

C++

David Hackbarth



Sources

git clone <https://bitbucket.org/dhackbarth/gpd-0324.git>

<https://bitbucket.org/dhackbarth/gpd-0324>

History

- 1985 by Bjarne Stroustrup
- successor to C with classes
- Major releases:
 - C++98
 - C++11 (ongoing every three years)
 - C++23 (current)
 - C++26 (draft)

Why?

- cross platform
- compiled language
- efficient
- high performance
- very flexible
- close to hardware
- low-level memory management
- OOP, generics & functional programming

Additional informations

- <https://cppreference.com>
- <http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines>
- <https://learn.microsoft.com/en-us/cpp/?view=msvc-170>
- <https://www.boost.org/>
- <https://www.sfml-dev.org/>
- <https://www.libsdl.org/>

Differences to C#

Files

C#

*.cs

contains complete type declaration
and definition

C++

*.h / *.hpp

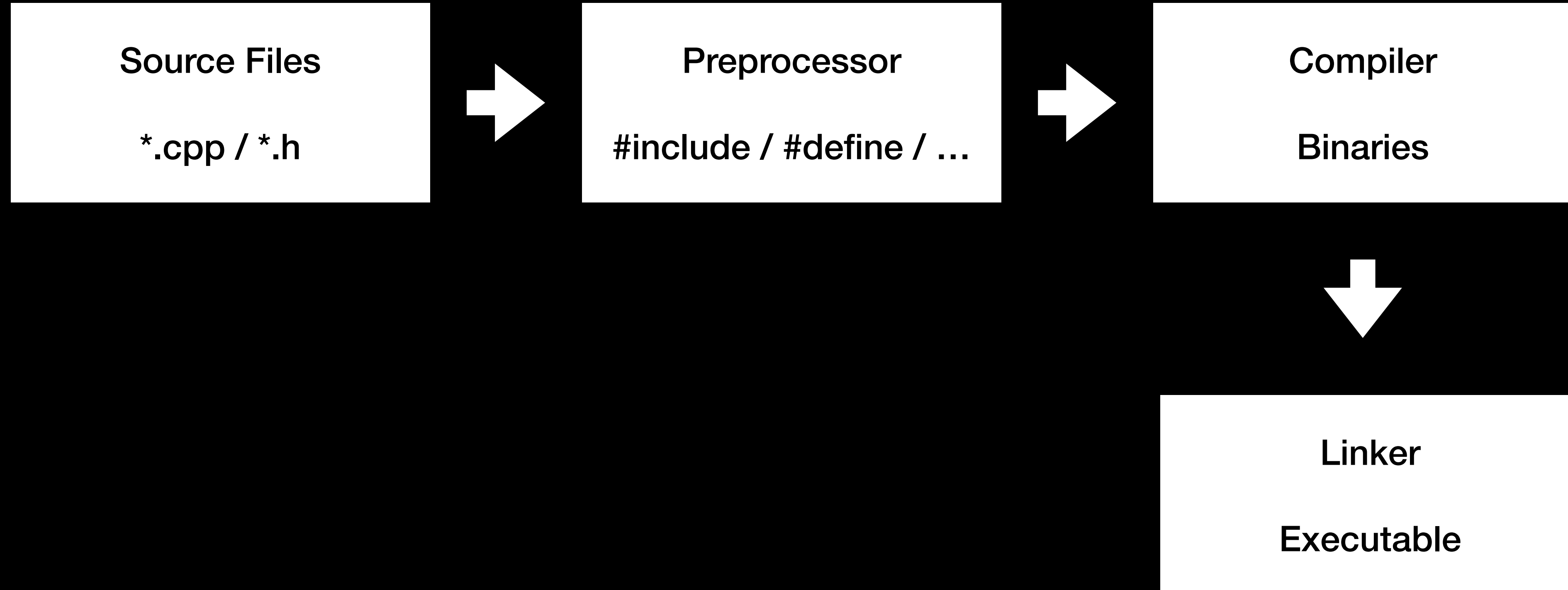
header files contains mostly
declaration and sometimes definitions

*.cpp

contains definition, header files
will be included

everything has to be known before
used

Build Pipeline



Entry Point

C#

```
void Main() { }
```

```
int Main() { }
```

```
void Main(string[] args) { }
```

```
int Main(string[] args) { }
```

