CS354: Machine Organization and Programming

Lecture 3
Wednesday the September 9th 2015

Section 2
Instructor: Leo Arulraj

Class Announcements

1. If you have an exam conflict or other reason why you want to be part of the smaller makeup exam, please email me your details : Name, CS Login, Which exam, Why?
2. Class slides and code put up at Handouts page linked from course website
3. Lecture feedback: Fast ? Hard to follow ?.
4. Section 2 still has room for a few more students and has no waitlist!
5. Don't cheat in Assignments. Sophisticated software that finds out similarity even if you change variable names, add comments etc.

Class Announcements

The identical sessions will be on (bring laptops if you can):
Monday, September 14, 5:30-6:30pm, in CS 1240. and
Tuesday, September 15, 5:30-6:30pm, in CS 1221.

* Command line basics: ls, cd, mv, cp, rm, tab completion, etc…
* Man pages
* CSL-specific features:
* Backups (and recovering lost files)
* Printing
* Directory structure and quotas
* Your personal web page
* Profile (setting environmental variables)
* Text editors: vi and emacs
* Compilation: javac, gcc, g++
* ssh (or: how to work from home)

Undefined Behavior

The C FAQ defines "undefined behavior" like this:

Anything at all can happen; the Standard imposes no requirements. The program may fail to compile, or it may execute incorrectly (either crashing or silently generating incorrect results), or it may fortuitously do exactly what the programmer intended.

Undefined Behavior Example

As a quick example let's take this program:

```
#include <limits.h>
#include <stdio.h>

int main (void)
{
  printf ("%d\n", (2147483647+1) < 0);
  return 0;
}
```

Undefined Behavior Allowed Results

```
$ ./test
1

$ ./test
0

$ ./test
42
And this:

$ ./test
Formatting root partition, chomp chomp
```

Arrays in C

- C Array is a contiguous collection of variables belonging to the same data type.
- Array might be of any of the data types.
- Array size must be a constant value.
- Always, contiguous (adjacent) memory locations are used to store array elements in memory.
- It is a best practice to initialize an array to zero or null while declaring, if we don't assign any values to array.

Lecture overview

1. Arrays
2. Pointers
3. Structure & Union

Arrays

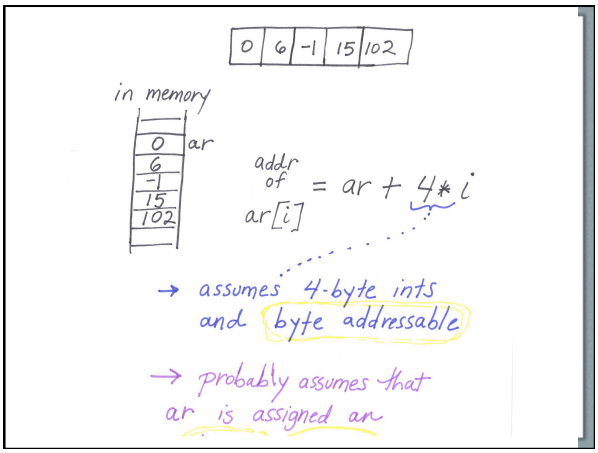
Declarations: /* an array of 100 integers */

int ar[100];

Arrays are always allocated consecutively in memory.

Access:

ar[4] = 10; // 5th element set to value 10



Arrays

- One dimensional arrays:
  Syntax: data-type arr_name[array_size];
  Examples: long myarr[10];
  int age[5]={0, 1, 2, 3, 4};
- Two dimensional arrays:
  Syntax: data_type arr_name [num_rows][num_cols];
  Examples: int arr[2][2];
  int arr[2][2] = {1,2, 3, 4};
- Multi dimensional arrays are also allowed.

Arrays

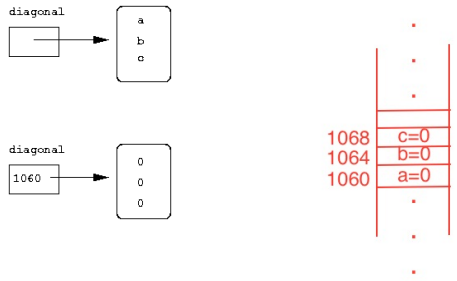
Example C Program on Arrays

Java Reference Recap

```
public class Line {
   private  int a, b, c;
  /* line is  ax + by = c  */
   public void setA(int aValue) {
      a = aValue;
   }
   public void setB(int bValue) {
      b = bValue;
   }
   public void setC(int cValue) {
      c = cValue;
   }
}
```

```
public class PlayWithLines {
   public static void main(String
args[]) {
      Line diagonal = new Line();
      diagonal.setA(1);
      diagonal.setB(1);
      diagonal.setC(0);
   }
}
```

Java Reference Recap



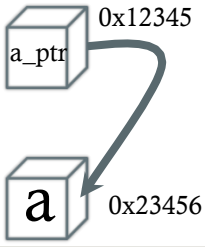
Pointers

A pointer is a memory address.

Simple example:

int a,b;

int*a_ptr = &a;



Pointers

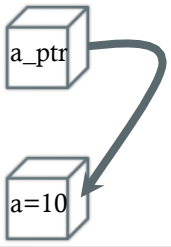
A pointer is a memory address.

Simple example:

*a_ptr = 10;

“*” operator is called

“Indirection Operator”



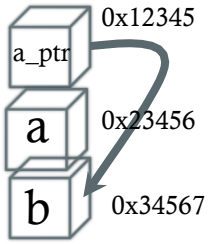
Pointers

A pointer is a memory address.

Simple example:

a_ptr = &b;

"&" operator is called "Address of" operator

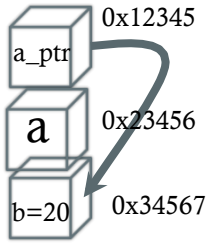


Pointers

A pointer is a memory address.

Simple example:

*a_ptr =20;

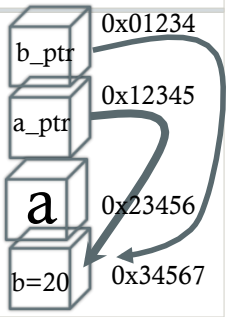


Pointers

Simple example:

int *b_ptr ;

b_ptr = a_ptr;



Pointers: Some Allowed Operations

1. Assignment to other pointers of the same type
2. Addition and subtraction of a pointer to an integer
3. Assignment of the value 0
4. Comparison to the value 0

Pointers: Some Allowed Operations

```
int a = 3; int b = 8; int c = 0;  /*declaration and initialization */
int *ap; int *bp; int *cp;  /*declaration of pointers to integers */
ap = &a; bp = &b; cp = &c;


c = *ap + *bp;
a = b + *cp;
(*bp)++;
cp++;
```

Pointers: Some Unwise Operations

1. Multiplication or division on a pointer
2. Addition or subtraction of two pointer values
3. Assignment of a value (a literal) other than 0 to a pointer

Pointers: Some Unwise Operations

```
int a = 3; int b = 8; int c = 0;
int *ap;
int *app;
int *bp;
int *cp;
ap = 34;   /* Unwise */
app = &ap;  /* Unwise */
```