**Виртуальные функции и полиморфизм**

Виртуальные функции позволяют вызывать динамический полиморфизм – тоесть, использование методов родительского class через указатели/ссылки в дочерних классах.

class Animal {

public:

virtual void speak() { std::cout << "Animal\n"; }

virtual ~Animal() {} // Виртуальный деструктор!

};

class Dog : public Animal {

public:

void speak() override { std::cout << "Dog\n"; }

};

int main() {

Animal\* a = new Dog();

a->speak(); // Dog (без virtual было бы Animal)

delete a; // Корректное удаление благодаря виртуальному деструктору

}

Override – указывает на то, что данный метод виртуальный.

**Указатели на производные классы и полиморфизм.**

void feed(Animal\* animal) { // Принимает любой класс-наследник Animal

animal->speak();

}

int main() {

Dog dog;

Cat cat;

feed(&dog); // Woof!

feed(&cat); // Meow!

}

В данном случае функция (void feed) будет принимать любый дочерние классы от родительского класса (Animal).

Пространство имен namespace

При написании своих функций, возможно, ты хочешь написать целую библиотеку, заточенную под что-то конкретное, но могут возникнуть конфликты при одинаковых методах, например: print();