

Хитрости RAI

Максим Лысков

Минск, 2015

Знакомимся?



Максим, старший инженер-разработчик в компании EPAM Systems.

В настоящее время участвую в разработке системы иерархического хранения и репликации данных.

Оно течет!



```
int importantFunc(int someParam)
{
    char * buf = new char[BUFSIZE];
    if (conditionFunc1(someParam)) {
        lessImportantFunc1(buf); // (0)
        delete buf;
        return 1;
    }
    else if (conditionFunc2(someParam)) {
        lessImportantFunc1(buf);
        return 2; // (1)
    }
    delete [] buf;
    return 0;
}
```

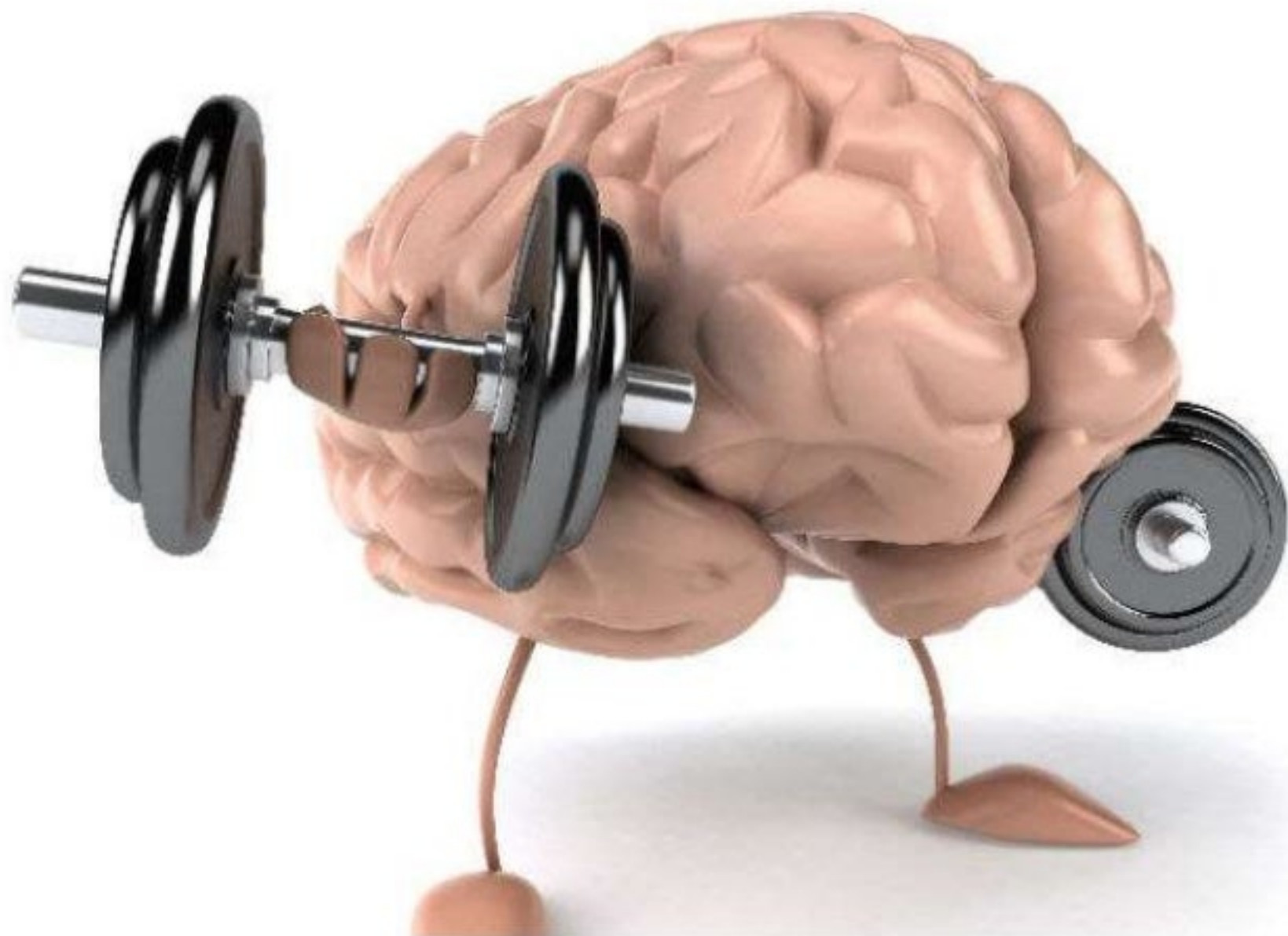
Ресурсы



Память

`new/delete`

`malloc/free`





Мьютексы











lock/unlock













open/close

Файлы

Хитрости RAII

стандартный	50 мм		70 мм	
		257		269
				
		273		271
				
		281		280
				
		275		277
				
		287		286

разборный		
	695	683
		
	696	684
		
	698	687
		
	697	686
		
	700	689





acquire/release

Что угодно

Гарантии безопасности исключений по Абрахамсу

1. No throw / no except.
2. Basic - нету утечек, состояние объектов - консистентно.
3. Strong - транзакционная модель, состояние объекта не изменилось

Resource acquisition is initialization

```
class RAIIObject
{
public:
    RAIIObject() : m_handle(openHandle()) {
        if (!handleValid(m_handle)) {
            throw std::logic_error("Invalid handle");
        }
    }
    ~RAIIObject() {
        if (handleValid(m_handle)) {
            closeHandle(m_handle);
        }
    }
    operator handle() { return m_handle; }
private:
    handle m_handle;
};
```

```
int handleHandle(handle h);

int bar()
{
    RAIIObject safeHandle;
    handleHandle(safeHandle);
}
```


Работает из коробки

smart pointers - `unique_ptr`, `shared_ptr`, `weak_ptr`, `auto_ptr`

STL containers - `vector`, `string`

STL streams - `fstream`

thread

thread locks - `lock_guard`, `unique_lock`, `shared_lock`

Работает из коробки - 2

vector в качестве in/out buffer

```
int importantFunc(int someParam)
{
    vector<char> buf(BUFSIZE, 0);
    if (conditionFunc1(someParam)) {
