Working with Pointers and Arrays in C++ 20

Pointing to an Address in Memory

Mateo Prigl
Software Developer

Prerequisites

C++ syntax and structure

Compiling and running C++ programs

Basic concepts: condition statements, loops, functions...

Basics of object-oriented programming

Why Learn About "Raw" Pointers?

Legacy

Legacy codebases

C++ is a superset of C

Raw pointers are a common occurrence

To understand the abstractions

Modern C++ uses abstractions, like smart pointers

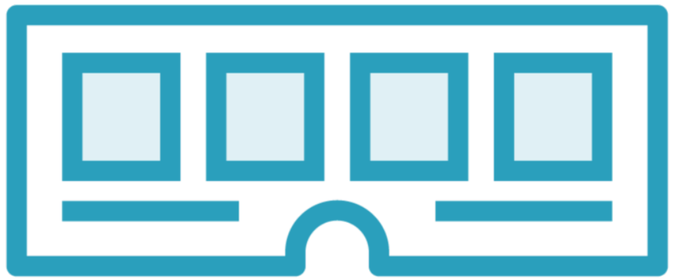
These abstractions are wrappers around raw pointers

“People who think they know everything really annoy those of us who know we don't.”

Bjarne Stroustrup, creator of the C++ programming language

Welcome to the course!

C++ Programs and Memory



Compiled program will get loaded into RAM

Process is an execution of a program

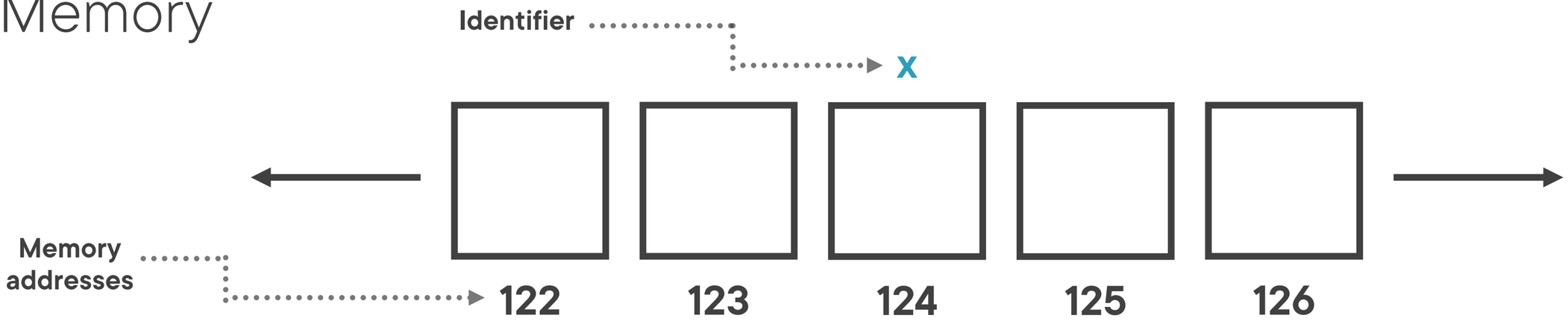
Each process will get its own piece of memory

C++ implements abstractions to simplify management of that piece of memory

```
int x = 2;
```

```
int x = 2;
```

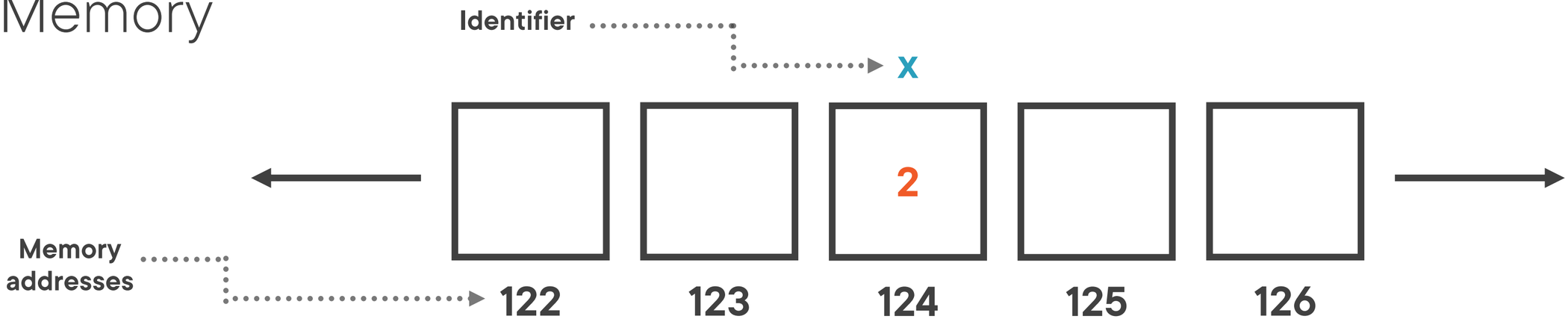
Memory



```
int x = 2;

&x;      // 124
// & - address-of operator
```

