующий результат:

```
Initializing X
Initializing Y
Initializing Z
200
```

Обратим внимание, что конструкторы базовых классов вызываются в том порядке, в котором они указаны в списке при объявлении класса Z.

В общем случае, когда используется список базовых классов, их конструкторы вызываются слева направо. Деструкторы вызываются в обратном порядке — справа налево.

ВАРИАНТЫ ЗАДАНИЙ

1) Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».
2) Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».
3) Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».
4) Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».