The Callstack

Contents

- Memory Overview
- What is the Callstack?
- What goes on the callstack
- Stack overflows

Memory Overview

- All the data your program uses is stored inside your computer's memory.
- Every single variable you ever make has space in RAM.
- Even the machine instructions themselves are stored in RAM
- You can think of memory as a giant list of bytes.

Memory Overview

- We keep track of where we store values by how far into the giant list they are.

- Just like an index into an array is how many elements from the start a value is, a byte's **memory address** is how many bytes from the first byte of memory it is.

Memory Overview

- All the memory on a computer is controlled by the operating system. Your program has to ask for all the memory it gets access to.

- There are two ways your program gets memory from the OS.
  - The Stack
  - The Heap.

- We'll be talking about the heap in a later lecture.

The Stack

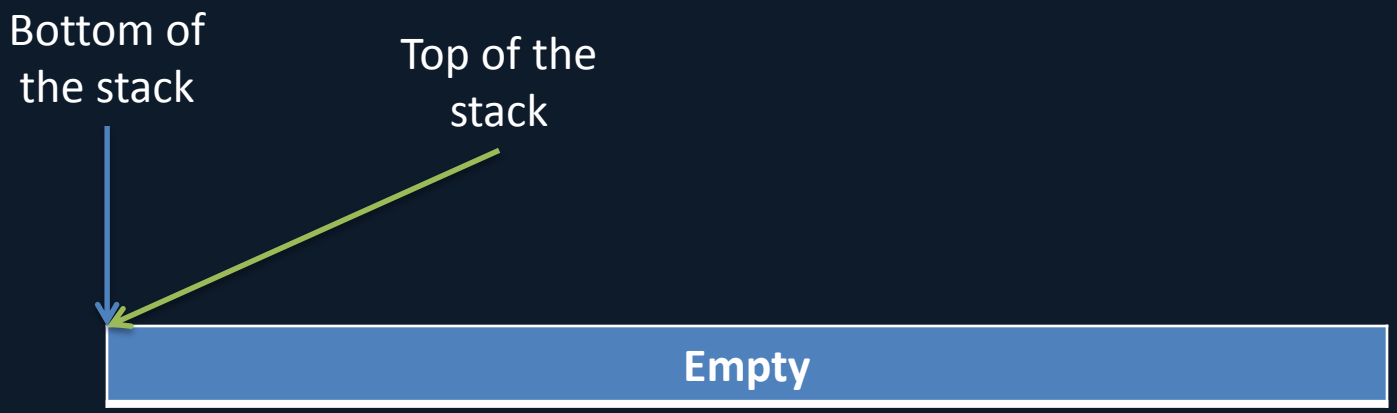
- When your program starts, a small amount of memory is allocated from the operating system.
  - This is the stack.
  - By default in VS, this is 1MB
- The stack is where every variable we've used so far has been stored.
- The stack also stores the list of functions that were called to get to the current point in the code.

What is a Stack

- We'll talk much more about stacks in a later lecture.
- A stack is an array that you add and remove from.
- It has a maximum size, and you add items (called push) and remove items (called pop) on to and off of the current end.

What is a Stack

- A stack grows and shrinks over time.

