CS 354 - Machine Organization & Programming
Tuesday, September 10, 2019

Waitlisted? Complete the form at: https://forms.gle/CRvL1oR8i9Bymvyo6

Jim Skrentny, 5379 CS, skrentny@cs.wisc.edu
Course website: https://canvas.wisc.edu/courses/154937

Project p1 (3%): DUE at 10 pm on Monday, September 23rd

Exam Conflicts: report any by this Friday using form at: https://forms.gle/6TwXssFmUCh7o8GS8
TA lab consulting & PM drop-in hours: are scheduled, see links on course front page
Linux Workshop: tonight 5:30 pm 1240 CS repeated on Friday 5:30 pm 1240 CS

Last Time

- Course Info and Coursework
- Java vs. C
- Coding in C Remotely
  - Get Connected to CS
  - Edit your Source
  - Compile/Run/Debug

Today

- C Program Structure
- C Logical Control Flow
- Recall Variables
- Meet Pointers

Next Time

- Pointers: Arguments and 1D Arrays
- **Read:**
  - K&R Ch. 5.1: Pointers and Addresses
  - K&R Ch. 5.2: Pointers and Function Arguments
  - K&R Ch. 5.3: Pointers and Arrays
  - K&R Ch. 5.4: Address Arithmetic
- **See:** Piazza post for web alternatives to K&R readings

C Program Structure

✵ *Variables and functions must be declared before they're used.*

➢ What is output by the following code?

```
#include <stdio.h>

int bing(int x) {
   x = x + 3;
   printf("bing %d\n", x);
   return x - 1;
}

int bang(int x) {
   x = x + 2;
   x = bing(x);
   printf("BanG %d\n", x);
   return x - 2;
}

int main(void) {
   int x = 1;
   bang(x);
   printf("BOOM %d\n", x);

   return 0;
}
```

Passing Arguments

argument:

parameter:

pass-by-value:

Return Value

return-by-value:

C Logical Control Flow

Sequencing

Selection

→ Which value(s) means true?   **`true`**   42   -17   0

if - else

→ What is output by this code when `money` is 11, -11, 0?

```
if (money = 0)      printf("you're broke\n");
else if (money < 0) printf("you're in debt\n");
else                printf("you've got money\n");
```

→ What is output by this code when it's 2/14? 11/31?

```
if (         month)
   if (         day)
      printf("Happy Halloween!\n");
else
   printf("It's not October.\n");
```

switch

C Logical Control Flow (cont.)

Repetition

```
int i = 0;
while (i < 11) {
   printf("%i\n", i);
   i++;
}
```

```
for (int j = 0; j < 11; j++) {
   printf("%i\n", j);
}
```

```
int k = 0;
do {
   printf("%i\n", k);
   k++;
} while (k < 11);
```

Recall Variables

What? A *scalar variable* is

→ Draw a basic memory diagram for the variable in the following code:

```
void someFunction(){
   int i = 44;
```

Aspects of a Variable

identifier:

value:

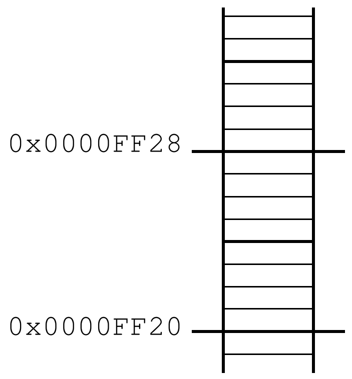
type:

address:

size:

Linear Memory Diagram

A linear memory diagram is



	most significant							least
base 10	44							
base 2	0000	0000	0000	0000	0000	0000	0010	1100
base 16	0	0	0	0	0	0	2	C

byte addressability:

endianess:

little endian:

big endian:

Meet Pointers

What? A *pointer* variable is

- ◆
- ◆

Why?

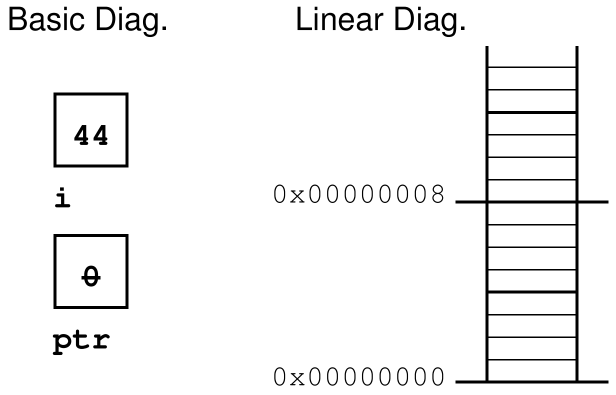
- ◆
- ◆
- ◆
- ◆

How?

→ Consider the following code:

```
void someFunction(){
   int i = 44;

   int *ptr = NULL;
```



→ What is ptr's initial value? Address? Type? Size?

pointer:

pointee:

& *address of:*

***** *dereferencing:*