Лекция 1

Требования:

Списывание строго карается.

1-4 задания на C++ остальные лабораторные работы на шарпе.

На все задания есть строки (если все они выполнены, то возможен автомат).

На форуме есть методичка (где-то).

У преподавателя из методичек списывать можно.

Литература:
matanit (сайт)
Нреди Буч "введение в объектно ориентированное программирование"

Джон фон Нейман создал основную архитектуру процессора.

Чарльз Бэбидж — первый компьютер.

Эниак 1944 первый компьютер с разделением памяти.

1940-1950 все писали на машинном коде.(языков программирования ещё нет)

± **50-е годы** появляются ассемблеры и Алгол. (1-е поколение языков)

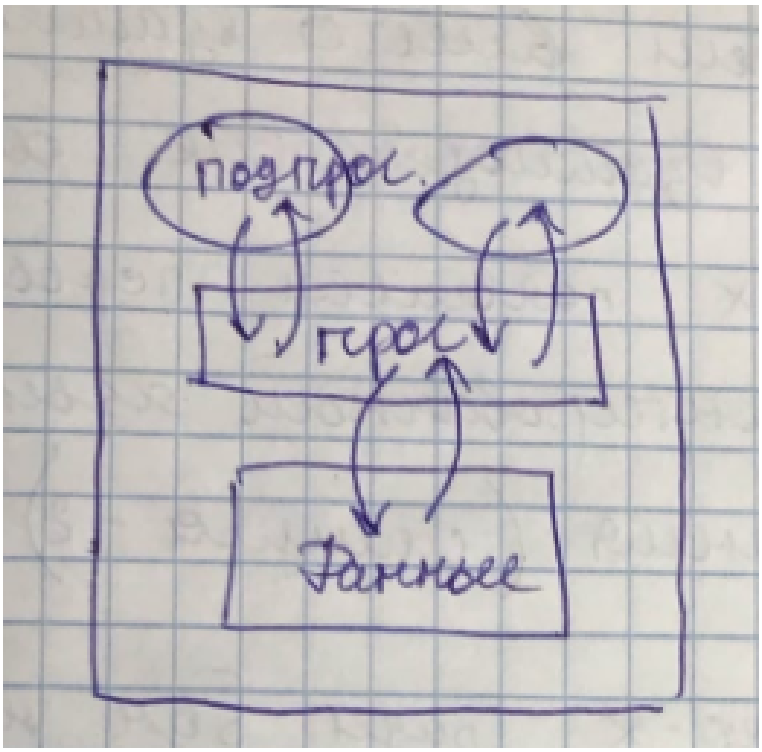
?Мнемоническая команда?

В ассемблере появилась формальная адресация, в алголе появились первые высокоуровненвые команды.

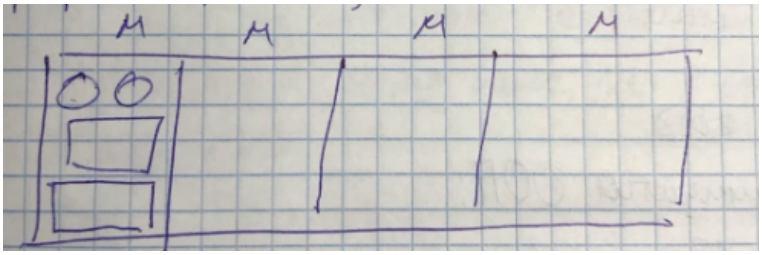
Любой 16-тиричный код ассемблер рассматривает как команды.

В алголе впервые появилось жёсткое деление на программу и данные.

Во втором поколении языков появились функции и подпрограммы + появились докальные и глобальные переменные. (алгол, фортран, ада)



После этого появилось деление на модули (для возможности совместного программирования).



Связь между модулями осуществляется строго интерфейсами или протоколами, прямого доступа нет. (3-е поколение языков)

Дальше ввели вместо взаимодействия алгоритмов, взаимодействие объектов.

В середине 60-х появился первый объектно ориентированный язык программирования (симула-2)

С середины 80-х Гради бУч начинает развивать ООП.

?Принцип абстракции в программировании?

Лекция 2

Принципы ООП:

1) Абстрагирование.

2) Инкапсуляция (регулировка доступа к характеристикам и поведению объекта)

1. Проблема метода заключается в возможности злоумышленного изменения характеристик.

```
struct cat
{
    string poroda;
    color cEye;
    string name;
    int weight;
};
```

2. Преимущество по сравнению с первым вариантом в защищённости характеристик.

```
class cat
{
private:    // личная часть класса
    string poroda;
    int cEye;
    string name;
    int weight;

    void changeWeight() //защищённый доступ к изменению характеристики
    {
        .....
    }

public: // общедоступная часть класса
    cat(string p, int ceye, string n, int w) // конструктор класса
    {
        poroda = p;
        cEye = ceye;
        name = n;
        weight = w;
    }
    string getP() //геттер
    {
        return poroda;
    }
    int getEye()
```

```
    {
        return cEye;
    }
    string getName()
    {
        return name;
    }
    void setName(string n) //сеттер
    {
        name = n;
    }
    int getWeight()
    {
        return weight;
    }
    void Eat()
    {
        changeWeight();
    }
};
```

Если действие может привести к разрушению объекта, то, как правило, доступ к нему ограничивают (размещают в private) + характеристики (все) обычно не хранятся в общем доступе.

Пользователь = вредитель, поэтому максимально защищайтесь и ограничивайте доступ.

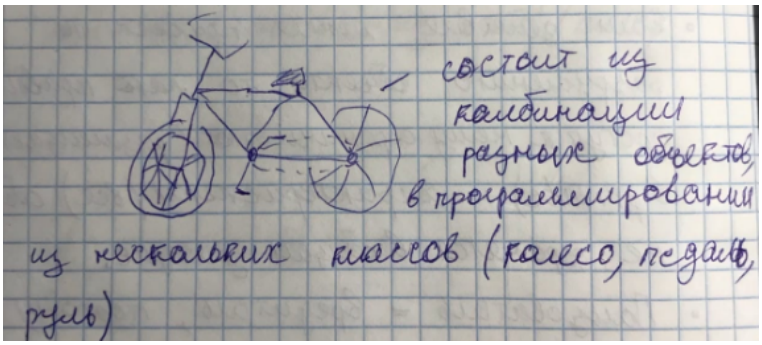
?protected?

?published?

Иерархия бывает двух видов:

1) part of (это часть того).
2)is as (это разновидность того)

part of экономит на написании кода, так как выделяет общий набор характеристик и действий, которые объекты могут совершать.



4.

```
(is as)
class Ep
{
private:
    string marka;
    int power;
    int voltage;
public:
    Ep(){}
    void on(){}
    void off(){}
    void print(){}
};
/* удобно возможностью разом совершать какие"=то действия +
экономия кода + возможность создания классов наследования
(в данном случае классов электроприборов, например, класс телевизора)
*/
class TV : public Ep
{
private:
    int diag;
    string matrix;
public:
    TV() : Ep(){} //конструктор класса
    void print(){}
};
```

Если сделать следующим образом

```
E.print();
t.print();
f.print();
E=t;
E.print();
```

то выведится метод print для E, а не для t

Если на Ep указан virtual (виртуальный класс), то при

```
Ep* ee;
ee = &t;
ee->print();
```

то используется метод print для t, если оставить

```
    E.print();
    t.print();
    f.print();
    E=t;
    E.print();
```

то результат останется прежним.

Лекция 4