# Конспект №7: RAII, finally.

## *Теоретический материал.*

Идиома RAII берёт своё начало в языке C++ и заключается в том, что некоторый ресурс захватывается в конструкторе объекта, и освобождается в его деструкторе. А поскольку деструктор локальных объектов вызывается автоматически при выходе из метода независимо от причины, то использование этой идиомы является самым простыми и эффективным способом написания сопровождаемого C++ кода, безопасного с точки зрения исключений. RAII может использоваться для любых разделяемых объектов или ресурсов:

* для выделения памяти.
* для открытия файлов или устройств.
* для критических секций и мьютексов.

Дополнительно:

*Мью́текс* (англ. mutex, от mutual exclusion — «взаимное исключение») — примитив синхронизации, обеспечивающий взаимное исключение исполнения критических участков кода

*Критическая секция* — участок исполняемого кода программы, в котором производится доступ к общему ресурсу (данным или устройству), который не должен быть одновременно использован более чем одним потоком выполнения

Важный случай использования RAII - "умные указатели", классы, инкапсулирующие владение памятью. Например, в стандартной библиотеке C++ для этого есть auto\_ptr или unique\_ptr.

Основной смысл RAII заключается в автоматическом гарантированном освобождении ресурсов, не только при завершении программы и вызове деструктора, но и при выбрасывании исключения. Например: ресурсы, создающиеся по ссылке, освобождаются автоматически при покидании зоны видимости, а те, что создаются по указателю, нужно освобождать вручную. При этом в самом объекте должен быть предусмотрен запрет копирования объекта, чтобы данный объект гарантированно являлся единстенным ресурсом в памяти.

Как раз-таки по причине наличия идиомы RAII, C++ не поддерживает finally блоков.

## Примеры использования.

Пример №1: абстрактное определение.

Tobj A; // ссылка на стек - освобождается автоматически

Tobj\* B = new Tobj; // выделяется память - освобождать вручную

Пример №2: применение RAII для динамической памяти.

class Stack {

int \*A;

int A\_size;

public:

Stack(int \_size):A\_size(\_size) {

A = new int[\_size];

}

~Stack() {

delete [] A;

}

Stack& operator=(const Stack &right) {

delete [] A;

A\_size = right.A\_size;

A = new int[A\_size];

for(int i = 0; i < A\_size; i++)

A[i] = right.A[i];

}

};

void stack\_use() {

Stack a(50), b(10);

a = b;

}

int main() {

cout << "Hello world!" << endl;

stack\_use();

return 0;

}

Источник:

* [Мью́текс](https://ru.wikipedia.org/wiki/%D0%9C%D1%8C%D1%8E%D1%82%D0%B5%D0%BA%D1%81)
* [Критическая секция](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B5%D0%BA%D1%86%D0%B8%D1%8F)
* [RAII (1)](https://www.youtube.com/watch?v=4hZLYpCJZyE&ab_channel=%D0%9D%D0%9E%D0%A3%D0%98%D0%9D%D0%A2%D0%A3%D0%98%D0%A2)
* [RAII (2)](https://foxford.ru/wiki/informatika/raii-poluchenie-resursa-est-initsializatsiya)
* [RAII (3)](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%B0_%D0%B5%D1%81%D1%82%D1%8C_%D0%B8%D0%BD%D0%B8%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)
* [Почему С++ не поддерживается finally](https://coderoad.ru/161177/%D0%9F%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%B2%D0%B0%D0%B5%D1%82-%D0%BB%D0%B8-C-%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8-finally-%D0%98-%D1%87%D1%82%D0%BE-%D1%8D%D1%82%D0%BE-%D0%B7%D0%B0-RAII-%D0%BE-%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%BE%D0%BC-%D1%8F-%D0%B2%D1%81%D0%B5-%D0%B2%D1%80%D0%B5%D0%BC%D1%8F-%D1%81%D0%BB%D1%8B%D1%88%D1%83)