CS214-system programming

Section 03/08 recitation 01

https://rutgers.webex.com/meet/yg397
Office Hour: Weds 5:00-6:00 pm

Yunhe Gao
yg397@scarletmail.rutgers.edu

About Recitation

- Not a lecture. It's for you to refresh your memory and ask questions
- No attendance required
- If any questions, please email me before recitation
  - Include [cs214] in the subject

About CS 214

- Low-level “system” programming
- C programming
  - Basic syntax, data structure, pointer, dynamic memory management
- Programming under UNIX
  - Linux commands, File I/O, network programming, signal handling
- Build large-scale projects
  - Modules, headers, linking, makefile
- Concurrent programming
  - Multiple process, multiple threads, communication and synchronization

- Programming Heavy!

Grading

- 4 programming projects (50%),
- quizzes short homework (25%)
- Final exam (25%) (Sakai exam)

- Make sure your code can compile and run on iLab
- We grade your project on those iLab machines and if it doesn't compile, you will get a ZERO.
- Be careful of the due date, any late submission will not be accepted

Overview

- iLab
- Text Editor and command line
- Basic C Programminig

iLab

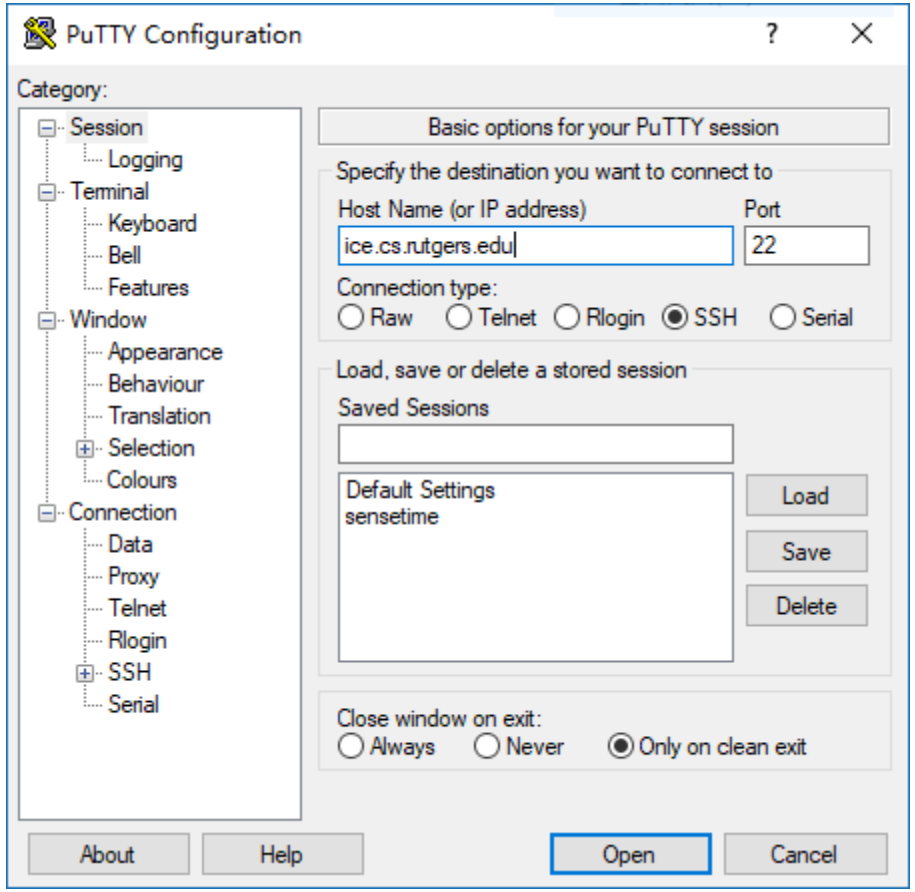
- Step1: Go to iLab Website: https://services.cs.rutgers.edu/accounts/
- Step2: Activate your account using NetID and set up a password
- Step3: Try to login any of the remote computer (using ssh)

  A list of computers could be found in:
  https://report.cs.rutgers.edu/nagiosnotes/iLab-machines.html

  (Use H248, H252, H254 machines, no need to always use the same one, all your files are stored in the file server and are shared among all iLab machines)