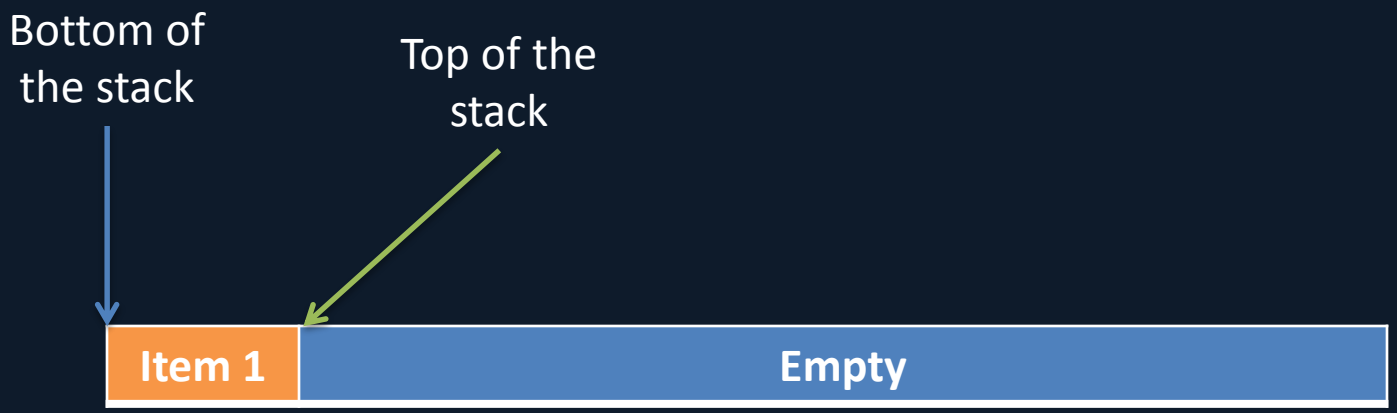


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

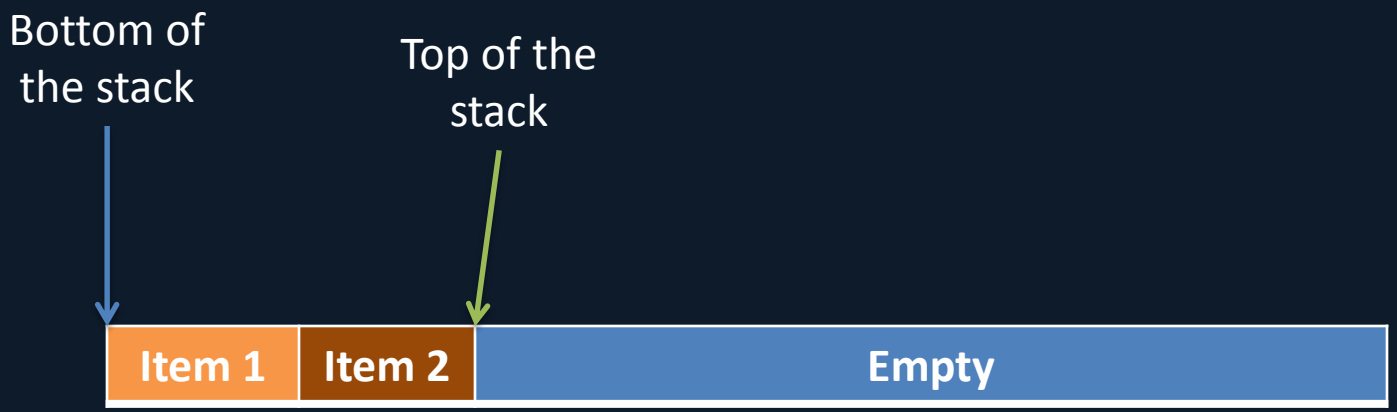
What is a Stack

- A stack grows and shrinks over time.



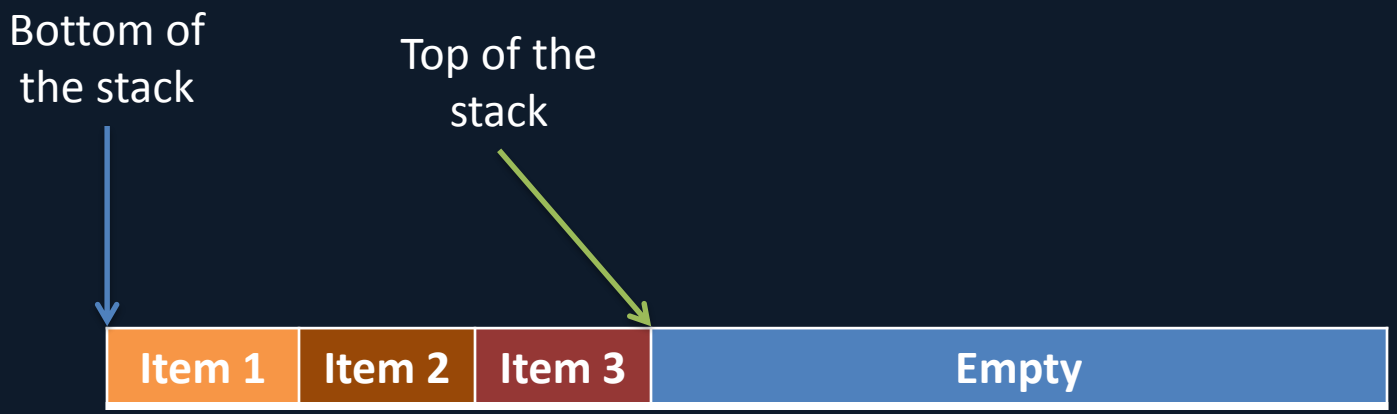
What is a Stack

- A stack grows and shrinks over time.