Memory

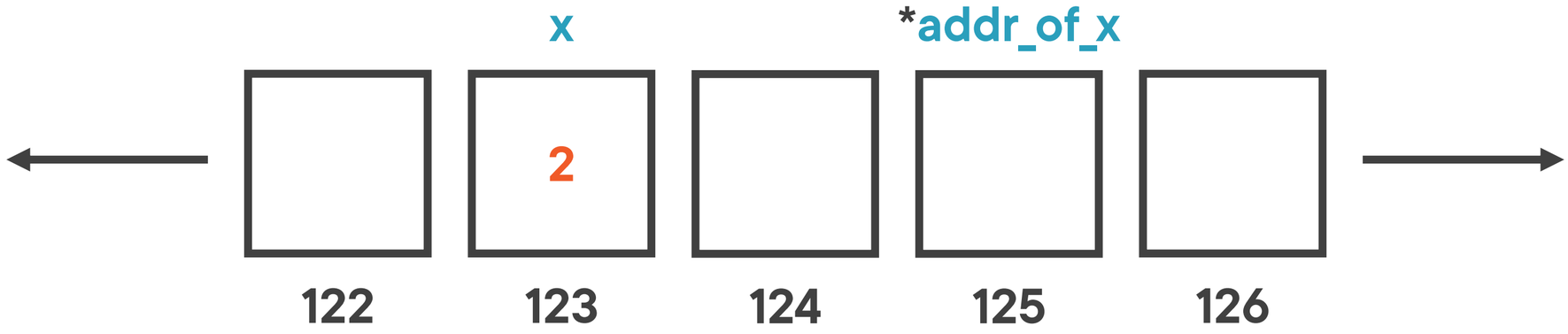


Pointer is a variable that stores
a memory address.

```c
int x = 2;

int *addr_of_x = &x;
```

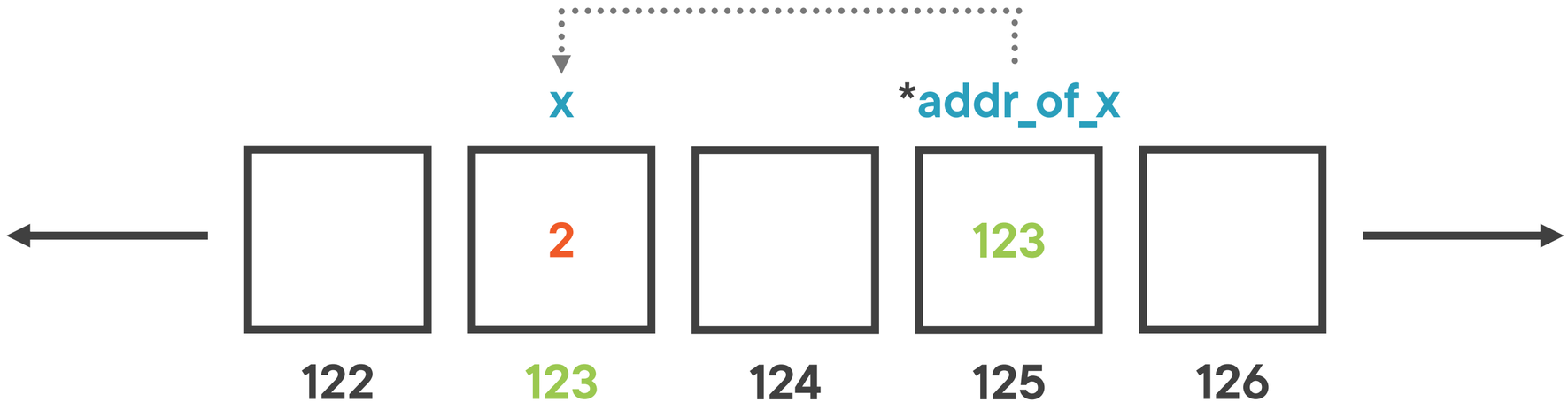
Memory



```
int x = 2;

int *addr_of_x = &x;
```