C++

```
void main() { }
```

```
int main() { }
```

```
int main(int _argc, char* _argv[])  
{  
}
```

Namespace

C#

Standard: `System`

Operator: `.` (`System.Console`)

Using: `using System;`

C++

Standard: `std` (STL - Standard Template Library)

Operator: `::` (`std::cout`)

Using: `using namespace std;`
different includes needed

Data Types

C#

(s)byte (1 Byte)
(u)short (2 Byte)
(u)int (4 Byte)
(u)long (8 Byte)

float (4 Byte)
double (8 Byte)
decimal (16 Byte)

char (1 Byte)
string

C++

do not exist, char as alternative
(unsigned) short (2 Byte)
(unsigned) int (4 Byte)
(unsigned) long (4 Byte)
(unsigned) long long (8 Byte)

float (4 Byte)
double (8 Byte)
long double (10 Byte)

char (1 Byte)
char[] or std::string

Classes & Structs

C#

```
class TestClass : BaseClass
{
    // everything is private
    public int number;
}
```

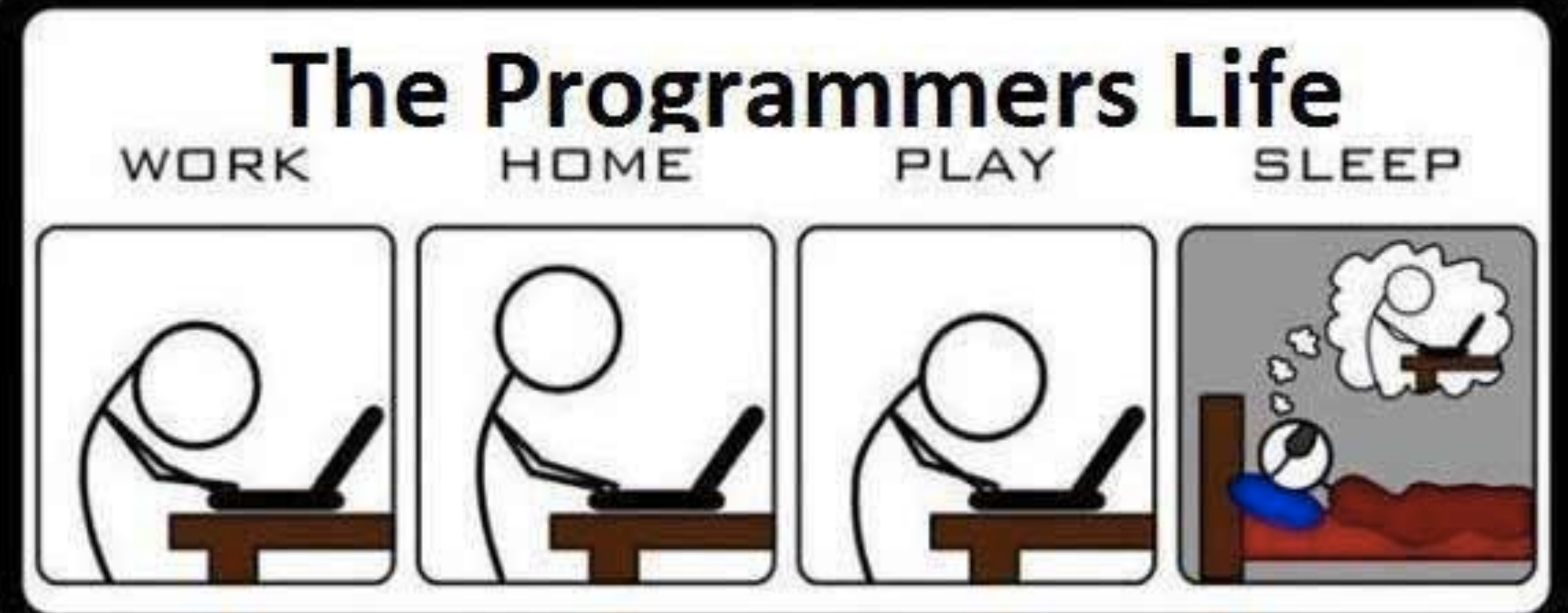
```
struct TestStruct
{
    // everything is private
}
```

C++

```
class TestClass : public BaseClass
{
    // everything is private
    public:
        int number;
};
```

```
struct TestStruct
{
    // everything is public
};
```

