Intro to Comp Eng ECSE-221

Tutor: John Harrison

Tutorials:

Monday 5:35-6:25pm WONG1020
Wednesday 3:35-4:25pm ENGTR0070

Office Hours:

Monday 4:35-5:25pm
Wednesday 4:35-5:25pm

Tutorials 1 and 2:
Intro to "C" programming

- Comparing Java and "C"
- Writing your first "C" program
- Data types
- Creating and using variables
- Operators
- Pointers
- Input / Output
- Debugging Example
- Baseconvert program: Java to "C"

Comparing Java and "C"

Java

- Object-oriented language
- Runs on virtual machine
- Slow, but portable

"C"

- Procedural language
- Runs on native processor
- Fast, but not as portable
- ("C++" is object-oriented version of "C")

Structure of a "C" program

```
#include <stdio.h>

void foo(int x, int y);
int bar(int x);

main(int argc, char *argv)
{
...
}

void foo(int x, int y)
{
...
}

int bar(int x)
{
   ...
   return x;
}
```

←insert contents of library file stdio.h

←forward declarations of procedures/functions

←main program

←procedure "foo"

←function "bar"

Writing your first "C" program

From WebCT:

- Get a copy of LCC (if necessary)
- Refer to "Learning C from Java"

Demonstration :

- Creating a new project in LCC

  (note: no spaces in path!)
- Entering a simple "C" program
- Running and debugging program

Hello.c

```
#include <stdio.h>

int main()
{
  printf("Hello World\n");
}
```

Basic "C" Data Types

Simple types

- **`char x;`** integer (1 byte)
- **`int x;`** integer (2 or 4 bytes)
- **`long x;`** long integer (4 bytes)

Can also be unsigned:

- **`unsigned char x;`**
- **`unsigned int x;`**

Floating point:

- **`float x;`** single-precision floating point
- **`double x;`** double-precision floating point

Creating and Using Variables

- variable declaration (start of procedure)

```
int x, y;
char c;
```

- assigning a value to a variable (in procedure)

```
x = 42;
y = x + 7;
x += 3;
x -= 5;
```

(same as `x = x + 3`)

(same as `x = x - 3`)

Basic "C" Operators

+ - * / %	add, subtract, multiply, divide, remainder
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equal to
!=	not equal to

Increment/Decrement Operators

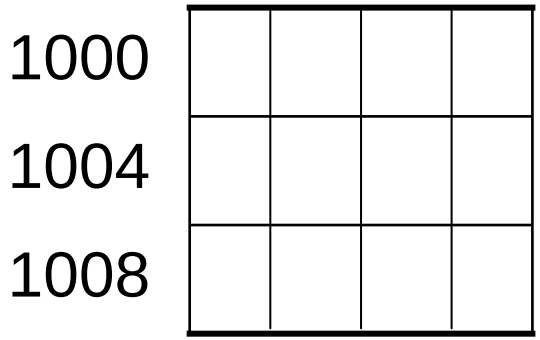
x++	use x, then increment
++x	increment x, then use it
x--	use x, then decrement
--x	decrement x, then use it

Examples:

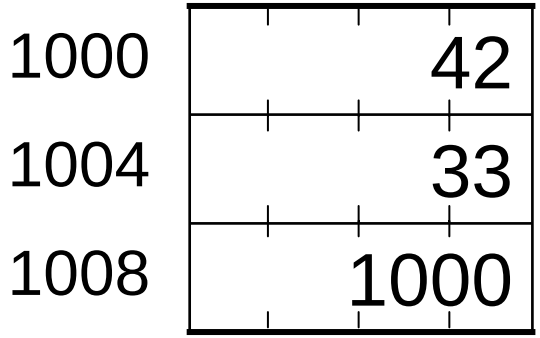
`x = 12;`
`y = x++;`
 	
y: 12
x: 13
	`x = 12;`
	`y = ++x;`
	
	y: 13
	x: 13
		`x = 12;`
		`y = x--;`
		
		x: 11
		y: 12
			`x = 12;`
			`y = --x;`
			
			x: 11
			y: 11

Intro to Pointers

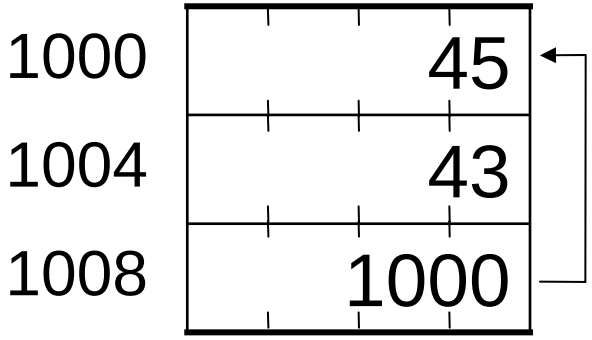
```
int x, y;
int *ptr;
```




```
x = 42;
y = 33;
ptr = &x;
```

