

8. Статични член-даници. Изключения. Обработка на изключения. Щерархия на изключенията и примери. Изключения в конст. и дестр. Нива на exception safety.

Static - Деструктира се при края на програмата и
- ф-ции се създават при първото създаване на обекта.
- член-даници
- член-ф-ции
- данни в ф-ции

I. Статични ф-ции

A. CPP

```
static void f() {  
    void g();  
}
```

Отговарят на namespace ϵ f();

Действителна е до една компилационна единица (един.cpp файл)

edge case
Същото важи и за static променливи.

B. CPP

```
f();  
g(); v
```

ϵ f();

II. Статични член-даници

Up: class X {

```
    static int z;  
    int p;
```

```
    void f() {  
        z++; p--;
```

}

```
    static void f() {  
        z++;  
        p--; }  
}
```

3

Глобален обект съхранен в X.

Не е обвързан с конкретна инициал.

Зададена се бройка int X::z = 0, ако не се зададе се приема def. стойност при int този е 0.

III Член ф-ции

Нр: class A {

 static f() { ... }

}

Всички статични ф-ции могат да се използват
само други статични ф-ции и променливи.
Те не са обявени с ун.

edge case

Не могат да са const/mutable, защото те
фактически не са членове.

edge case

Класът само от статични ф-ции е адекватен
клас.

IV Статични данни във ф-ции

class Obj {

 void f() {
 static int called = 0;
 called++;

}

3

main.cpp

```
Obj x;  
x.f() // called = 1  
x.f() // called = 2  
x.f() // called = 3
```

Създава се 1 път
при първоначалното
извикване на ф-цията.

Всеко следващо из-
викване дара би със
същата ф-ция.

edge case:

static не променя зоната на обектите

Изключения

- сигнален механизъм, скойто програмата използва, че е възникната грешка

Tip:

```
class A
A() {
    try {
        B();
        int x=3;
    } catch(int) {}
}
```

```
class B
B() {
    g();
}
g() {
    throw 3;
}
```

Обработка на изключения

try- метод за дефиниране на блок от код, в който може да настани грешка
catch- позволява за дефинирането на блок от код, който да се изпълни при грешка в try блока

При грешка в кода, програмата:

1) проверява дали има "подходящ" catch блок, който да хвърле грешката

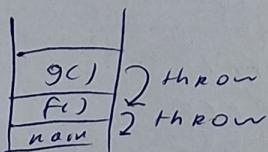
Tip: try {

 throw 3;

} catch (char* ex){ .. }

ако няма програмата се прекратява чрез std::terminate, а ако има:

2) изпълнява се stack unwinding - програмата се връща назад по стека докато не намери правилното място, където да се обработи грешката