Coding Time



Collections

C Arrays

C#

- Object
- knows length
- Range tests

```
int[] array = new int[size];
```

```
int[,] array; or  
int[][] array;
```

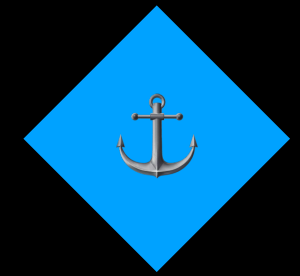
C++

- Pointer on first element
- Do not know length
- No range tests

```
int array[size];
```

```
int array[][];
```

Standard Template Library (STL)



- array - immutable array
- vector - mutable array
- forward_list - single linked list
- list - double linked list
- set - unique sorted keys
- map - key-value pairs, keys are unique and sorted

Coding Time

Have a great weekend
I hope your code behaves on Monday
the same way it did on Friday



Pointers

C#

- no pointers in managed context
- value & reference data types
- garbage collector tidy up memory
- `ref` & `out` parameter modifier

C++

- no difference between value & reference data types
- no garbage collector
- no `ref` & `out` parameter modifier
- pointers & references
- smart pointer since C++11

Stack & Heap

Stack

- limited memory
- local variables
- only accessible in same scope
- variable will be deleted by exiting scope

Heap

- “unlimited” memory
- data created with `new`
- only accessible with pointer
- data has to be deleted with `delete`

Pointer

- variable type
- saves an address
- address can point everywhere on memory (e.g. on stack or heap)
- should be initialized with `nullptr`

Pointer

- will be declared with *

```
int* pI = nullptr;
```

- address will be get by &

```
int i = 22;
```

```
int* pI = &i;
```

- will be dereferenced by *

```
*pI = 33;
```

- or for members of objects with ->

```
obj->member = 11; instead of (*obj).member = 11;
```

Coding Time

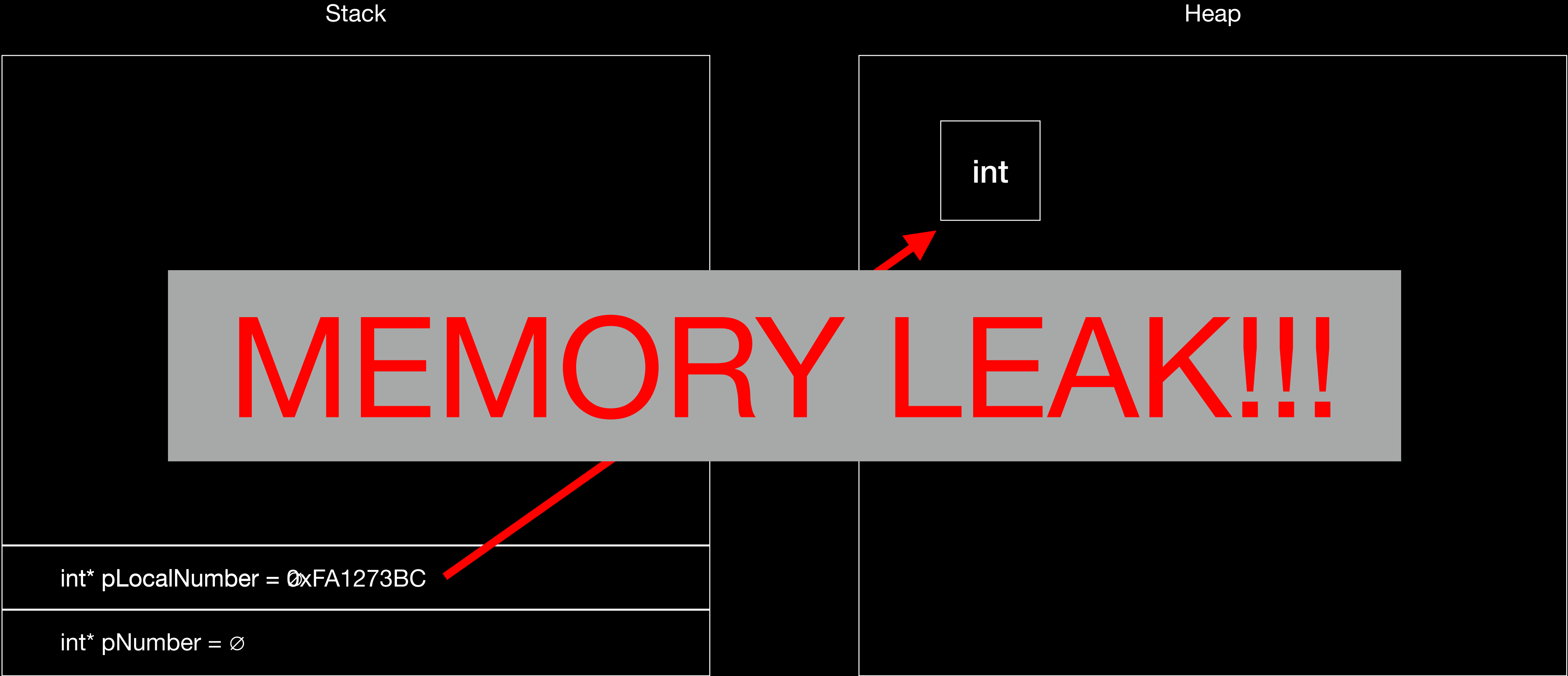
There are two types of people.

```
if (Condition)
{
    Statements
    /*
     ...
     */
}
```

