ISA 563: Fundamentals of Systems Programming

Arrays and Character Strings

Jan. 29, 2013

Outline

- Arrays
  - Data collection for multiple objects of the same type
- Strings
  - Array of characters: common and important enough to discuss separately
  - String operations
  - String literals and string constants
- Command line argument processing

Simple Data Collections

- Store a students grade:
  - `int jacks_grade = 90;`
- What if we have 100 students?
  - `int student0_grade = 0;`
  - `int student1_grade = 0;`
  - `int student2_grade = 0;`
  - …
  - `int student99_grade = 0;`

  Whew!

Arrays

- Arrays are typed collections of the same data type
- Arrays allow for grouping of data under one common name
- Array elements are accessed by giving an offset or index into the array
- Offsets (indexes) are always integer values
    - it does not make sense to say “give me the value at element number 3.58”

Array Organization

- Arrays have a name
  - e.g., student_grades
- Arrays have a size
  - not the number of elements in the array, but rather how much memory the array takes up
- Array have a length
  - in C, this length is not stored with the array. You, the programmer, must keep track of it
  - no bounds checking while accessing the array
- Array elements must be consistently typed

Accessing Arrays Elements

- Each element is at a unique position in the array
  - position is indicated by the subscript or index value
  - the value of the subscript or index is NOT the value the element at that index or position

```
int student_grades[100];
student_grades[0] = 98;
student_grades[45] = 85;
student_grades[99] = 79;
```

Declaring an Array

- Very similar to declaring a single variable of that type:

```
// declare an integer variable
int my_integer;
```

- Just add brackets and size:

```
// declare an integer array
int my_integers[400];
```

Initializing an Array

- There are several ways to initialize the data in an array.
  - at definition
    - `int temperatures[] = {89, 54, 100, 23, -12};`
      - compiler will figure out the size
  - an explicit loop

```
for ( i = 0; i < array_size; i++) {
   my_array[i] = 0;
}
```

  - series of statements

```
int i = 0;
my_array[i++] = 0;
my_array[i++] = 0;
...
```

Array Notes

- Index:
  - Arrays start indexing from 0, not 1
  - thus, the array has the maximum index of (length-1)
- C does not check array bounds
  - Compiler and the execution environment do not check out-of-bound reads and writes. Such operations are not what you wanted to do, and are errors most of the time.

Advanced Array Topics

- There are other ways to access array elements
  - We'll see one when we cover pointers
- Arrays can be nested:
  - Arrays of arrays
  - Just add more []s per dimension
  - A two-dimensional array is an array of arrays, or a table

Multi-dimensional Arrays

```
// declare a two dimensional array of integers
int no_students = 100;
int no_subjects = 7;

int class_grades[no_students][no_subjects];
// or
int class_grades[100][7];

// access an element by providing subscripts
class_grades[45][6]=86;

// print the 6th student's grade on 5th subject:
printf(“%d\n”, class_grades[5][4]);
```

Strings

- Strings are arbitrarily long sequences of characters
- C keeps many things as simple as possible
  - strings are not first class data objects
  - strings are simply character arrays
  - have to keep some rules in mind when operating on strings
- Just remember that a string is always an array of characters (and treat it as such) and you'll be fine

Character Basics

- Characters in C are 8-bit (1-byte) values that sometimes be treated like small integers
- How many unique integer values can you specify with 8 bits?
- In a program, you may represent a character like:

```
char somechar = 'B';
```

but numbers work equally well:

```
char anotherchar = 66;
```

Example of Strings

- You've seen some strings before:
  - String literals: a sequence of characters in quotation marks inside the text or body of a program:

```
printf(“result is %d\n”, result);
```

    the “result is %d\n” is a string literal
  - A character array is the other common way to refer to a string

```
char student_name[30];
```

String Notes

- In order to truly treat a character array as a string, you must make sure that it is null-terminated
  - the last character in the array must be a null character
  - the null character is written as '\0' (backslash zero)
  - recall the '\n' for newlines
  - the C compiler automatically null-terminates string literals

Char arrays as … char arrays

- Every string is a character array
- Not every character array is string
  - character arrays are just collection of chars
  - can hold any legal char value (8 bits of information)
  - interpretation depends on context
  - the data stored in a character array does not need to be treated like a string
  - nevertheless, you can still treat it like a string. C allows you to shoot yourself in the foot if you really want to

String Operations

- Many basic string operations are tedious to write
- So these operations are provided as functions in the standard C library
    - to use them you program should

      ```
      #include <string.h>
      ```
- Operations include:
    - strlen (return the length of the string)
    - strncmp (compare two string lexicographically)
    - strncpy (copy on string to another)

String Properties

- The length of the string
  - the number of characters in the string, NOT counting the '\0' (null terminator)
- Strings are compared by comparing their basic elements: the characters that they contain
  - compared in lexicographic order
- Semantics are consistent when you deal with multi-dimensional char arrays:

```
char class_names[100][30]; is an array of
character arrays (array of strings)
```

String Comparison

calling
  strncmp(“hello”, “hello”, 5);
return 0, because the strings are equal

calling
  strncmp(“yes”, “nah”, 3)
returns a positive number, because the strings are different, and “yes” is lexicographically greater than “nah”

calling
  strncmp(“nah”, “yes”, 3)
return a negative number, because the strings are different and “nah” is lexicographically less than “yes”

What are return values of:
strncmp(“hello”, “hello!”, 5)
and
strcmp(“hello”, “hello!”);

Command Line Input

- One way to supply input to your program
- Data is provided by execution environment
  - How do you refer to it in your code?
- C provides a place for this input:
  - argc: an integer specifying the number of args
  - argv: an array of strings holding actual values

Demo

argtest.c