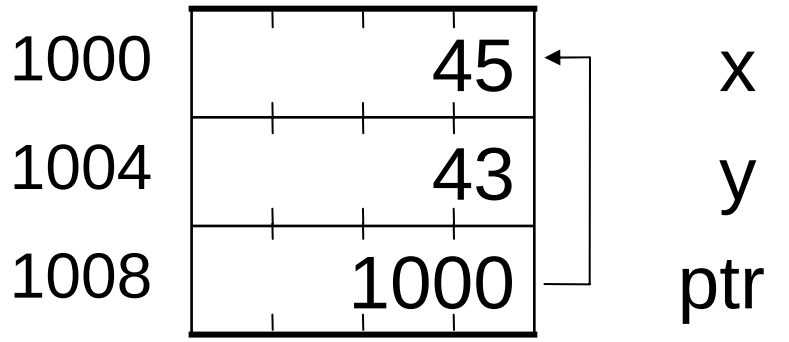



```
y = y + 10;
*ptr = *ptr + 3;
```

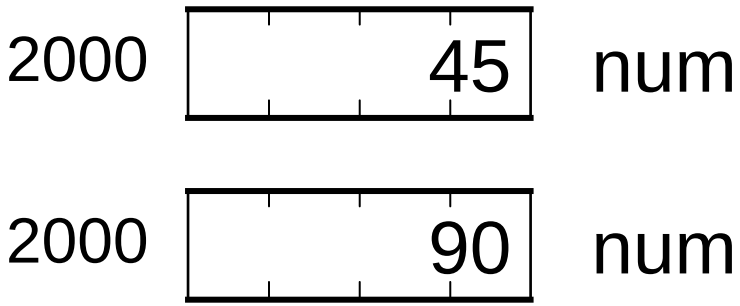


Passing Variables

```
twotimesBAD(x);
```

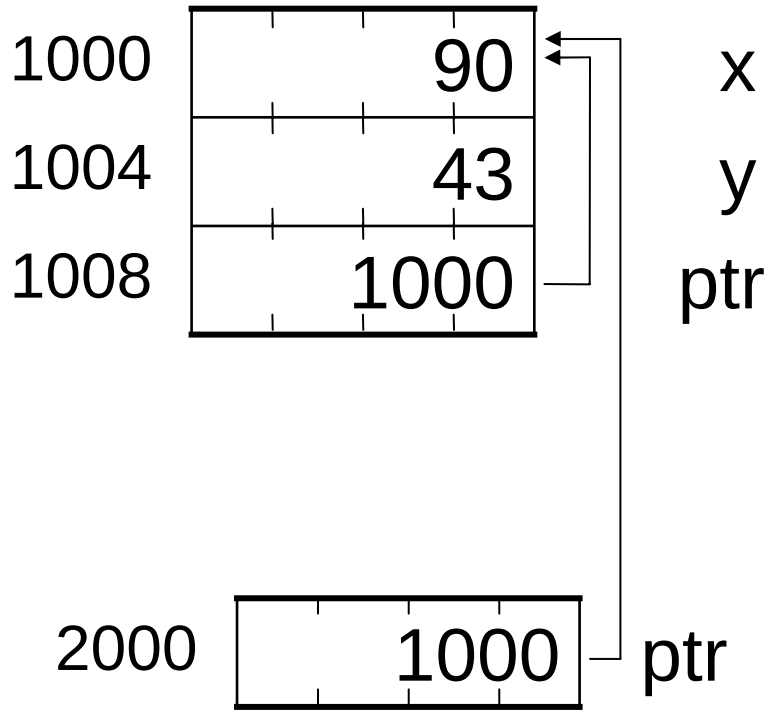



```
void twotimesBAD (int num)
{
  num = num + num;
}
```



Passing Variables (2)

```
twotimesGOOD(&x);
```




```
void twotimesGOOD(int *ptr)
{
  *ptr = *ptr + *ptr;
}
```

Arrays

Can make an array of any of the basic types:

- **`char x[3];`** an array of 3 bytes
- **`int x[5][11];`** a 5x11 array of integers

Arrays are indexed starting at zero. E.g.

```
float x[3];
x[0] = 23.4;
x[1] = 3.223;
x[2] = 32.333;
```

Using **`x[3]`** is incorrect, and *might* give an error

Strings

- Array of characters, ends with special character: '\0' (NULL)
- So, string arrays need one extra character.
- e.g. to store "This\n", need array of 6 characters:

```
char s[6];
s[0] = 'T';
s[1] = 'h';
s[2] = 'i';
s[3] = 's';
s[4] = '\n'
s[5] = '\0';
```

Output

`printf(<string>, <arg1>, <arg2>, ...);`

Special characters for <string>:

\n	newline
\\	single slash
%d	integer argument
%f	float argument
%c	character argument
%s	string (array of characters) argument

e.g.: **`printf("x is %d and y is %d", x, y);`**

output: **`x is 90 and y is 43`**

Input

`scanf(<string>, <address1>, <address2>, ...);`

Special characters for <string> (similar to printf):

%d	integer argument
%f	float argument
%c	character argument
%s	string (array of characters) argument

e.g.: **`scanf("%d %d", &x, &y);`**

- will read in values for x and y.

Input from a string

```
sscanf(<input_string>, <format_string>,
       <address1>, <address2>, ...);
```

Special characters for <format_string> (are the same as scanf).

e.g.

```
char str[] = "42 33";
sscanf(str,"%d %d", &x, &y);
```

- will set x to 42 and y to 33.

Example

- Debug
  - a buggy program
  - fixing compile errors
  - debugging

Example

- BaseConvert
  - Java example program
  - "C" equivalent