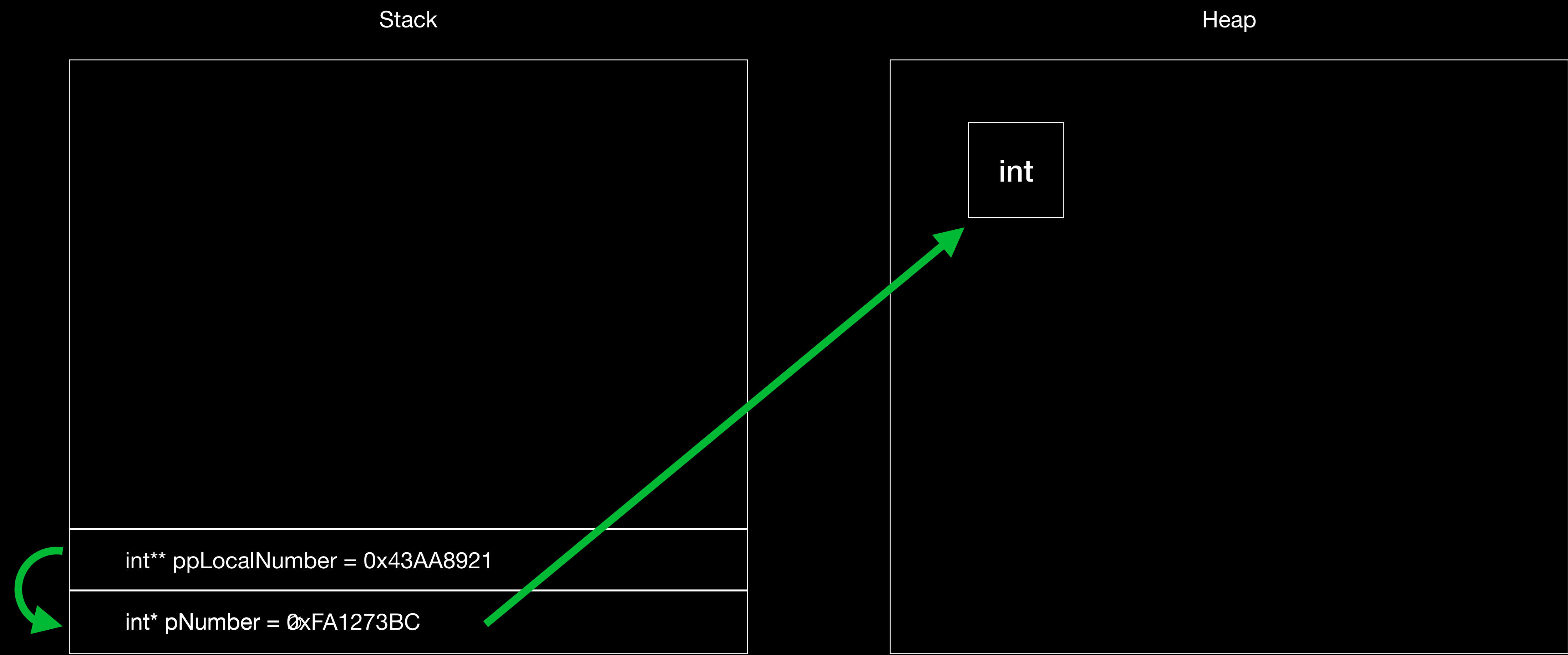
```
if (Condition) {
    Statements
    /*
     ...
     */
}
```

Programmers will know.