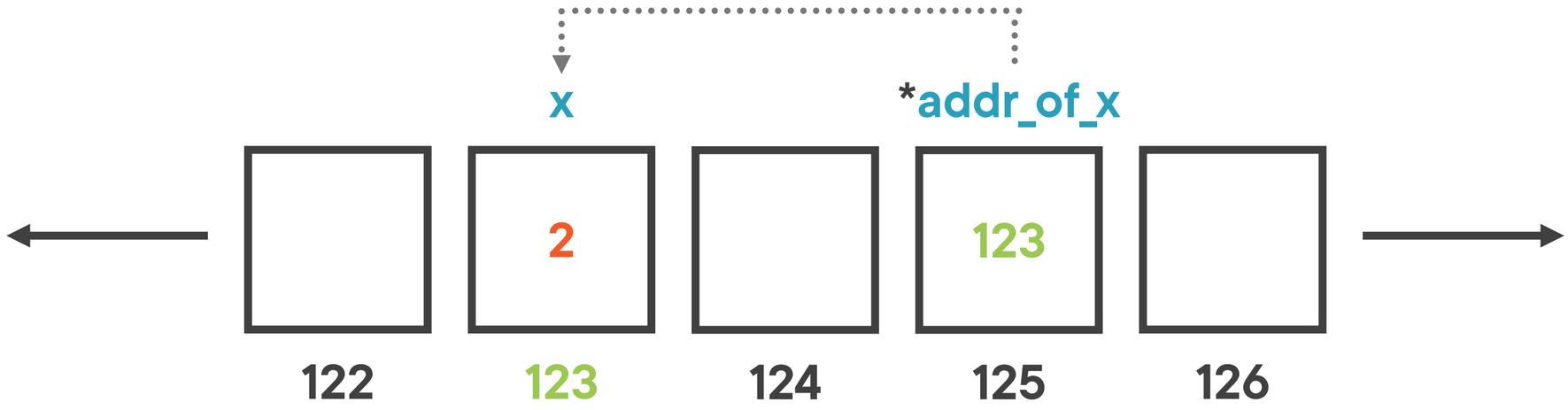
Memory



```
int x = 2;

int *addr_of_x = &x;
```

Memory

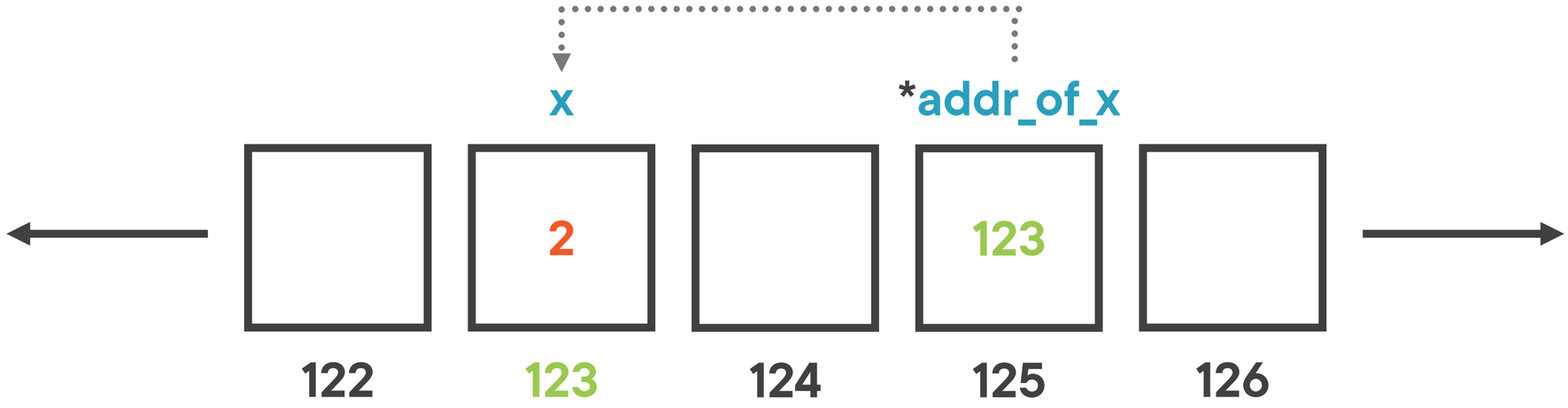


```c
int x = 2;

int *addr_of_x = &x;

*addr_of_x
// * - dereference operator
```