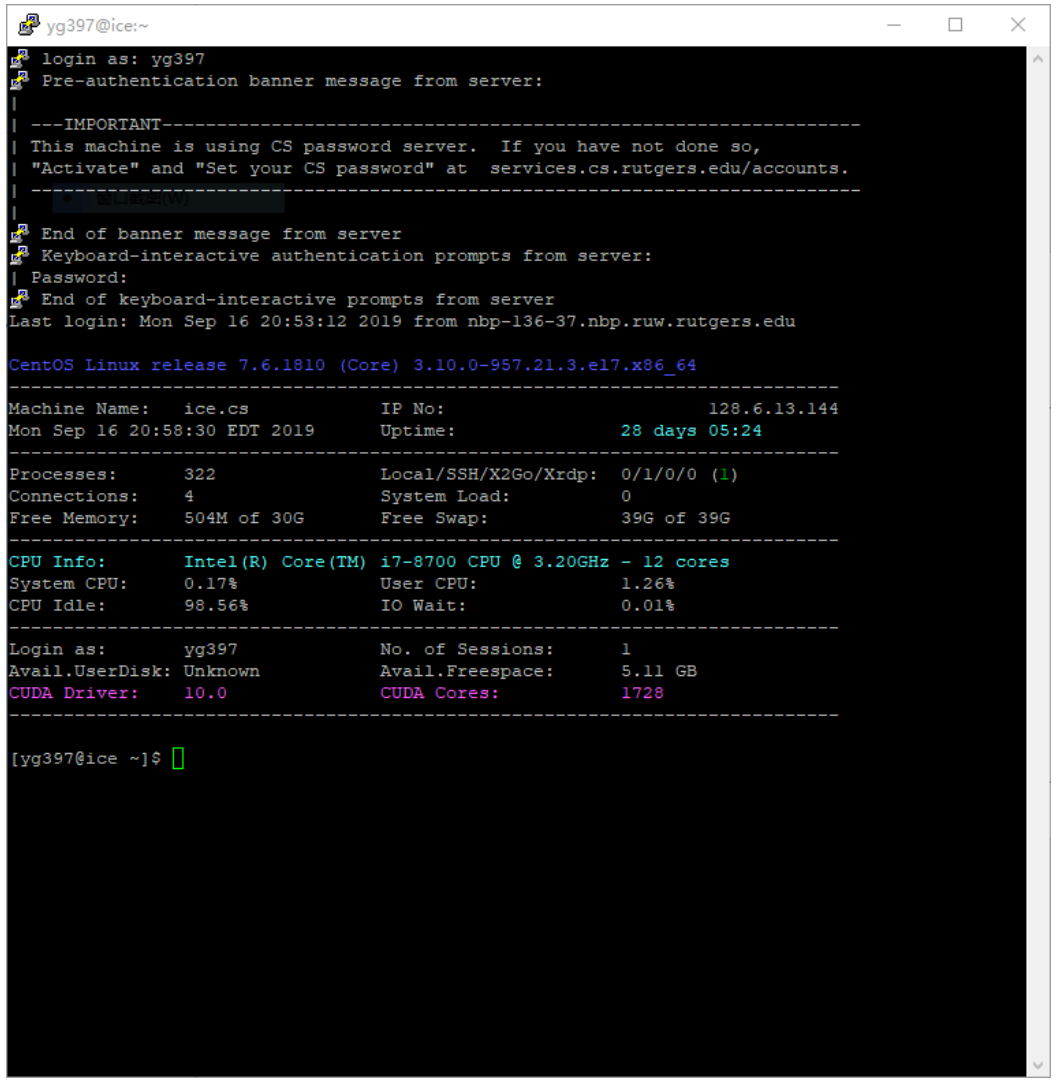
iLab – ssh from Windows

- Download a ssh client (PuTTY, WinSSH…)
  - PuTTy for example
  - Download and install: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
  - Fill in **host name** with an iLab host
  - Select SSH connection
  - Open
  - blah blah bla… Yes



iLab – ssh from Windows

- Command line open
- Type in NetID and your CS password
- Done



iLab – ssh from Linux/MacOS

- Open a terminal
- Run ssh command:
  - ssh (netid)@(hostname)
  - e.g. ssh yg397@ice.cs.rutgers.edu
- …….Are you sure …? Yes
- Type cs password
- Done

