A Mental Model for Computation in C++

Christian DeLozier
University of Pennsylvania
January 26th, 2015
Why do we need a mental model for C++?

- Most languages abstract away memory

```java
public void JavaFunction()
{
    int x = 1;
    Integer y = 2;

    y = x;
}
```

- C and C++ do not abstract away memory!
How do we picture a program’s memory?

- Assume your program has all of the memory
  - For now, let’s assume 32 bits for pointers

<table>
<thead>
<tr>
<th>Addresses</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000000000</td>
<td>0x00</td>
</tr>
<tr>
<td>0x0000000001</td>
<td>0xDE</td>
</tr>
<tr>
<td>0x0000000002</td>
<td>0xAD</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0xffffffffffe</td>
<td>0xBE</td>
</tr>
<tr>
<td>0xffffffffffffff</td>
<td>0xEF</td>
</tr>
</tbody>
</table>
How is this memory organized?

- Regions of data: globals, stack, and the heap
- Globals: Global variables, static variables, constants
How is this memory organized?

• Stack: Holds function data
  • Local variables, return values, return addresses
  • Automatically allocated and deallocated at each call

Memory

0x00000000

Stack
(0xFFFFFFFF - 0xFFFF0000)

Globals
(0x00020000 – 0x00040000)
How is this memory organized?

- **Heap**: Holds explicitly allocated data
  - e.g. “new” in Java and C++, “malloc” in C
  - In C++, explicitly deallocated by the programmer

```
Memory
 Globals (0x00020000 – 0x00040000)
 Heap (0x00060000 – 0x00F00000)
 Stack (0xFFFF0000 – 0xFFFF0000)
```

```
0x00000000
0xFFFFFFFF
```

A Real Program’s Stack

```c
int f(int a, int b) {
    return a + b + g(a);
}

int g(int c) {
    int d = 5;
    return c + d;
}

int main() {
    int x = 1;
    int y = 2;
    int z = f(x, y);
    return 0;
}
```

- **g**
  - c [1]
  - d [5]

- **f**
  - a [1]
  - b [2]
  - g(a) [6]

- **main**
  - x [1]
  - y [2]
  - z [9]
  - f(x,y) [9]