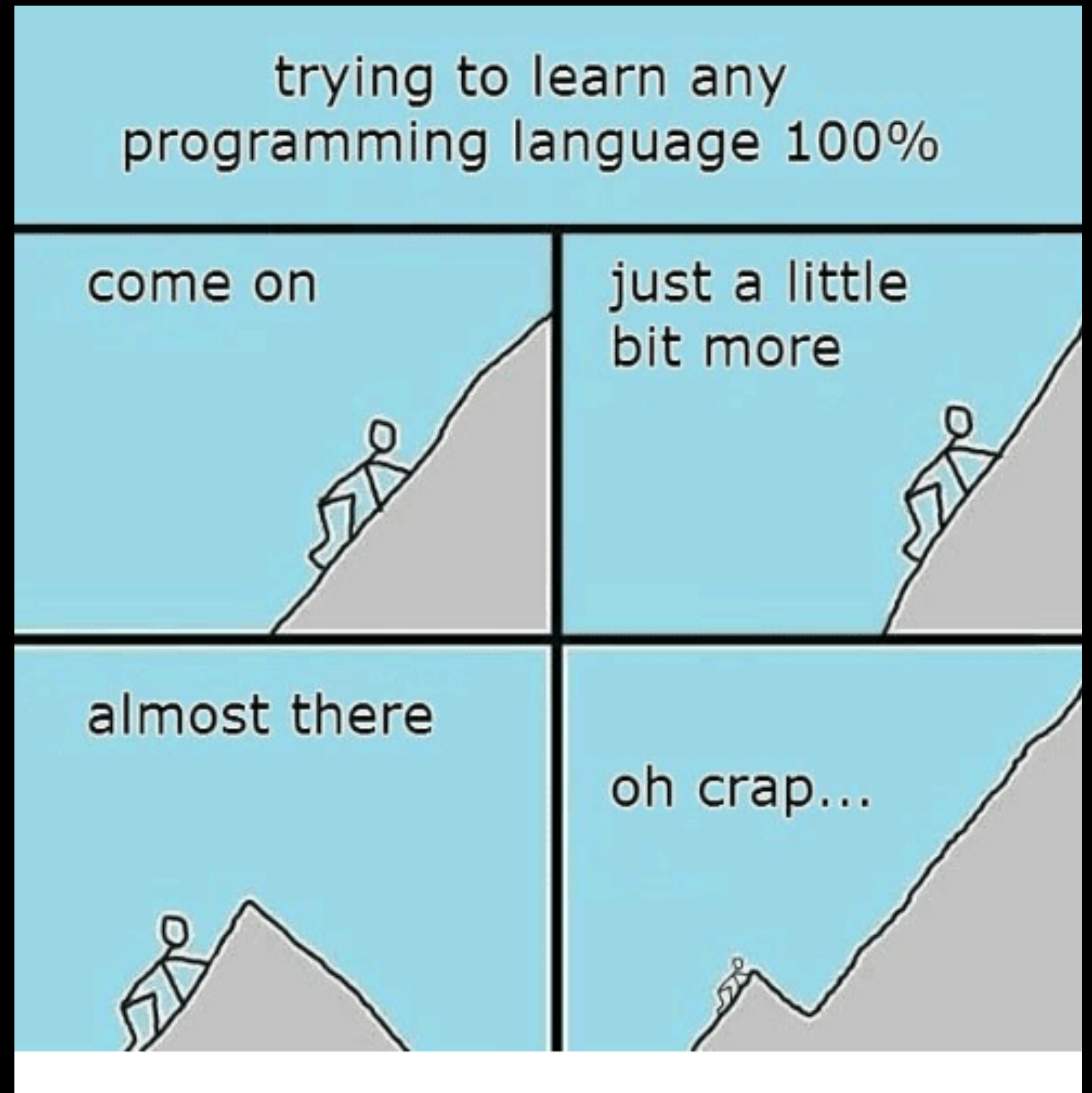
Pointer-Pointer



Pointer-Pointer



Coding Time



Other

Casts

- C cast
- `dynamic_cast<>`
- `static_cast<>`
- `reinterpret_cast<>`
- `const_cast<>`