        lessImportantFunc1(buf.data());
        return 1;
    }
    else if (conditionFunc2(someParam)) {
        lessImportantFunc1(buf.data());
        return 2;
    }
    return 0;
}
```

Работает из коробки - 3

unique_ptr в качестве in/out buffer

```
int importantFunc(int someParam)
{
    unique_ptr<char[]> buf(new char[BUFSIZE]);
    if (conditionFunc1(someParam)) {
        lessImportantFunc1(buf.get());
        return 1;
    }
    else if (conditionFunc2(someParam)) {
        lessImportantFunc1(buf.get());
        return 2;
    }
    return 0;
}
```

Работает из коробки - 4

`unique_ptr` / `shared_ptr` с пользовательским удалением (*)

(*) ограничение - ресурс должен быть подобен указателю

```
handle* openHandle1();  
    int openHandle2(handle** pph);  
    void closeHandle(handle* h);  
    void fooBar()  
    {  
        shared_ptr<handle> handle_shared(openHandle1(), &closeHandle);  
        handle* h = nullptr;  
        openHandle(&h);  
        unique_ptr<handle, void(*) (handle*)> handle_unique(h, &closeHandle);  
        doSomethingWithHandles(handle_shared.get(), handle_unique.get());  
    }
```

Еще одна дыра

```
int foo(int i1, char* p, int i2) {
```

```
    ...
```

```
    delete p;
```

```
}
```

```
void bar() {
```

```
    ...
```

```
    result = foo(calculatel1(), new char[SIZE], calculatel2());
```

```
    ...
```

```
}
```


`make_shared`

`make_unique`



RAII своими руками

```
class RAIIObject
{
public:
    RAIIObject() : m_handle(openHandle()) {
        if (!handleValid(m_handle)) {
            throw std::logic_error("Invalid handle");
        }
    }
    ~RAIIObject() {
        if (handleValid(m_handle)) {
            closeHandle(m_handle);
        }
    }
    operator handle() { return m_handle; }
private:
    handle m_handle;
};
```

Разработка безопасных классов

все члены - RAII

инициализация - в списке инициализации, если нужно - отдельная функция

Советы

Все в RAI

Избегайте работы с “голыми” ресурсами

Используйте стандартные механизмы языка и его стандартную библиотеку

...

PROFIT!



Спасибо!