Text editors

- Command line editors
  - Vim https://vimsheet.com/
  - Emacs
  - Nano

- Text editor software
  - Sublime
  - Notepad++

Operating Systems terms

- UNIX – a multiuser and multitasking operating system
  - Linux is a free and open source version of UNIX:
  https://opensource.com/article/18/5/differences-between-linux-and-unix
- Kernel – Core of a computer's operating system
  - Handles memory management, task scheduling, and file management
- Shell – A user interface for access to an operating system's services
  - Translates user commands into a language understood by the kernel
  - Command line interface examples: C Shell(csh), Bourne-again Shell (bash), Z Shell (zsh)
  - Applications, like Terminal or Command Prompt are containers for shell

Command line - Shell

- Commands
  - pwd: print working directory
  - cd: change directory
  - ls: list files and directories
  - cp: copy files
  - mv: move files
  - rm: remove files
  - mkdir: make directory
  - ps: list processes
  - man: displays manual page for command (e.g. man ls)

- Flags
  - --help or –h

- Other notes
  - ~: home directory e.g. cd ~

C Programming – Hello World !

- hello_world.c:

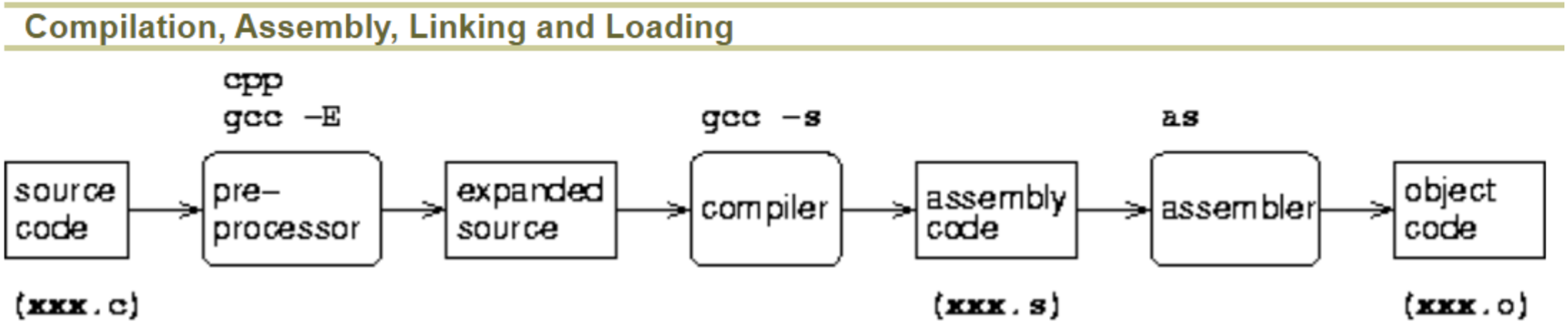
```
#include<stdio.h>

int main(){
   printf("Hello World!\n");
   return 0;
}
```

In terminal:

- gcc hello_world.c
  - compile hello_world.c into executable file: a.out
  - or use –o options to assign name to the executable file:

    gcc hello_world.c –o hello_world
- ./a.out
  - run the output file
  - or ./hello_world if you use –o option

C Programming



- http://www.cs.fsu.edu/~baker/opsys/notes/linking.html

C Programming – scp to iLab

- Once you test all functions of your code in your own PC, you need test it on iLab
- You can secure copy (scp) files to your iLab user directory
- `scp (-r) (local files or directories) (netid)@(host):(remote directory)`

Example:

`scp hello_world.c yg397@ice.cs.rutgers.edu:/ilab/users/yg397/cs214/`

  - The remote directory must already exist, -r means recursively copied files in local directory to remote directory

- Familiar with git? Push in local machine and pull in iLab

C Programming – Variables

Declare variables before use them.

Initialization of variables:

```
type identifier = initial_value;

int i;
int a = 0;
int b = 5;
int result;
result = a + b;
```

- A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z, a to z, or an underscore '_' followed by zero or more letters, underscores, and digits (0 to 9). Case sensitive.