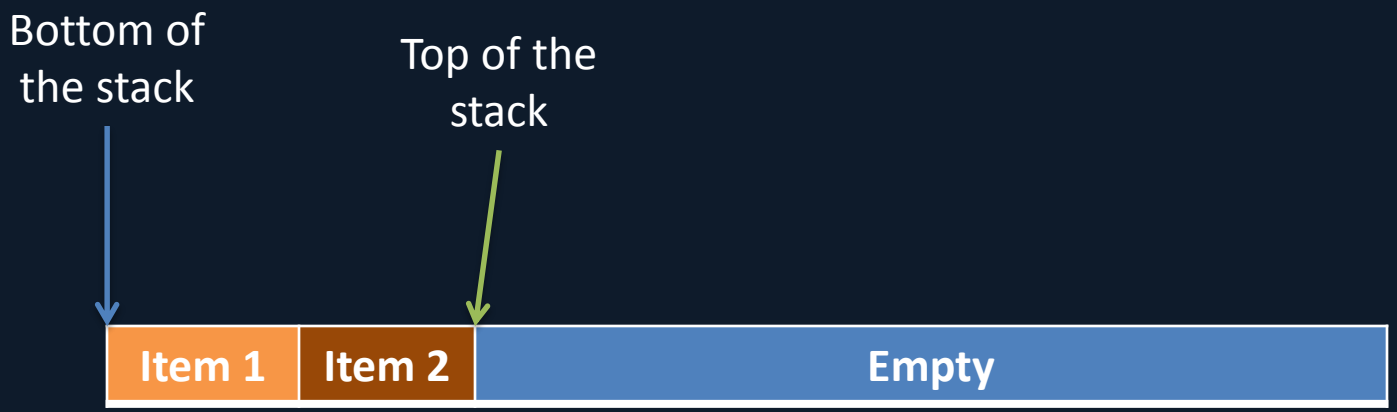
What is a Stack

- A stack grows and shrinks over time.



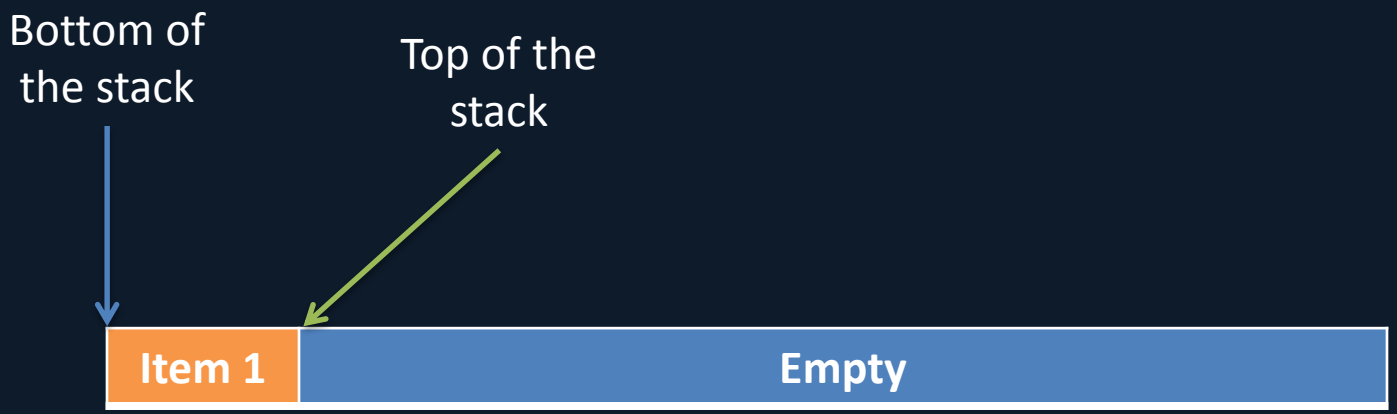
What is a Stack

- A stack grows and shrinks over time.



