

# Evolution of Programming Abstraction Mechanisms: C-style Stack Implementations

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**



**Professor of Computer Science**

**Institute for Software  
Integrated Systems**

**Vanderbilt University  
Nashville, Tennessee, USA**



# Learning Objectives in This Part of the Lesson

- The rest of this overview examines several alternative methods of implementing a Stack
  - We'll begin with C & evolve up to various C++ implementations



See [en.wikipedia.org/wiki/Stack \(abstract data type\)](https://en.wikipedia.org/wiki/Stack_(abstract_data_type))

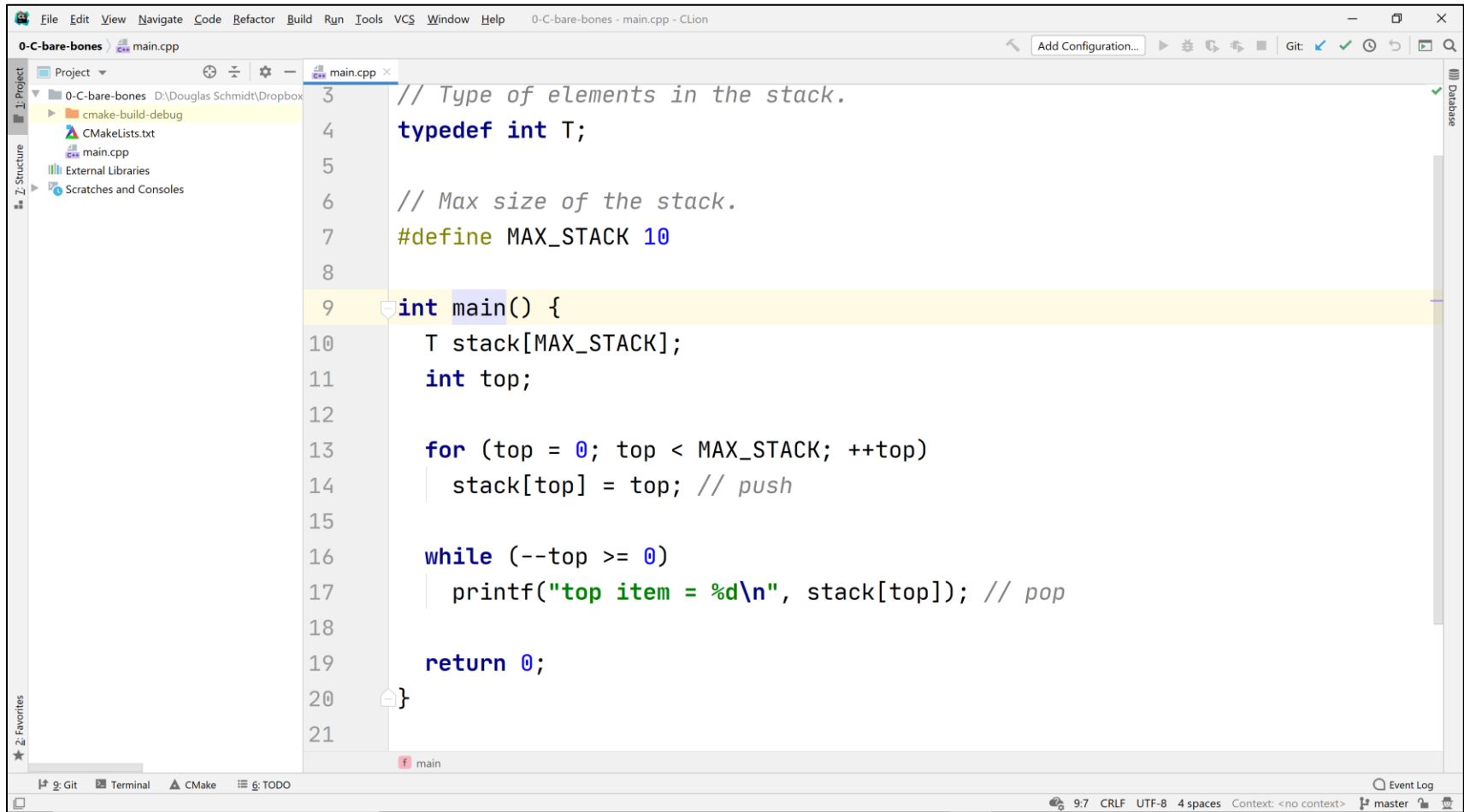
---

# C-style Stack Implementations

---

# "Bare-Bones" C Stack Example

- First, consider the "bare-bones" C implementation:



```
3 // Type of elements in the stack.
4 typedef int T;
5
6 // Max size of the stack.
7 #define MAX_STACK 10
8
9 int main() {
10     T stack[MAX_STACK];
11     int top;
12
13     for (top = 0; top < MAX_STACK; ++top)
14         stack[top] = top; // push
15
16     while (--top >= 0)
17         printf("top item = %d\n", stack[top]); // pop
18
19     return 0;
20 }
21
```

See [CPlusPlus/tree/master/overview/capabilities/0-C-bare-bones](https://github.com/DouglasC-Schmidt/CPlusPlus/tree/master/overview/capabilities/0-C-bare-bones)

# Pros of "Bare-Bones" C Stack Example

- Highly "efficient," i.e., no function call overhead!



# Cons of "Bare-Bones" C Stack Example

- It's not very abstract, so small mistakes can cause big problems!

