SYSC 2006
Fall 2019

Carleton
UNIVERSITY

Canada's Capital University

C: Pointers

D.L. Bailey, Systems and Computer Engineering, Carleton University

- Last revised (C.-H. Lung) Oct 2nd, 2019

Memory Organization

- Memory can be viewed as a collection of consecutively-numbered cells
  - usually, each cell holds an 8-bit **byte**
  - a *char* is stored in one cell (byte)
  - a 32-bit *int* is stored in 4 adjacent cells
    - on smaller machines, values of type *int* are 16 bits wide and are stored in pairs of adjacent cells
- The cell numbers are known as *addresses*

Variables & Pointers

- A variable:

  a symbolic name for a group of cells

  - # of cells depends on the variable's type
  - It contains an actual value

- Pointer:

  a variable that contains the **address** of a variable

Pointer Variable Declarations

```
int *p;
```

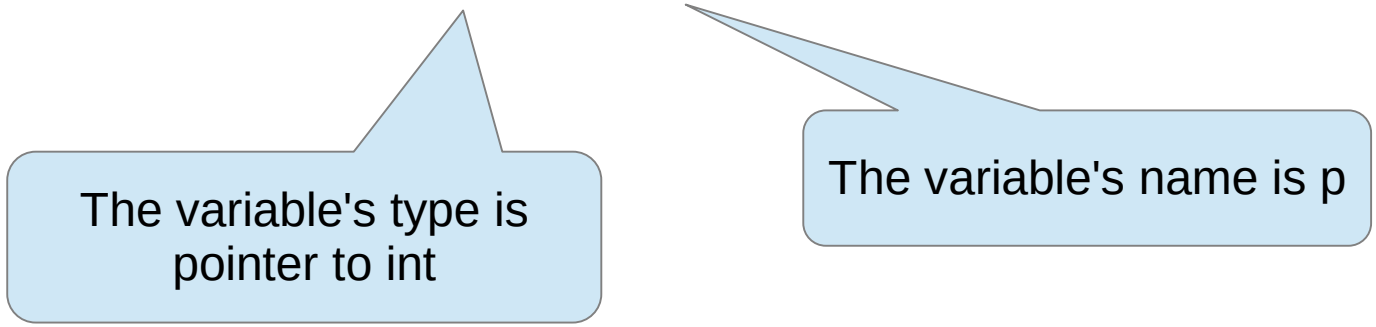
- Declares a variable named `p`, whose type is "pointer to `int`"
  - *`type`* `*` means "**pointer to *`type`***"

- Common error: thinking that this statement declares a variable named `*p`, whose type is `int`

Pointer Variable Declarations

- Some programmers prefer to place the * beside the type identifier instead of the variable name; e.g.,

```
int* p;
```

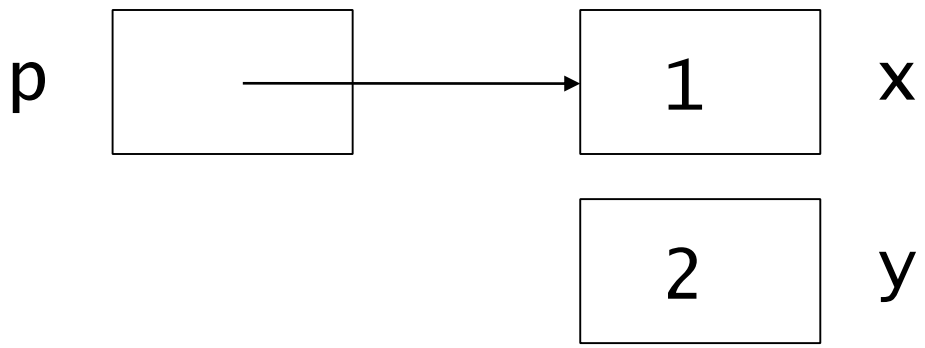


Address-of (&) Operator

- Unary operator **&** yields the address of its operand

```
int *p;
int x = 1, y = 2;
p = &x;
```

