Resources and RAII

Agenda

- Resources
  - Definition
  - Examples

- RAII
  - Definition
  - Concept
  - Example
  - Reasons (not) to use

- Demo

Resources

Resources

Definition:

- Managed parts of program
- Sometimes limited
- Have to be acquired and freed ➔ possible source of problems

Examples:

- Files
- Heap
- Sockets
- Interfaces
- Functionalities beyond scope of program (e.g. (USB) devices)

RAII

RAII - General

- „Resource Allocation Is Initialization"

- Widely spread concept in higher programming languages

- Reason: Avoiding problems while managing resources

- Part of some programming language specification (e.g. `std::vector`, `std::string` in C++)
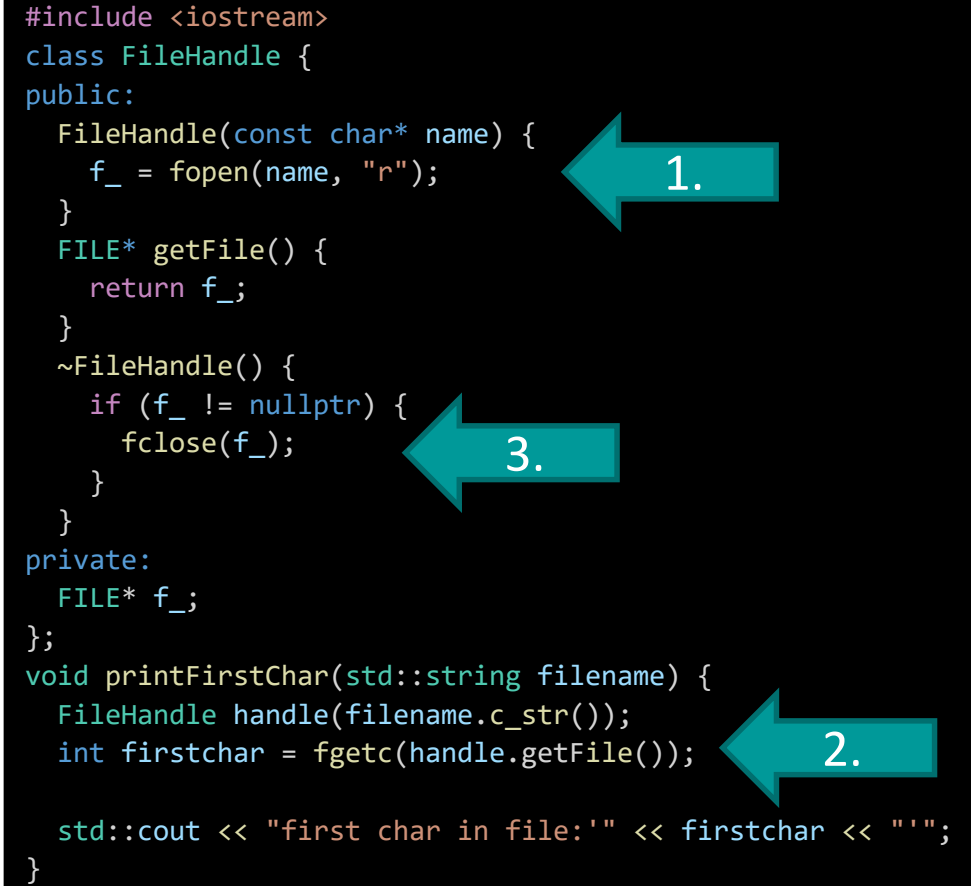
RAII - Concept

Using classes:

1. Acquire resource in class constructor

2. Use resource

3. Free resource in class destructor

```cpp
#include <iostream>
class FileHandle {
public:
  FileHandle(const char* name) {
    f_ = fopen(name, "r");          // 1.
  }
  FILE* getFile() {
    return f_;
  }
  ~FileHandle() {
    if (f_ != nullptr) {
      fclose(f_);                   // 3.
    }
  }
private:
  FILE* f_;
};
void printFirstChar(std::string filename) {
  FileHandle handle(filename.c_str());
  int firstchar = fgetc(handle.getFile());   // 2.

  std::cout << "first char in file:'" << firstchar << "'";
}
```

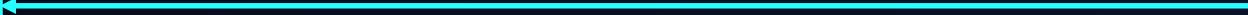Example in C++

RAII - Example in Java

Try-with-resources:

```java
static String readFirstLine(String path) throws IOException {
    try (BufferedReader br = new BufferedReader(new FileReader(path))) {
        return br.readLine();
    }
}
```

Without try-with-resources (no RAII):

```java
static String readFirstLine(String path) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(path));
    String text = br.readLine();
    br.close();
    return text;
}
```

Freeing resource

RAII - Example in C#

Using:

```csharp
public static void Main(){
    string path = @"c:\temp\MyTest.txt";
    if (!File.Exists(path)){
        using (StreamWriter sw = File.CreateText(path)){
            sw.WriteLine("Hello");
            sw.WriteLine("And");
            sw.WriteLine("Welcome");
        }
    }
}
```

RAII - Reasons (not) to use it

+

- Resource handling ensured even if problem while using resource

- Clear distinguishing between resources and other parts of program

- Preventing resource leaks

- Already used in many programming libraries

- "You do not have to care about it while using the resource"

-

- Some very rare cases do not allow use of RAII

Demo

Thanks.

YOU MAY NOW FREE THIS RESOURCE.