Memory

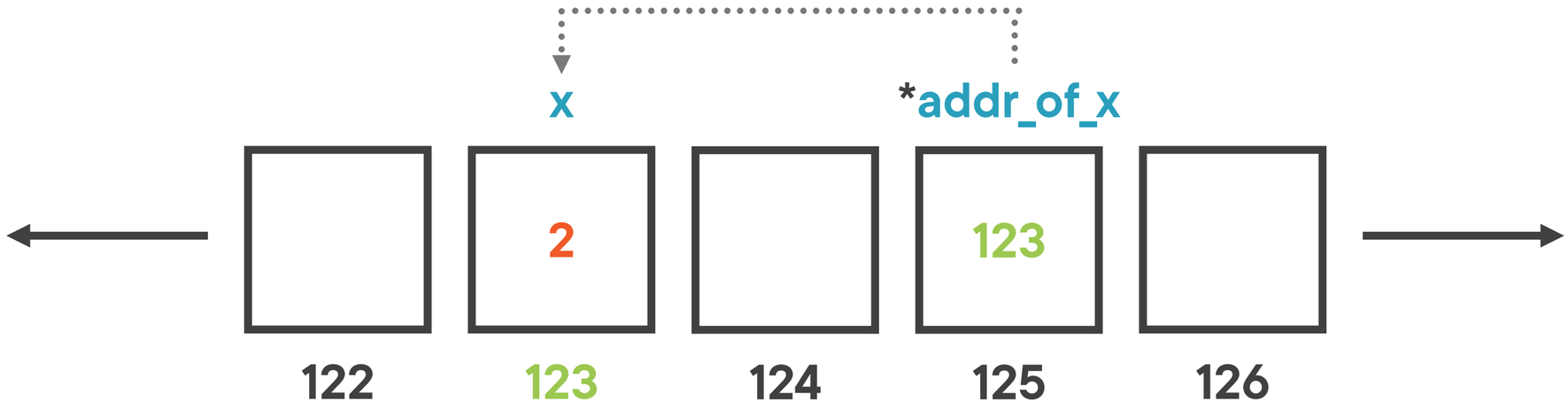


```c
int x = 2;

int *addr_of_x = &x;

*addr_of_x  // 2
// * - dereference operator
```