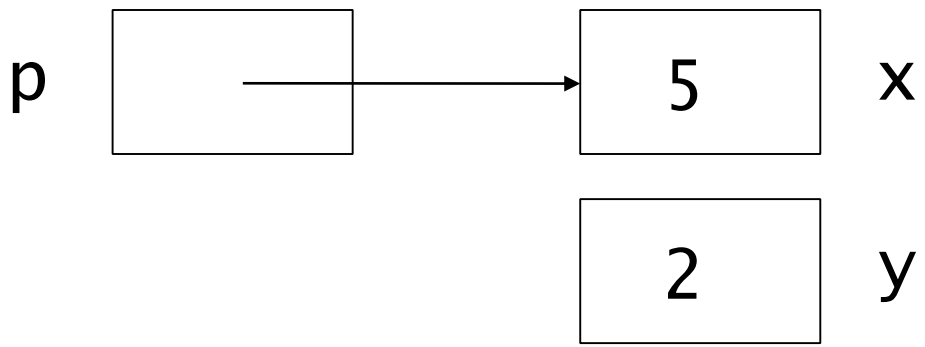
- This assigns the address of x to variable p
  - p now "points to" x



Dereferencing (*) Operator

```
*p = 5;
```

- the variable pointed to by p is assigned 5
- x now contains 5



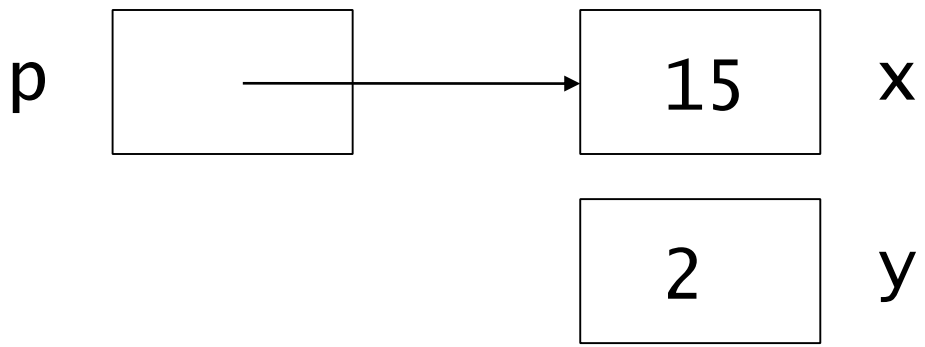
Dereferencing (*) Operator

- If p points to x, then *p can be used anywhere x can be used

  `*p = *p + 10;` is **equivalent** to

  `x = x + 10;`

  – x now contains 15



Precedence of Operators

- Unary **&** and ***** have higher precedence than the arithmetic, comparison and logical operators

```
y = *p + 1;
```

  - dereferencing the pointer is performed before the addition
  - takes the value pointed to by p, adds 1, then assigns the result (16) to y

- This statement means something completely different:

```
y = *(p + 1);
```

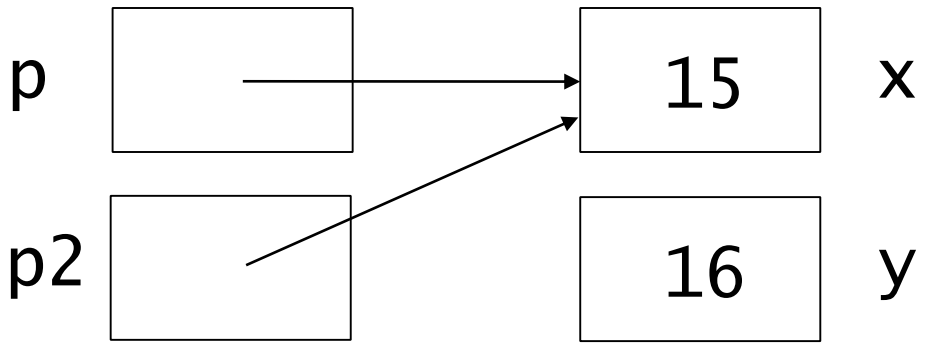
  - we'll learn what it means, later

Pointer Assignment

- A pointer can be assigned to another pointer:

```
int *p2;
p2 = p;
```

  - copies the contents of p into p2
  - p2 and p now point to the same variable



Pointer Assignment

- zyBooks (optional): Section 8.2 Pointer basics
- C Tutor