Programming in C

Lecture #6
Kenneth W. Flynn
RPI CS

Outline
- Theory: Arrays
- Application: C Arrays
- Exercise: Activity 6.1
- Application: Multi Dimensional Arrays
- Application: Passing Arrays to Functions
- Exercise: Activity 6.2

Arrays
- Sometimes we need information on a list of values.
- For example, the current grade of each member of the class.

What Is An Array?
- Take a series of values and put them together
- Number them
- Example: Seats in a room

When Use An Array?
- When you have a series of values for a certain property (cars in parking spaces, students in chairs, etc.)
  - Cars in Parking Spaces
  - Students In Chairs
  - Deck Of Cards
  - Hockey Scores (By Period)
- Arrays are a fundamental construct in computer programming
Arrays In C

- All values must be of the same type
- Any type is permitted
- Declare an array much like any other value:
  - `int A[5]; /* An array of 5 integers */`
  - `char text[80]; /* Array of 80 char's */`

Zero-Indexed?

- Arrays in C are 0 indexed, i.e. the first element is A[0], not A[1]
- This means the highest element in an array is the size - 1
- Remember This!

No =

- You cannot compare to arrays using `==` to see if all the elements are the same
- You cannot set two arrays `=` to each other
- In fact, no operations will work correctly on arrays
- You must perform any operation on individual members of the array
- Can you smell lots o' for loops?

Array Bounds

- Typically, we almost never want numbers to appear in our code
- Say you have an array of 23 students in a program to keep track of your grades
- Throughout the code, I used the 23 as the size of the array
- What happens if a student adds the class?
- Better to only have the size appear once, and refer to it by a name, such as "CLASS_SIZE"

#define

- Weird syntax that allows us to create a constant
- A constant is a fixed value which cannot change during the program
- `#define CLASS_SIZE 23`
- is a compiler directive that tells the compiler to replace every instance of CLASS_SIZE in the code by the value 23
- Always use UPPERCASE for #define's
- Goes after #include's, before main
- Note: There is no ;
Array Example

```c
#define SIZE 6
int main (void)
{
    int max, c;
    int A[SIZE];
    /* Initialize A somehow */
    max = A[0];
    for (c=0; c<SIZE; c++)
        if (A[c] > max)
            max = A[c];
    return 0;
}
```

Array Practice

- 6.1a) Read 10 numbers from the user. Store them in an array. Find the average.
- 6.1b) Convert your program to read in the number of integers to read from the user, first.

Multi-Dimensional Arrays

- Arrays that instead of having one index, have more than one
- Example, a chess board has a rank and a column.
- int chess[8][8];
- Declare a multidimensional array by placing each index in brackets
- Refer to an element by enclosing the indexes in brackets
  - `x = chess[3][1];`

Passing Arrays to Functions

- In prototyping the function, instead of writing
  - `void f (int x)`
- Write
  - `void f (int x[])`
- You can enclose the size in the [], but you do not need to do so for the first dimension.
- Frequently, you may need to pass the size of the array (what elements are valid) as well
"Multi-Passing?"

- void f (int chess[][8]);
- For each element after the first, enclose the size in brackets

Pass By Value!

- Normally, you cannot modify the value of a variable passed to a function
- However, arrays are special
- If you pass an array to a function, you can alter the contents of the array
- You should probably do this frequently
- Do not return arrays

Array Passing Example

- void init_array (int A[], int size)
  {
    int c;
    for (c=0; c<size; c++)
      A[c] = 0;
    return;
  }

Array Passer

- Rewrite exercise 6.1b, using a function that returns the average of an array