**15. Понятие класса. Члены класса и доступ к ним. Модификаторы доступа. Указатель this. Поиск имен в области видимости классов. Отличие класса от структуры.**

* **Понятие класса (инфа из лекции Чабанова).**

**Объект (экземпляр класса)** — это основная единица программирования в ООП (объектно-ориентированном программировании). Каждый объект представляет собой уникальный экземпляр класса и имеет свойства (атрибуты) и методы (поведение).

**Класс**, с одной стороны — это программная конструкция в языках программирования, с другой стороны — шаблон (образец), на основании которого создаются объекты. Поэтому объект ещё называет **экземпляром класса**.

Например, возьмём двух котов. У них есть индивидуальные отличия, но в тоже время существует как бы общая схема, по которой собраны все коты. В ООП эта схема называется классом, а каждый из котов – объектом.

* **Члены класса и доступ к ним.**

**Члены класса** — это **переменные состояния (свойства)** и **функции (методы)** этого класса, т.е. объявленные внутри него.

**Свойства** — данные, характеризующие объект (цвет глаз, имя, количество денег на счету и т.д.). Синонимы: атрибуты, поля класса, члены-данные. Основная задача — хранить состояние, в котором находится объект.

**Методы** — функции, реализующие поведение класса, изменяющие состояние класса или предоставляющие к нему доступ (идти, говорить, посмотреть баланс на счету и т.д.). Синонимы: поведение, члены-функции. Основная задача — обеспечивать пребывание объекта только в допустимых состояниях и обеспечивать переход из одного допустимого состояния в другое.

* **Модификаторы доступа (инфа из лекции Чабанова).**

**Инкапсуляция** – один из основных принципов ООП, объединение данных и функций, которые работают с этими данными, в один компонент (например, класс).

В некоторых языках к этому определению добавляют ещё и **управление доступом к данным и функциям (сокрытие)**. Т.е. "капсула" выполняет не только функцию объединения компонентов, но и функцию защиты этих компонентов от внешних воздействий.

В С++ управление доступом реализовано посредством модификаторов доступа: **public, protected**, **private** и ключевого слова **friend**.

1) Для **public**: к члену класса разрешено обращаться внутри класса, в наследниках и во внешнем по отношению к классу коде. Установлен по умолчанию для **struct**, конструкторов, деструкторов и других спец. членов класса, созданных компилятором.

2) Для **protected**: к члену класса разрешено обращаться внутри класса и в наследниках, но запрещено обращаться внешнем по отношению к классу коде.

3) Для **private**: к члену класса разрешено обращаться только в коде, который является частью класса. Для **class** он установлен по умолчанию.

4) Дружественные функции и классы (т.е. к которым применено ключевое слово **friend**) имеют полный доступ к членам класса в независимости от модификатора доступа этого члена (нарушен принцип инкапсуляции).

Чтобы функция или класс стал дружественным, нужно указать объявление внутри целевого класса:

void print(const A& object){

std::cout << object.i;

}

class B{

void print(const A& object){

std::cout << object.i;

}

};

class A{

// объявление дружественных функций

friend void print(const A& a);

// объявление дружественного класса

friend class Person;

private:

int i;

};

Модификатор доступа, указанный в теле класса, создаёт секцию, которая действует до следующего модификатора и все члены класса, которые попадают в эту секцию приобретают указанный модификатор.

Такие секции могут идти в любом порядке и повторятся любое количество раз.

* **Указатель this (взято с сайта otus.ru).**

Ключевое слово this в C++ представляет собой указатель на текущий объект данного класса. Таким образом, через this мы сможем обратиться внутри класса к любым членам этого класса. Пример:

class Point

{

public:

Point (int x, int y){

this->x = x;

this->y = y;

}

private:

int x;

int y;

};

Так вот, чтобы обращаться именно к переменным внутри пользовательского конструктора Point(int x, int y), а не тем, что находятся вне его в теле класса Point, применяется указатель this. При этом после this надо ставить не точку, а стрелку «->».

Иногда указатель this применяется неявно, например:

class SetShowData

{

char someName[128];

int someAge;

public:

void enterData() {

cout << "Ваше имя: ";

cin.getline(someName, 128);

cout << "Ваш возраст: ";

cin >> someAge;

}

void showData() {

cout << "\n" << someName << " " << someAge << endl << endl;

}

};

* **Отличие класса от структуры (из лекции Чабанова, раздел «Чем ООП отличается от структурного программирования»).**

В структурном программировании мы оперировали данными и функциями, которые эти данные преобразовывали, в ООП же мы оперируем объектами, которые взаимодействуют друг с другом.

ООП оперирует более высокоуровневыми идеями, чем структурное программирование, поэтому разные языки предоставляют разный набор инструментов реализующих идеи ООП. Например, в Python полиморфизм реализован за счёт утиной типизации и при этом нет языковых конструкций, обеспечивающих сокрытие доступа, есть договорённость.