C Programming – Data type and size

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

C Programming – Operators

- Assignment Operator

```c
int main ()
{
  int a, b;          // a:?,  b:?
  a = 10;            // a:10, b:?
  b = 4;             // a:10, b:4
  a = b;             // a:4,  b:4
  b = 7;             // a:4,  b:7
```

- Arithmetic Operator

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

- Compound Assignment

expression	equivalent to...
y += x;	y = y + x;
x -= 5;	x = x - 5;
x /= y;	x = x / y;
price *= units + 1;	price = price * (units+1);

C Programming – Operators

- Increment and Decrement

Example 1	Example 2

x = 3;
y = ++x;
// x contains 4, y contains 4
	x = 3;
	y = x++;
	// x contains 4, y contains 3

- Relational and Comparison Operator

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

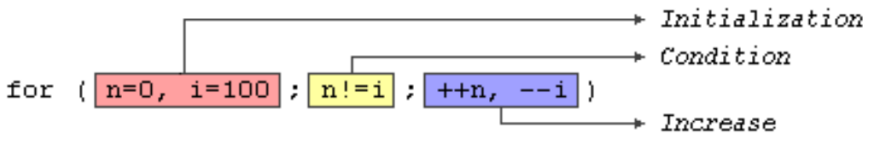
Here there are some examples:

```
(7 == 5)     // evaluates to false
(5 > 4)      // evaluates to true
(3 != 2)     // evaluates to true
(6 >= 6)     // evaluates to true
(5 < 5)      // evaluates to false
```

- Conditional Ternary Operator

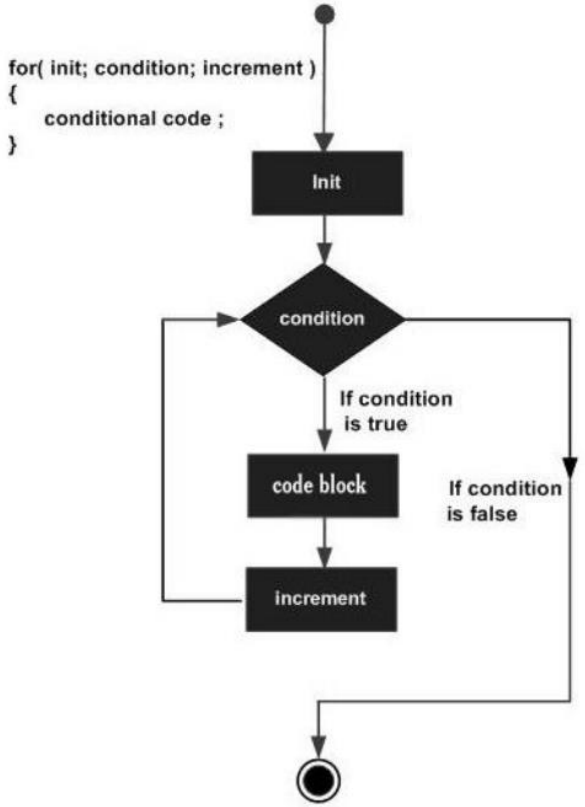
```
7==5 ? 4 : 3     // evaluates to 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3   // evaluates to 4, since 7 is equal to 5+2.
5>3 ? a : b      // evaluates to the value of a, since 5 is greater than 3.
a>b ? a : b      // evaluates to whichever is greater, a or b.
```

C Programming – for loop




```c
#include <stdio.h>

int main () {

   int a;

   /* for loop execution */
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }

   return 0;
}
```

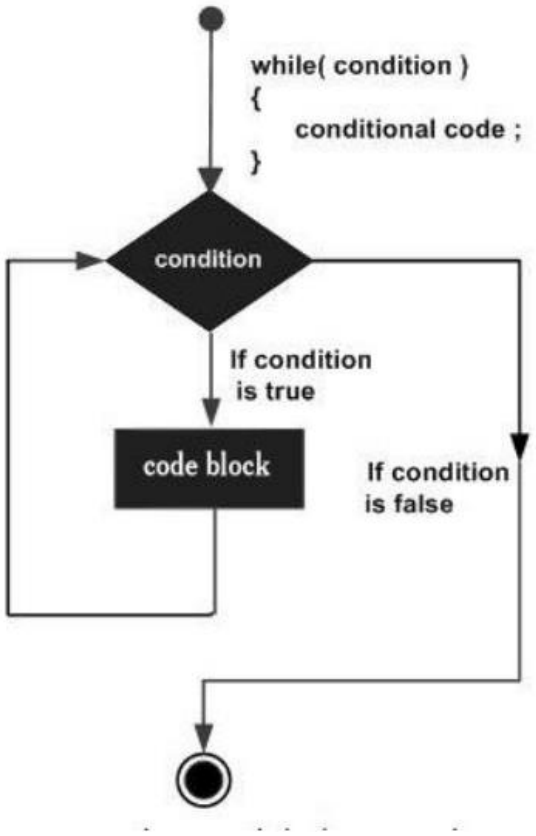
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```



