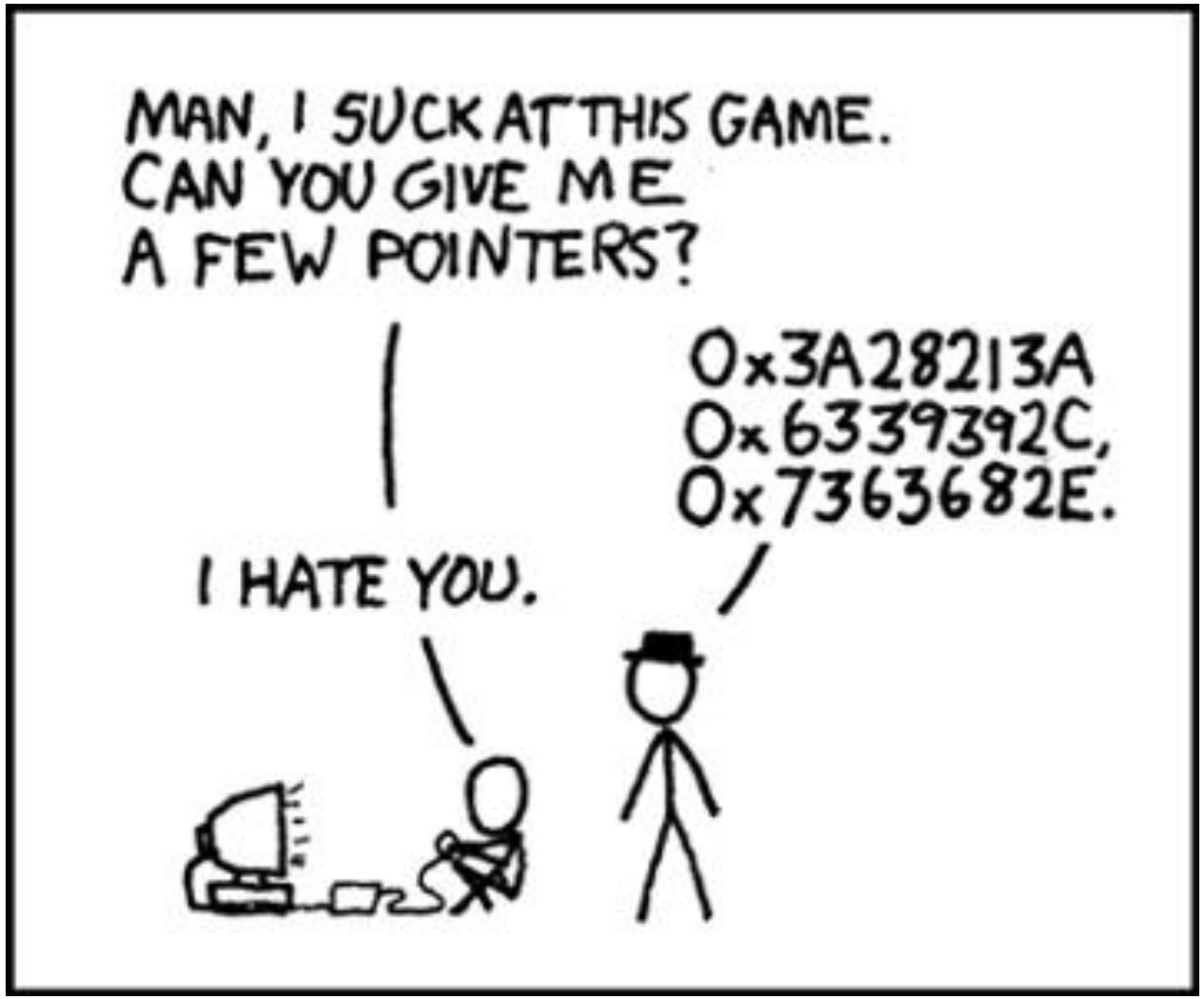
CS 161, Lecture 15: Pointers and Memory Model



Warm-Up

Function: increments_by

Description: increments num1 by the value num2

Input: num1, num2

Return type: void

	Pass By Value	Pass By Reference	Pass By Pointer
Parameter Listing			
What is actually being passed?			
Does the function body need to change? If so, how?			
Function Call			
How will things change from where the function was called?			

More Pointers

- Declaring pointers

```
int *i; //This will hold an int memory address
int j; //This holds an int
i = &j; //set the address that i holds to the address of j
j = 2; //sets the value of j to 2, *i is 2
(*i)++; //increments the value i points to, j is 3
```

Demo

What if we don’t have an address to point to?

- We make one with the **new** keyword (dynamically allocate)

```
int *p;
p = new int; //new returns an address
*p = 10;
```

Demo

Different Types of Memory

- CPU: central processing unit, “brain” of the computer system
- Main memory
  - where current programs are executed
  - CPU has direct and quick address
  - Volatile: contents are lost when the power goes out
- Secondary Memory
  - Nonvolatile, long term storage
  - Ex. Files, hard drive, USB, etc.

How Main Memory is Structured

- Stack
  - Variables defined at compile time go on the stack (global variables, constants)
  - Functions have their own stack frame
  - When a function ends, the stack frame collapses and cleans up the memory for you -> sometimes referred to as automatic variables
- Heap
  - Variables defined at runtime (**new** keyword)
  - Variables declared dynamically in a function do not disappear when the function ends as they are on the heap and not the function stack
  - Can run out of heap space
  - Need to free dynamic memory when done with it, otherwise memory leaks

Static vs Dynamic

- Static

```
int *i, j=2;
i = &j;
```

- Dynamic

```
int *i = NULL;
i = new int;
*i = 2;
```

Fixing Memory Leaks

- How to tell you may have a memory leak
  - You used the **new** keyword
  - You never used the **delete** keyword
  - You run valgrind
    - Compile and produce an executable for your program
    - Run valgrind with your executable (valgrind executable_name)
- How to fix memory leaks
  - Delete dynamic memory when you are done with it

```
int *i = new int;
delete i;
```

Demo

Feedback

https://tinyurl.com/y7c79hap