16P: теги

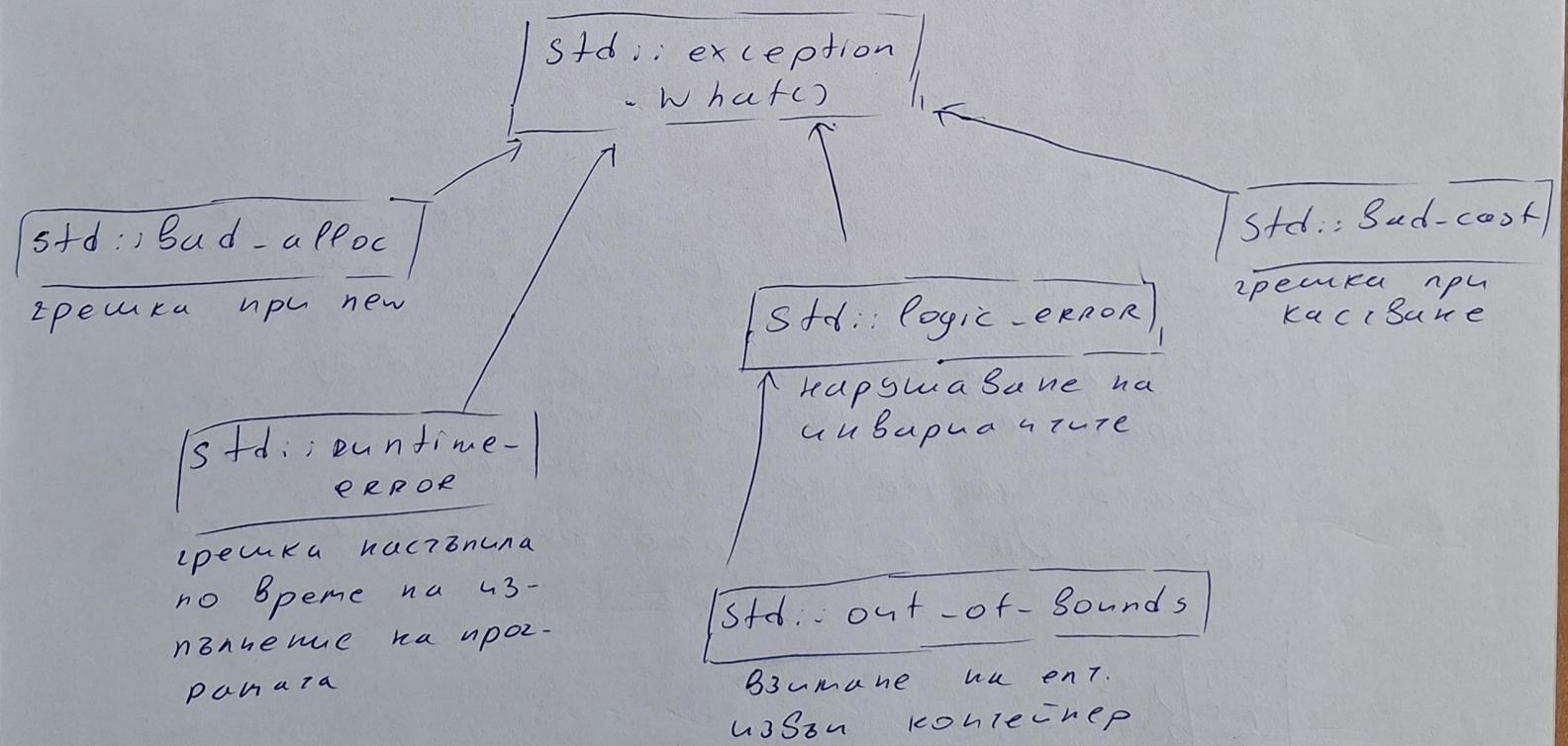
```
3 catch (int a) {  
3 catch (double d) {  
3 catch (const A&){  
3 catch (std::bad_alloc) {  
3 catch (std::exception) {  
3 catch (...) {
```

Чтобы не было конфликтов при конкретном
запуске на catch блоке от ~~как-других~~
до как-одного

1) catch (std::exception) - ~~хвата~~ само зреинки от STL

catch (...) - хвата все как бы зреинки

Черектър на изключения и пример



Изключения в конструктора.

- при хвърляне на изключение в констр, се избикват дескр. на та и напълно заделени обекти.

~~трябва~~
- трябва да се напечатат за този или ресурс

Лп:

1) class A

```
A() {  
    throw 37; // не се бука  
}  
            // ~A()
```

2) A() {

```
char* _data = new char[37];  
:  
throw 37; // трябва да се напечатат за  
изисването на _data
```

3) A: B, C, D

```
A(): B(), C(), D()  
    :  
    throw
```

// B(), C(), ~B()

За да решим проблем от 2), ни трябва следната стратегия:

A() {

char* _data = new char[37];

try {

throw 37;

} catch (int a) {

delete[] _data; // изисване

throw; // rethrow на своята грешка

}

⚠ При хвърлянето на 2 последователни грешки също се избика std::terminate

Лп: A(): B()
 :
 throw 42;

~B() {
 throw 420; // std::terminate

3

Изключения в деструктора

- след C++11 деструктора е автоматично генериран с поексепт
- не трябва дестр да хвърля грешка nothrow

Начин за избягване на грешка при задаване на параметър

Tip: char* str = new (std::nothrow) char[ST];

При грешки str = nullptr;

- главна опасност при .clone()

Нива на exception safety

1. Strong exception guarantee -

ако ф-ция хвърли грешка, програмата ще се върне до свързаното и преди истичане на грешката

Tip: vector(10), push на 11-ти елемент, останалите 10 се запазват

2. Basic exception guarantee -

ако ф-ция хвърли грешка, програмата ще продолжи да работи и е във валидно състояние

Tip: vector(10), push на 11-ти елемент, не се знае какво ще е състоянието на останалите, но все още ще може да се добавят

3. No exception guarantee -

нито едно гаранция за ниво

4. No throw guarantee -

никога нито да се върне грешка

Tip: move(cc, oP = & ~cc)