C Programming – while loop

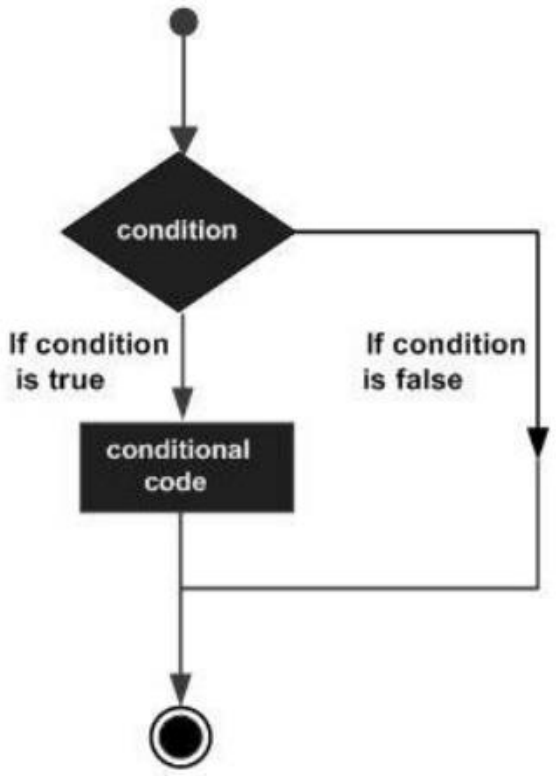
```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```



C Programming – if else condition

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a < 20 ) {
      /* if condition is true then print the following */
      printf("a is less than 20\n" );
   }
   else {
      /* if condition is false then print the following */
      printf("a is not less than 20\n" );
   }

   printf("value of a is : %d\n", a);

   return 0;
}
```




```
a is not less than 20;
value of a is : 100
```

C Programming – if else condition

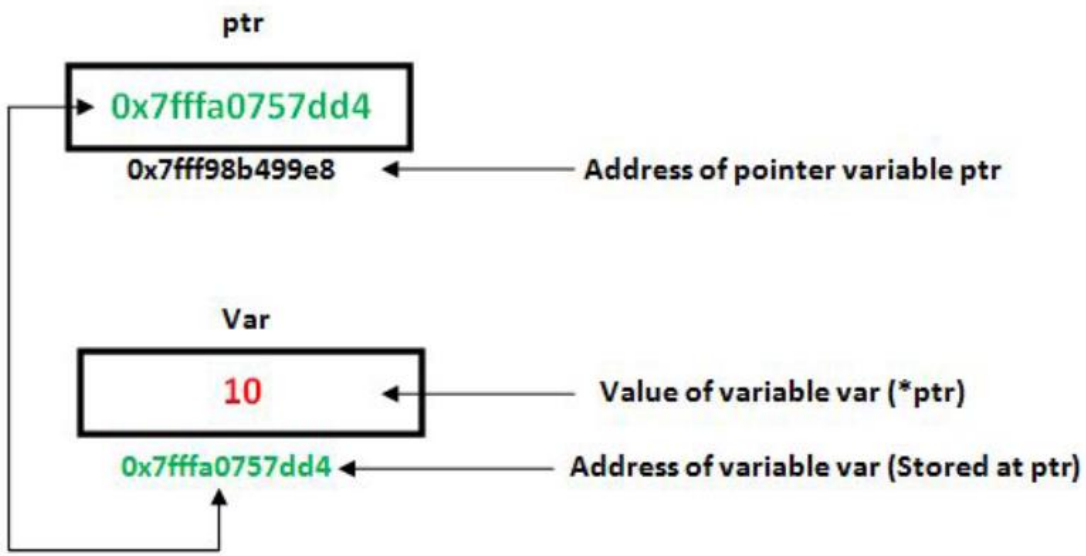
- More else

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a == 10 ) {
      /* if condition is true then print the following */
      printf("Value of a is 10\n" );
   }
   else if( a == 20 ) {
      /* if else if condition is true */
      printf("Value of a is 20\n" );
   }
   else if( a == 30 ) {
      /* if else if condition is true  */
      printf("Value of a is 30\n" );
   }
   else {
      /* if none of the conditions is true */
      printf("None of the values is matching\n" );
   }

   printf("Exact value of a is: %d\n", a );

   return 0;
}
```

```
None of the values is matching

Exact value of a is: 100
```

C Programming - Pointer

- **Pointer** "points" to locations in memory. They store memory addresses.
  - <variable type> *<variable name>
    - e.g. int *pi




