

Классы в C++

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

- ① Модификаторы доступа.
  - private - (доступно только внутри класса)
  - public - (доступны все вне класса)
  - protected - (доступны наслед. функциям)
- ② Конструкторы и деструкторы

RAII - принцип ресурса есть инкапсуляция.

- выделяет память на объект при входе в конструктор.
- - при выходе из обл. видимости память осв. через деструктор.



```

Class MyClass {
 my-class {

 }
 ~ my-class () {

 }
};

```

① Конструктор по умолчанию  
 MyClass()  
 - вызован автомат.  
 без передачи параметров.

MyClass() default.

② Констр. с параметром.

```

MyClass (int v) {
 // кон-во
}

```

а) MyClass a - по умолчанию

б) MyClass a (5, ...) - с параметром

• констр. с параметром может  
 быть несколько.



int value; - now

1) MyClass (int v) {  
    value = v;  
}

2) Стандартный универсальный  
MyClass (int v):  
    value (v) {  
        ...  
    }

3) Дефолтный конструктор

MyClass (int v): value (v) {...}  
- MyClass () {  
    value = 42;  
}

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

4) Конюрование

int a = 5;

int b = a;



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

приватные переменные доступны.

(5) Деструктор  
 ~ MyClass() {

} - ручная генер. dealloc. абн.

Наследование

public; private; protected  
 наслед.

class A;

class B: public A {

| A ↓<br>(no наслед.) | наслед. B<br>B: private     | A<br>public                 | A<br>protected |
|---------------------|-----------------------------|-----------------------------|----------------|
| private             | не наследуется B<br>private | не наследуется B<br>private | private        |
| public              | private                     | public                      | protected      |
| protected           | private                     | protected                   | protected      |

наслед. B и наслед. B  
 наслед. B

наслед. B и  
 no наслед.



```

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

```

①

```

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

```

void: update {  
y = 100;  
z = 100;  
}

❌  
не пробовать.

②

```

class B: public A {}
};

```

```

class C: public B {
 void update() {
 y = 100; - можно
 z = 100; - можно
 };
};

```

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 - не

b.z - не

~~RAII~~ RAII при наслед.

Констр - от базового кл.

к наследникам.

A B C

деструкторы - в обратном порядке

C, B, A

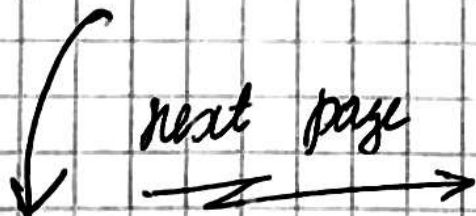
Запрет на наследование класса

Class A final {

...

};

Множ. наследование.



```

class Base1 {
 public:
 void func1(); };

```

```

class Base2 {
 public:
 void func2(); };

```

```

class C:
 public Base1,
 public Base2,
 ...
}

```

```

class D: virtual public Base1,
class K: virtual public Base1
class X: D, K

```

```

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

```

Виртуальное наследование.

1. 1 экземпляр базового класса.

2. Расположение в памяти во время работы ПО.



Виртуальные ф-ции.  
Переопределен ф-ции:

```
class A {
```

```
public:
```

```
void func() {
```

```
 cout << "text";
```

```
}
```

```
};
```

```
class B {
```

```
public:
```

```
void func() {
```

```
 cout << "text2";
```

```
}
```

```
};
```

```
B b;
```

```
b.func();
```

```
 // выв. text a.
```

```
A *a = new B;
```

```
a->func(); → text
```



в А:

```
virtual void func(...){
```

```
 A *a = new B;
```

```
 a.func() - text A
```

Позже или одновременно

же класса с вир. ф-ей

созд. табл. вирт. ф-ций.

Если f(x) перегружен, то по таблице  
вирт. ф-ций, во время выполнения  
программы найдем её позицию  
реализации.

Табл. вирт. f(x) создается до  
и класса при инициал. объект  
класса он получает указатель  
на vtable класс

vtable

• При вызове вирт. ф-ции происходит  
поиск адреса ф-ции в vtable.

std::string

```
#include <iostream>
```

```
class Person{
```

```
private:
```

```
std::string _name;
```

Цена покупки и перепродажа  
из прибыли.

m\_name — это компьютерный псевдоним,  
\_name

это имя!

! Использование классов — быть окруженным  
использовать virtual.