

Arrays

CSE 130: Introduction to C Programming
Spring 2005

1

What We've Covered

- Algorithms
- Simple programs
- Variables and data types
- Expressions
- Conditional (selection) statements
- Loops

2

What's Coming Up

- Arrays
- Functions and recursion
- Sorting
- Structures
- Pointers

3

Arrays

- Programs often operate on large quantities of similar data
- Assigning a unique variable (and name) to each piece of data is tedious
 - Ex. var1, var2, var3, ...
- An array is a collection of many variables of the same type, all under one name

4

Declaring An Array

- To declare an array, follow the array name with a size, enclosed in square brackets:

```
double foo[5];
```
- Array sizes must be integer values
- Array sizes must be positive (> 0)

5

Array Storage

```
int a[6]; /* a holds 6 ints */
```

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	

6

Array Elements

- Individual elements of an array are accessed by using the array name, followed by an (integer) index value, enclosed in brackets
 - Ex. `myArray[1]`
- Indices are numbered starting with 0
 - Thus, `myArray[1]` refers to the second element in `myArray`

7

Array Numbering

- The name of an array (e.g., `values`) actually refers to the location in memory where the first array value is stored
- The value in brackets (the index) is an *offset* that indicates how many “values” to jump ahead from the array beginning
 - Ex. `values[3]` means three “jumps” from where `values[]` begins in memory

8

Array Access Examples

```
int numbers[10];

numbers[0] = 14; /* first element of numbers */
int temp = numbers[5];

numbers[15] = 21; /* what will this do? */
```

9

Array Boundaries

- Remember that the elements of an array are numbered from 0 to $n-1$
- C does not check to make sure that your program accesses a valid array element!
- This means that you can (accidentally) read memory that doesn't belong to your array
- This is a common programming error

10

Initializing Arrays

- Arrays can be initialized when they are declared:

```
int bar[5] = {5, 4, 3, 2, 1};
```
- If the array size is greater than the number of elements, the remaining array elements are set to 0:

```
int foo[20] = {2, 4, 6, 8};
```

11

Initializing Arrays

- Specific array elements can be initialized in any order
 - Use the element number to do this:

```
float sample[5] = {[2] = 500.5, [1] = 300.0, [0] = 100.0};
```

12

Arrays and Characters

- We can create arrays of characters just as did with ints and floats

```
char letters[5] = {'a','b','c','d','e'};
```

```
char myString[] = {"Hello!"};
```

- We can also omit the braces when initializing a character string:

```
char word[] = "Hello!";
```

13

The Null Character

- Arrays are not required to be full
- This means that your program needs to keep track of how many elements are in use
- The null character ('\0') solves this problem for arrays of characters
- This is added automatically when we use a string to initialize a char array

14

The Null Character

```
char word[] = {"Hello!"};
```

actually creates an array with seven characters: 'H', 'e', 'l', 'l', 'o', '!', and '\0'

The same is true for:

```
char word = "Hello!";
```

This is *not* true if you fix the size of the array!

15

The Null Character

- The null character is used to terminate a string of characters
- Most C functions automatically look for the null character when using a string
- For example, in calls to printf(), the control string is automatically null-terminated
- To print a string with printf(), use %s:
printf("%s\n", word);

16

Arrays and Loops

- Loops (especially for loops) are the perfect way to manipulate arrays

```
int a[5];
```

```
int i;
```

```
for (i = 0; i < 5; i++)
```

```
    a[i] = i * 2;
```

17

Another Example

```
char message[] = "Hello, world!";  
int j;
```

```
for (j = 0; j < 13; j += 2)
```

```
    printf("%c", message[j]);
```

will print: Hlo ol!

18

Multidimensional Arrays

- Arrays can have more than one dimension
- We can have arrays of arrays
- Ex. matrix of $n \times n$ elements
- Just add additional pairs of brackets to indicate additional dimensions
- Ex. `int b[2][7]; /* 2 rows, 7 columns */`

19

More on Multidimensional Arrays

- When used as a function parameter, all sizes except the first must be specified
- Ex. `int sum(int a[][5])`
- Initialization:
 - `int a[2][3] = {1, 2, 3, 4, 5, 6};`
 - `int a[2][3] = {{1, 2, 3}, {4, 5, 6}};`
 - `int a[][3] = {{1, 2, 3}, {4, 5, 6}};`

20

Example

```
int i, j, sum = 0;
int a[3][5] = { ... }; /* assume a is initialized */
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 5; j++)
    {
        sum += a[i][j];
    }
}
```

21

Variable-Length Arrays

- Some (modern) C compilers will allow you to specify a non-constant size
- You can use a variable to indicate the array's size
- Ex. `int size = 40;`
`float values[size];`
- We can also do this (later in the semester) with dynamic memory allocation

22

Next Time

- Functions
- Recursion

23