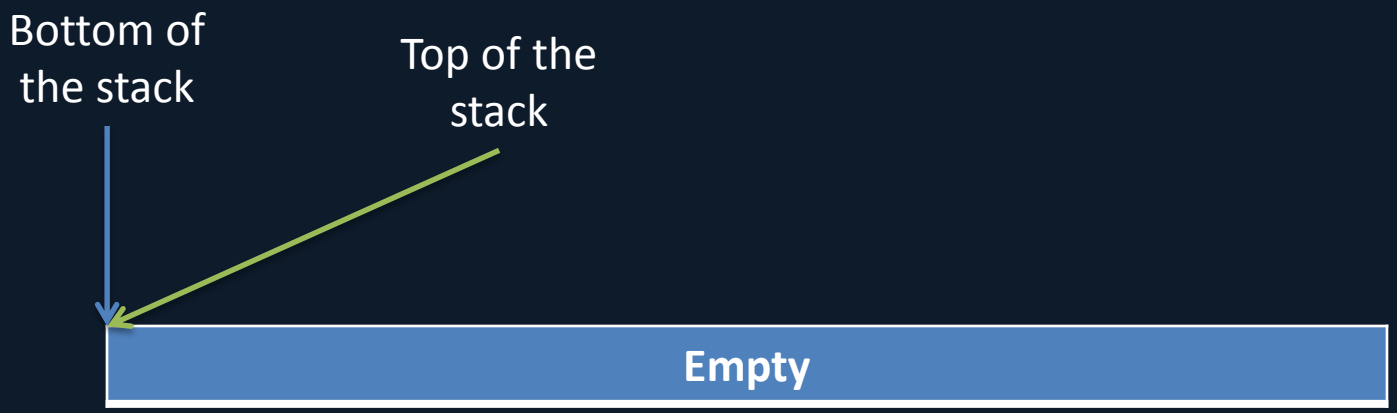
What is a Stack

- A stack grows and shrinks over time.



What is a Stack

- A stack grows and shrinks over time.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

The Stack

- The Stack is central to how functions work.
- When a function is called, it adds some data to the top of the stack. This data is called a stack frame.
- When the function returns, the stack frame for that function is removed from the stack.

The Stack

- The exact functionality of the stack depends on what compiler and compiler settings you are using.

- In general a stack frame contains at least:
  - The memory address of the instruction to go back to when the function exits
  - All the local variables in the function.

Stack Example

- Consider this code:

```
float UpdateCoord(float coord_value, float coord_speed)
{
    float result = 0;
    result = coord_value + coord_speed;
    return result;
}

void MoveObject(float x_coord, float y_coord, float x_speed, float y_speed)
{
    x_coord = UpdateCoord(x_coord, x_speed);
    y_coord = UpdateCoord(y_coord, y_speed);
}

int main()
{
    float player_x = 10;
    float player_y = 19;
    float player_speed_x = 7;
    float player_speed_y = 4;

    MoveObject(player_x, player_y, player_speed_x, player_speed_y);

    return 0;
}
```

Stack in Action

- When your program starts, its 1MB of stack is allocated and all of the memory is empty.

```
...
int main()
{
    float player_x = 10;
...
```

...	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	...

Stack in Action

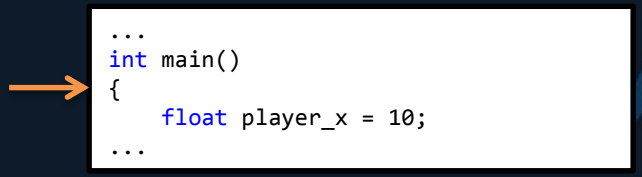
- The OS calls some functions before calling the main function, their stack frames are put on the stack.

```
...
int main()
{
    float player_x = 10;
...
```

...	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	...

Stack in Action

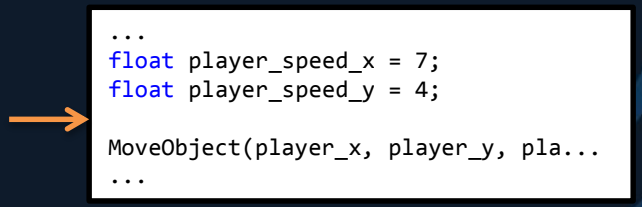
- Once main is called, its stack frame is added to the stack.



