Introduction to Arrays in C

CSE 130: Introduction to Programming in C

Stony Brook University

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

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)

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`

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 number in brackets (the index) is an offset that indicates how many elements to jump ahead from the array beginning
- Ex. `values[3]` means three "jumps" from where `values[ ]` begins in memory

Array Access Examples

```
int numbers[10];

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

int temp = numbers[5];

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

Array Boundaries

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

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};
```

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;
```

Array Examples

Program 1

- This program:
  - reads in a list of 10 integers
  - multiplies them together
  - prints their product
  - prints the list in reverse order

Program 1, part 1

```
#include <stdio.h>

/* constant declarations */
const int SIZE = 10; /* max elements in array */

int main (void)
{
  /* Variable declarations */
  int values[SIZE]; /* array to hold user input */
  int product = 1;  /* product of user input */
  int i, temp;      /* temporary variables */
```

Program 1, part 2

```
/* Read in (SIZE) values from the user */

for (i = 0; i < SIZE; i++)
{
   printf("Enter a value: ");
   scanf(" %d", &temp);
   values[i] = temp;
}
```

Program 1, part 3

```
/* Compute the product of the values */

for (i = 0; i < SIZE; i++)
  product = product * values[i];


/* Print the product */
printf("\n\nThe product is %d\n\n", product);
```

Program 1, part 4

```
/* Print the list in reverse order */
  for (i = SIZE - 1; i >= 0; i--)
    printf("%d\n", values[i]);
  return 0;
}
```

Program 2

- This program:
  - Generates a list of 200 random integers between 0 and 100
  - Counts the number of times each value occurs
  - Prints the number of times each value appears

Program 2, part 1

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>



/* Constant declarations */
const int SIZE = 200; /* # of values */
const int RANGE = 101; /* # of possible values */



int main (void)

{

  /* Variable declarations */

  int values[SIZE], counts[RANGE];
```

Program 2, part 2

```
/* Seed the random number generator */

srand(time(0));
```

```
/* Generate SIZE random integers */

for (i = 0; i < SIZE; i++)
{
  values[i] = rand() % RANGE;
}
```

Program 2, part 3

```
/* Initialize counts[ ] */
for (i = 0; i < RANGE; i++)
   counts[i] = 0;
/* Count # of occurrences */
for (i = 0;i < SIZE; i++)
{
   temp = values[i];
   counts[temp] = counts[temp] + 1;
}
```

Program 2, part 4

```
/* Print # of occurrences */

printf("Value\tOccurrences\n\n");

for (i = 0; i < RANGE; i++)
  {
    printf("%d\t%d\n", i, counts[i]);
  }
  return 0;
}
```