Inline

- keyword for function/method
 - body will be inlined
 - optimization for runtime
 - compiler decided if used or not
-
- explicit inline
 - implicit inline

Templates

- like generics in C#
- compiler generate variants if needed

```
template<typename T>
```

```
void Print(T value)
```

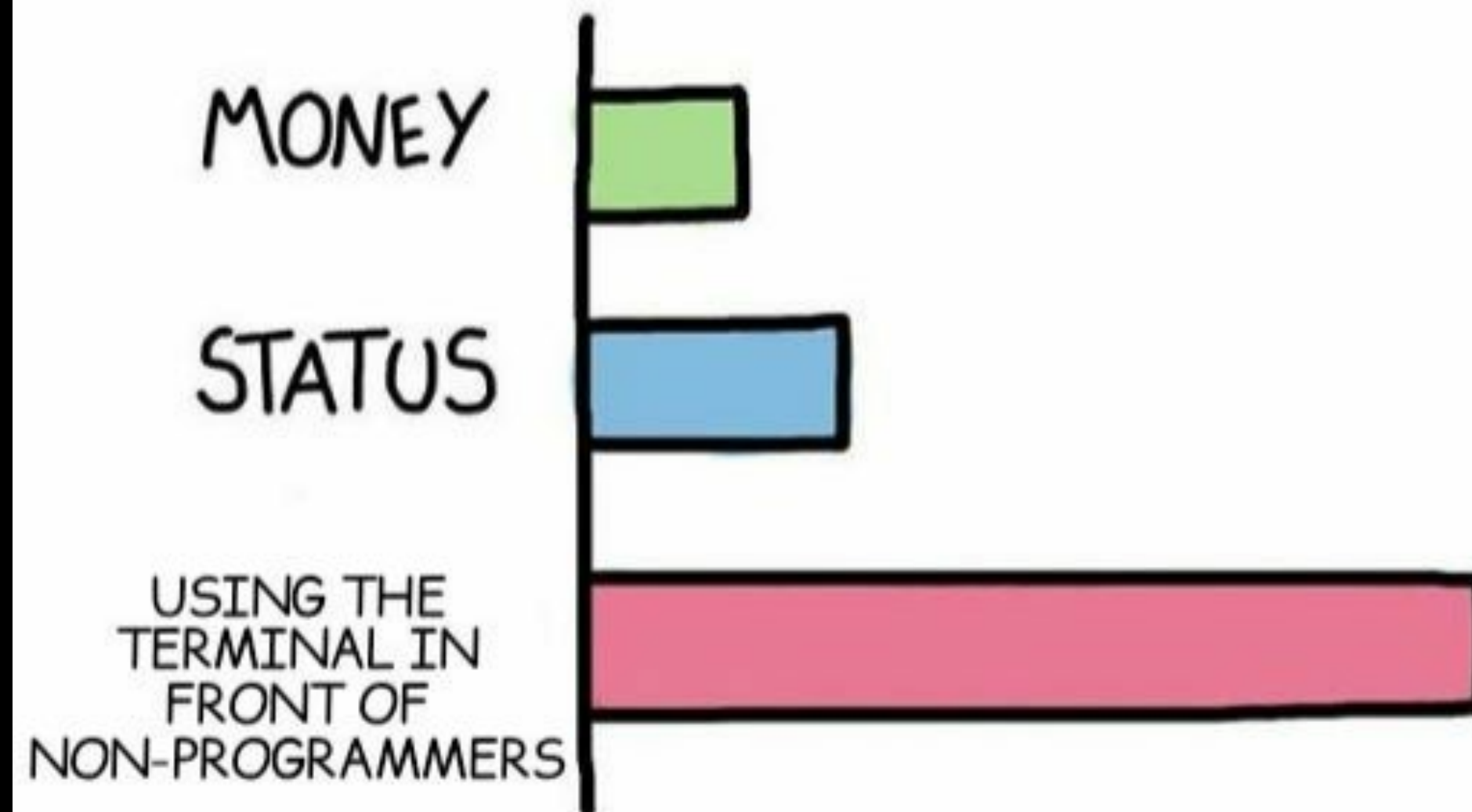
```
{
```

```
    std::cout << value << std::endl;
```

```
}
```

Coding Time

WHAT GIVES PEOPLE FEELINGS OF POWER



@iamnotanartist

