Класс

Класс – это тип, определяемый пользователем, доступ к которому осуществляется по ссылке.
Класс объединяет в себе данные (**поля**) и функциональность (**методы**).

Класс представляет собой шаблон, по которому создаются отдельные объекты, **экземпляры** класса.
Все экземпляры одного класса поддерживают одну и ту же логику, состоят из тех же полей, но данные в этих полях различаются у различных экземпляров.

Выделение классов в программе

Должны представлять единую логическую сущность во внешнем мире.

Цели разделения на классы: как можно меньший объект, содержащий максимум связей внутри себя и минимум связей с внешними объектами.

В основе выделения классов лежит принцип **абстракции**: из объекта реального (или виртуального мира) выделяется группа объектов, для которой создается единая модель. Эта модель должна включать все необходимые данные и действия, необходимые для успешного решения задач программы, но не включать то, что не нужно для решения практических задач.

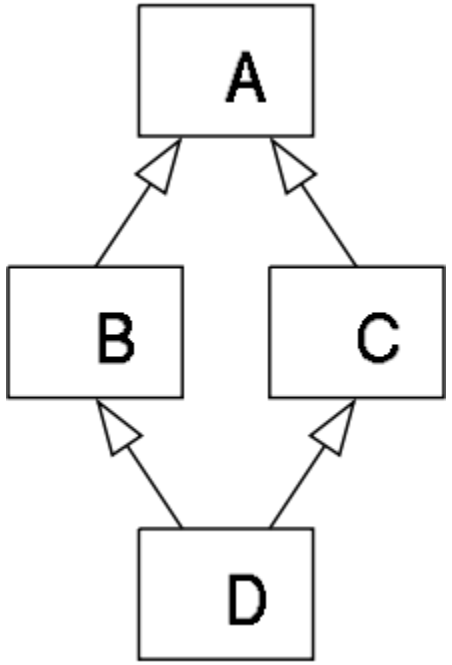
Пример класса в Python

```
class Human(object):

    def __init__(self, profession, is_woman, color_of_eyes, color_of_hair):
        self.profession = profession
        self.is_woman = is_woman
        self.color_of_eyes = color_of_eyes
        self.color_of_hair = color_of_hair

    def change_hair_color (self, color):
        self.color_of_hair = color

    def change_profession (self, profession):
        self.profession = profession
```

Наследование

Возможность расширить поведение класса за счет создания классов-потомков. Существуют языки со множественным наследованием (C++, Python) и наследованием от единственного класса (Java, C#, PHP).

Недостаток множественного наследования – «проблема ромба».



Реализация наследования в Python

```
class Animal(object):
    def __init__(self, legs, name):
        self.legs = legs
        self.name = name

    def sound(self):
        return ‘’
    def eat(self):
        return ‘Hrum-hrum’


class Dog(Animal):
    def __init__ (self, legs, name):
        super().__init__(legs, name)

    def sound(self):
        return ‘Huff’
```

float('nan')

Свойства в Python. Пример

```
import datetime
class Human(object):
    @property
    def Year_of_birth(self):
        return self.__year_of_birth

    @Year_of_birth.setter
    def Year_of_birth(self, year_of_birth):
        if year_of_birth < -5509 or year_of_birth > datetime.datetime.now():
            self.__year_of_birth = float('nan')
        else: self.__year_of_birth = year_of_birth;

    @property
    def Profession(self):
        return self.__profession
    @property
    def Is_woman(self):
        return self.__is_woman
```

Структуры данных

В программировании выделяется несколько основных структур данных. Существуют базовые структуры данных (массивы, хэш-таблицы, графы) и производные от них.
Массив – последовательный блок в памяти для хранения данных (ссылок на данные), доступ к которым можно получить по индексу.
Хэш-таблица – последовательность хранимых значений “ключ – значение”, в которых ключ посредством хэш-функции преобразовывается в адрес, по которому хранятся данные.
Граф – структура, представляющая из себя набор вершин (данных или ключей), которые связаны между собой связями (ребрами).

Частными случаями графа являются связные списки и деревья.

Связные списки

Связные списки представляют из себя набор узлов (нод). Каждый узел содержит в себе данные и ссылки на соседей. В случае **двунаправленного списка** ссылок – две (предшествующий и последующий элементы списка), в случае **однонаправленного списка** – одна (только на последующий элемент).

Однонаправленный список содержит в себе только одну ссылку – на **голову** (первый элемент) списка, двунаправленный – две – на **голову** и **хвост** (последний элемент) списка.

В отличие от массивов списки позволяют легко добавлять и удалять элементы (O(1)). В то же время поиск внутри списка происходит гораздо медленнее (O(N)).

Очереди и стеки

С помощью связных списков легко реализуются такие структуры данных как очереди (работают по принципу FIFO (first in – first out)) и стеки (работают по принципу LIFO (last in – first out)).