# C Primer #2

Thursday, Sept. 15th, 2016

UNIVERSITY OF MINNESOTA

Driven to Discover℠

# l-value vs. r-value

- In the assignment expression **E1 = E2**, the left operand **E1** must be an lvalue expression, the right operand can be any expression.
- An lvalue (locator value) represents an object that occupies some identifiable location in memory (i.e. has an address).
  - x = x + 7;  /* x is used as both an l-value and r-value in this statement */
  - int n, *p;

    p = &n; // OK

    &n = p;  // error: &n is an rvalue

# Pointers

- **Variable**: represents a memory location that stores data
  - size varies according to the data type
- **Pointer**: a variable whose value is a memory address location
  - size of a pointer is always the same on a system:
    - 32 bit machines → 4 bytes
    - 64 bit machines → 8 bytes

# Pointers cont.

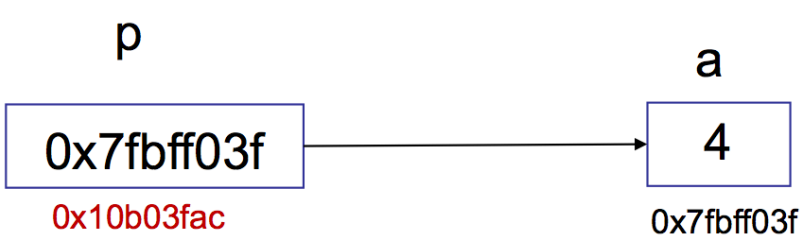
- Declaring a pointer:
  - int *p;
  - int * swap(int *num1, int *num2);
- **&** Address Operator: &<variable>
- * Dereferencing Operator: *<pointer>

# Pointers cont.

- *pointer_basic.c*

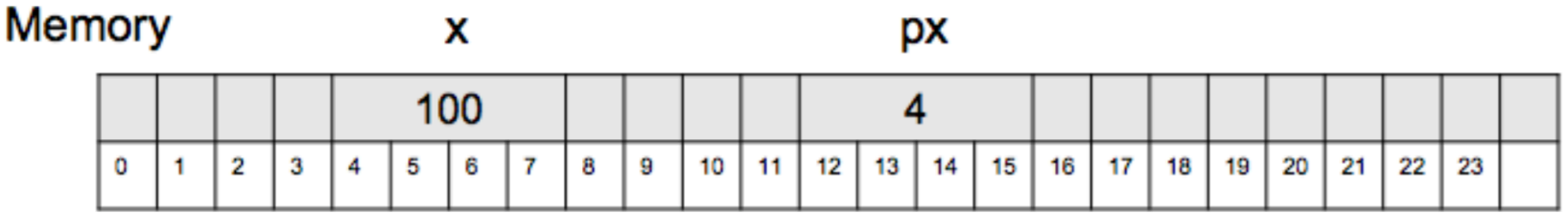
```
int a=4;
int *p;

p=&a;
```




- ‘a’ variable stores value = 4.
- We declare a pointer variable ‘p’.
- Now, using ‘&a’ --pointer variable ‘p’ points to variable ‘a’.
- And stores memory address of variable ‘a’ = 0x7fbff03f.
- De-referencing pointer variable ‘p’ will yield 4 i.e. *p value is 4.
- Yes ! ‘p’ will also have its own memory address as well = 0x10b03fac.

# Pointers cont.




```
int x;
int *px;

x = 100;
px = &x;
```

```
&x = ?
&px = ?

x = ?
px = ?
```

```
*x = ?
*px = ?
```

# scanf

```
 3 int main(void)
 4 {
 5     int a;
 6
 7     printf("Please input an integer value: ");
 8     scanf("%d", &a);
 9     printf("You entered: %d\n", a);
10
11     return 0;
12 }
```

*rec_scanf.c*

# Pointers Arithmetic

- We can perform an addition/subtraction on a pointer:
  - p = p + α → p = p + (α * size of the pointed data type)

```
int *p;      // p = 0x100     double *q;   // q = 0x100
p = p + 2; // p = 0x108       q = q + 2;   // q = 0x110
p = p − 1; // p = 0x104       q = q − 1;   // q = 0x108
```

*pointer_arithmetics.c*

# Parameter Passing

- **Pass by Value**
  - The local parameters are copies from the values that are passed in
  - Changes made to the parameters in the function do not affect the originals
- **Pass by Reference**
  - The local parameters are references to the values that are passed in
  - Changes made to the parameters in the function will affect the originals

*param.c*

# Practice 2

- Write a short C program that swaps two pointers' values.

# BigNum Lab

- (Go over the handout)

# QUESTIONS?