- This includes two pieces of data
  - The address of the instruction that main has to return to one it has finished.
  - The local variables of main
    - player_x, player_y, player_speed_x, player_speed_y

Stack in Action

- Now the actual code of main starts executing.
- When main gets up to a function call, we need to do some more work with the stack.



Canberra Institute of Technology

ACE
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Stack in Action

- When MoveObject gets called, its stack frame is put on the stack.

```
...
void MoveObject(float x_coord, float y_...
{
    x_coord = UpdateCoord(x_coord, x_s...
...
```

- Just like with main, we have two things inside the stack frame:
  - The address of the line in main to return to when we finish the function
  - Space for the local variables. In this case, the arguments for the function.

Stack in Action

- When MoveObject gets called, its stack frame is put on the stack.

```
...
void MoveObject(float x_coord, float y_...
{
    x_coord = UpdateCoord(x_coord, x_s...
...
```

- Just like with main, we have two things inside the stack frame:
  - The address of the line in main to return to when we finish the function
  - Space for the local variables. In this case, the arguments for the function.
  - The values of the variables for player_x and player_y are copied into the newly created variables x_coord and y_coord. The same is done for the speed variables.

Stack in Action

- Now the first call to UpdateCoord is called from the first line of MoveObject

```
...
float UpdateCoord(float coord_value, fl...
{
    float result = 0;
...
```

- Once again, the stack frame is pushed.

Stack in Action

- Once UpdateCoord gets to the last line, it uses the return address to know where to pick up from in the MoveObject function.

```
...
    result = coord_value + coord_speed;
    return result;
}
...
```

Canberra Institute of Technology

ACE
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Stack in Action

- The value of the result variable is used to fill in the x_coord variable

```
...
    x_coord = UpdateCoord(x_coord, x_s...
    y_coord = UpdateCoord(y_coord, y_s...
}
...
```

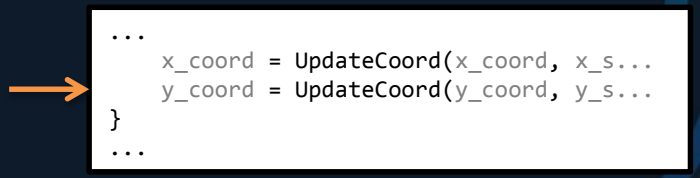
- Now that UpdateCoord has returned, its stack frame is removed.
  - All of the variables inside UpdateCoord have now been destroyed.

Canberra Institute of Technology

ACE
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Stack in Action

- The next line is just calling UpdateCoord again, so the new stack frame is pushed onto the stack.



- Its important to note here, that because different values were passed as arguments, different values are copied into the new stack frame.

Stack in Action

- UpdateCoord then runs, exactly the same as how we saw it before.
- Its result value is copied into y_coord and its stack frame is then destroyed.
- This is why variables go out of scope when a function ends.

```
...
    x_coord = UpdateCoord(x_coord, x_s...
    y_coord = UpdateCoord(y_coord, y_s...
}
...
```

Canberra Institute of Technology

ACE
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Stack in Action

- MoveObject is now complete.

- As it doesn't return anything, its stack frame is simply deleted.

```
...
    MoveObject(player_x, player_y, pl...
    return 0;
}
...
```

Stack in Action

- Now main has finished.

```
...
    MoveObject(player_x, player_y, pl...
    return 0;
}
...
```

- Its stack frame is deleted and control of the computer goes back to the OS.

...	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	...

Stack Overflows

- The stack has a maximum size.
- When you call a function, its stack frame goes onto the stack.
- If you put too much data on the stack it will cause a stack overflow and your program will crash.

Summary

- Memory is where all of the data for your program, including the instructions themselves, is stored.
- All memory is controlled by the operating system.
- When your program starts it gets a small amount of memory for local variables called the stack.
- Calling a function adds data to the stack for its arguments and local variables
- Returning from a function removes data from the stack, taking the variables out of scope and making them inaccessible.

References


Canberra Institute
of Technology


aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX