Классы в C++

```
class имя {
    int a; - атрибуты.
    int func() {...} - методы.
};
```

① Модификаторы доступа.
- private - (доступно только внутри класса)
- public - (доступны всем вне класса)
- protected - (доступны наслед. функциям

② Конструкторы и деструкторы

RAII - получение ресурса есть инициализация.
- выделяем память на объект при по иниц. конструктора.
- при выходе из обл. видимости память осв. через деструктор.

```
Class MyClass {
    My_class {
        .....
    }
    ~My_class () {
        .....
    }
};
```

① Конструктор по умолчанию.
MyClass()
- реализован. автом. без передачи параметров.

MyClass() default.

② Констр. с параметром.

MyClass (int ∀...){

∀ кол-во

а) My Class a - по. умолчанию
б) MyClass a (5,...) - с параметром.

. констр с параметром может быть несколько.

int value; - поле

а) MyClass (int v) {
    value = v;
}

б) списковая инициализация
MyClass(int v):
    value(v) {
        ....
    }

③ Делегирующий конструктор.
MyClass (int v): value (v) {...}

- MyClass ( ) {
    value = 42)
}

MyClass ( ) : MyClass (42) {...}

④ Копирование
int a = 5;
int b = a;

MyClass (const MyClass& a) {
value = a. value .
};
приватные переменные доступны.

⑤ Деструктор
~ MyClass() {
} - пустой дестр. реализ. авто.

Наследование
public, private; protected
насл.

class A;
class B: public A {

A ↓ (по прим.)	насл. B B: private	A public	A protected
private	не доступно в B private	не дост. в B private	private
public	private	public	protected
protected	private	protected	protected

дост. в main B

дост. в B и насл. B

дост. в B и по насл.

```
class A {
   private:
     int x;
   public:
     int y;
   protected:
     int z;
};
```

```
① class B: private A{ };
   class C: public B{ };
   B b;
   b.x ; - запрет
   b.y ; - запрет
   b.z ; - нельзя
```

```
void: Update
{
  y=100;
  z=100;
}
```

Все приватны.

```
② Class B : public A{
   };
   Class C: public B{
       void update(){
       y=100; - можно
       z=100; можно
   };
```

в main

```
B b:
b.x;- запрещено.
b.y;- можно.
b.z;- нельзя
```

```
(3). Class  B: protected A {..}
     Class  C: public B {
          void update() {
               y = 100;
               z = 100;
          }
```

B b;
b.y - нет
b.z; - нет

~~П~~ RAII при насл.
Констр - от базового кл.
к наследникам.
A, B, C

деструкторы - в обратном порядке.
C, B, A

Запрет на наследование класса

```
Class A final{
     ....
};
```

Множ. наследование.

next page

```
class Base1 {
    public:
        void func1(); };
class Base2{
    public:
        void func2(); };
Class C:
    public Base1,
    public Base2,
    ...
}

C c;
c.func1();
c.func2();
```

```
class D: virtual public Base1
class K: virtual public
            Base1

Class X: D, K
```

Виртуальное наследование.

1. 1 экземпляр базового класса.
2. Расположение в памяти во время работы ПО.

Виртуальные ф-ции.

Перегрузка ф-ции:

```
class A {
  public:
    void func() ~~{}~~ {
      cout<< "text";
    }
};
class B {
    public:
      void func() {
          cout<< "text2";
      }
};
B b;
b.func(); → выв. texta.
```

```
A *a = new B;
  a.func(); → text
```

в A:

```
virtual void func(..){
  A *a = new B;
    a.func() - texta
```

Позднее или динамическое

для класса с вир ф-ей

созд. табл. вирт. ф-ции.

Если ф(х) перегружаем, то по таблице вирт ф-ции, во время выполнения программы найдётся её послед. реализация.

Табл вирт. ф(х) создаётся для ∀ класса при инициал. объекта класса он получает указатель на vtable класс

vtable

• При вызове вирт. ф-ции происходит поиск адреса ф-ии в vtable std::string

```
#include <iostream>
 class Person{
    private:
        std::string _name;
```

! Имена функций и переменных всегда без пробелов.

m_name
_name — чтобы компилятор понимал, что знать!

! Наследование классов — быть аккуратней использовать virtual.