Memory



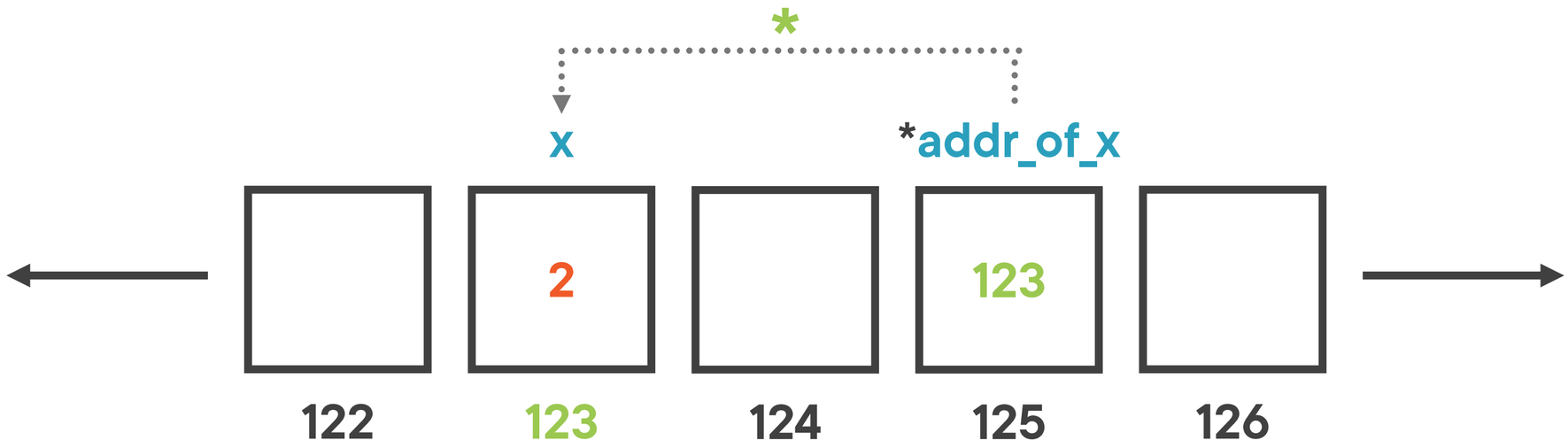
```c
int x = 2;

int *addr_of_x = &x;

*addr_of_x  // 2
// * - dereference operator

*addr_of_x = 5;
```

Memory



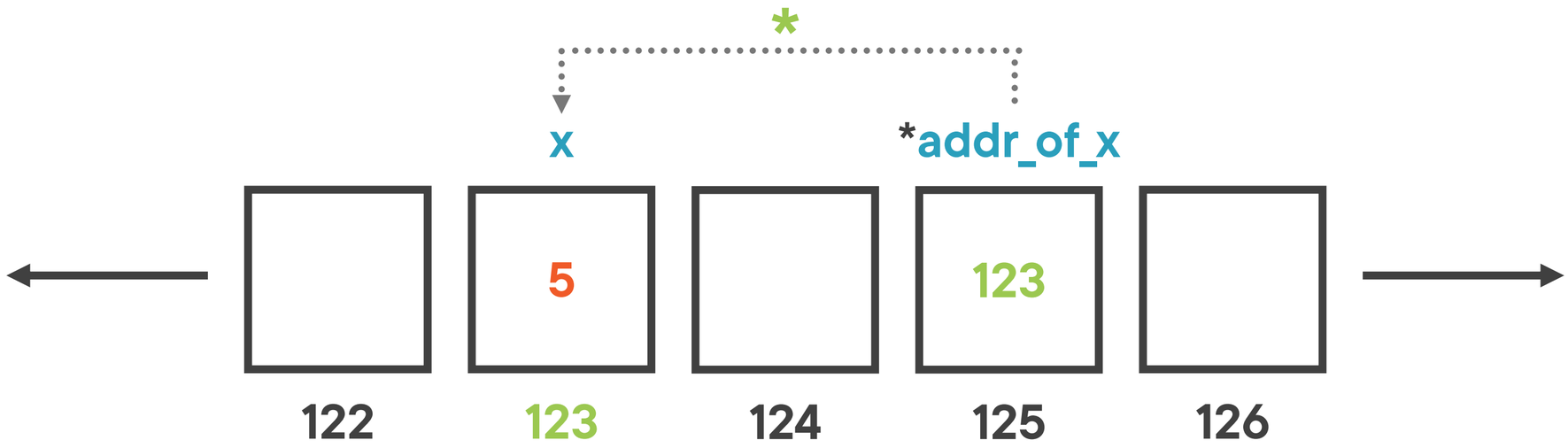
```c
int x = 2;

int *addr_of_x = &x;

*addr_of_x  // 2
// * - dereference operator

*addr_of_x = 5;
```

Memory



RAII

Resource acquisition is initialization

```
class ComplexClass
{
    ComplexClass()
    {
      // allocate resources
    }

    ~ComplexClass()
    {
      // deallocate resources
    }
};

int main()
{
    ComplexClass x;

    // end of scope
}
```

- Constructor
- Destructor
- Object instantiation, constructor is called
- End of scope, destructor is called

Global variables

Static (local) variables

References

Summary

Up Next:
Dynamic Memory Allocation