```
int Var = 10;
int *ptr = &Var;        // & = address of

printf("%p", &Var);     // print memory addr
printf("%p", &ptr);     // addr of ptr
printf("%d", *ptr);     // dereference a pointer
printf("%p", ptr);      // should be same as &Var
```

C Programming - Pointer

```
int *int_ptr;
double *dbl_ptr;
printf("%ld, %ld", sizeof(int_ptr), sizeof(dbl_ptr));
```

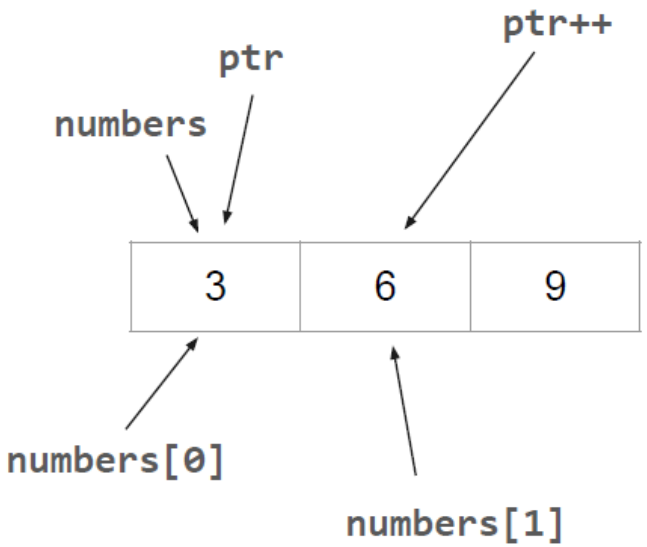
- How will the size of the pointers compare? Why?
- What is the need to include data type in pointer declarations at all?

C Programming - Array

- **Arrays** are collection of data items of the same type that can be accessed using a common variable name
  - int numbers[6] = {3, 6, 9, 12, 15, 18};

The type of numbers is *int. It points to the first item in the array.

```
numbers[0]          // = 3
*numbers            // = 3
int *ptr = numbers;
numbers[1]      // = 6
*(ptr+1)        // = 6
ptr++;
*ptr      // = 6
```



C Programming - Array

- Several ways to initialize array:
  - int numbers[3] = {1, 2, 3};
  - int numbers[] = {2, 4, 6} // Don't need to specify length
  - int numbers[3]; numbers[0] = 10; numbers[1] = 20;

  - int numbers[10] = {0, 1, 2}; // C don't care
  - int matrix[3][4] //multidimensional array

C Programming - String

String in C are just arrays of character data types with a NULL terminator(\0)

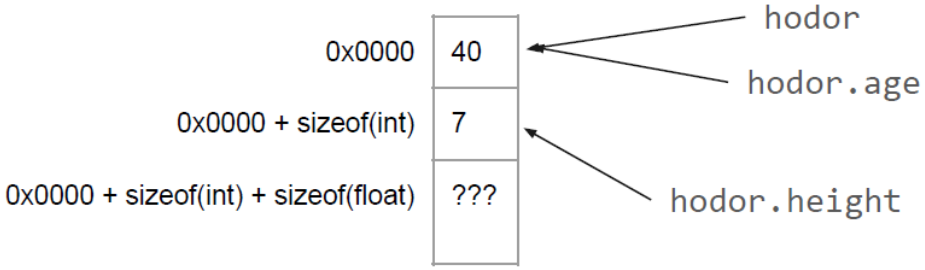
```
#include<string.h>
char str1 [] = “CAT”;
char str2[] = {‘D’, ‘O’, ‘G’, ‘\0’};

char not_str[] = {‘R’, ‘A’, ‘T’};  // This is just an array of chars. Not a String literal.
```

C	A	T	\0

C Programming – Struct

Structs (structures) provide a way of storing different values in variables of potentially different types under the same name. The data in a struct is grouped together spatially in memory.

```
struct Person{
     int age;
     float height;
};

int main()
{
     struct Person hodor;

     hodor.age = 40;
     hodor.height = 7;

}
```



Thanks

- yg397@scarletmail.rutgers.edu