CS1010 Programming Methodology

Week 8: One-Dimensional Arrays

> Every act of conscious learning requires the willingness to suffer an injury to one's self-esteem. That is why young children, before they are aware of their own self-importance, learn so easily. ~*Thomas Szasz*

To students:

Welcome back! We hope that you have made good use of the one-week recess to revise on past topics covered before the recess. Also, we hope that you have written a lot of programs on your own for your self-learning.

By now, you should know what is expected of you before and during the discussion session.

Your DL may skip some of the questions here and give you some other questions instead. They may send the questions to you via email.

Please be reminded that your **mid-semester test** is on **this Saturday, 12 October 2013**. Please refer to http://www.comp.nus.edu.sg/~cs1010/3_ca/termtests.html for more information.

I. For your own attempts

The questions in this section are meant for you to attempt on your own before your discussion session. You **must** do these questions on your own. **They will not likely be discussed in class**, as these are the basics which we expect you to know. You can verify the answers yourself, but if you do have doubts, please post them on the IVLE forum.

The programs are on the CS1010 website, under "Discussion" page, or you could copy them in your UNIX account, for example, to copy q1a.c:

```
cp ~cs1010/discussion/prog/week8/q1a.c .
```

1. **Spot the errors.**

   (a) Spot the errors in **q1a.c**. Are there any compilation errors?

```
#include <stdio.h>

int main(void) {
   float[5] values;
   int i;

   for (i=1; i<=5; i++)
      values[i] = 2.5 * i;

   for (i=1; i<=5; i++)
      printf("%f ", values[i]);
   printf("\n");

   return 0;
}
```

Can you print all the elements in an array by replacing the second **for** loop in the program above with just the following statement?

```
printf("%f\n", values); // to print ALL elements
```

(b) Spot the errors in **q1b.c**. Are there any compilation errors?

```
#include <stdio.h>

float sumArray(float, int);

int main(void) {
   float prices[6], total;
   prices = { 10.2, 5.3, 4.4, 6.8, 7.7, 9.5 };

   sumArray(prices[6], 6);
   printf("Total = %f\n", total);

   return 0;
}

float sumArray(float arr, int size) {
   int i;
   float sum = 0.0;

   for (i=0; i<size; i++)
      sum += prices[i];

   return sum;
}
```

Assuming that the program is corrected. In the main() function, what if we (i) pass 3 to the **size** parameter of sumArray() function; (ii) pass 10 to the **size** parameter of sumArray() function? Will there be compilation error?

2. **Manual tracing.**
   Trace the programs manually and write out their outputs. Then run the programs to verify whether your outputs are correct.

   The function **printArray()** used in **q2a.c** and **q2b.c** is shown below:

```
void printArray(int arr[], int size) {
   int i;

   for (i=0; i<size; i++)
      printf("%d ", arr[i]);
   printf("\n");
}
```

   (a) Program **q2a.c**

```
#include <stdio.h>
#define LENGTH 10

void printArray(int [], int);
void arrange(int [], int);

int main(void) {
   int numbers[] = {1,2,3,4,5,6,7,8,9,10};

   arrange(numbers, LENGTH);
   printArray(numbers, LENGTH);

   return 0;
}

void arrange(int arr[], int size) {
   int i;

   for (i=2; i<size; i++) {
      arr[i-2] += arr[i];
      arr[i-1] += arr[i];
   }
}

// printArray() omitted here for brevity
```

   You want to call **printArray()** somewhere in the program for checking/debugging purpose. Where is a good place to put it and how to call it?

(b) Program **q2b.c**

```
#include <stdio.h>
#define LENGTH 10

void printArray(int [], int);
void process(int [], int);

int main(void) {
   int numbers[] = {5,1,3,9,7,8,2,0,6,4};

   process(numbers, LENGTH);
   printArray(numbers, LENGTH);

   return 0;
}

void process(int arr[], int size) {
   int i;

   for (i=0; i<size; i++) {
      arr[i] = arr[arr[i]];
   }
}

// printArray() omitted here for brevity
```

You want to call **printArray()** somewhere in the program for checking/debugging purpose. Where is a good place to put it and how to call it?

II. Exploration: Array and Pointer

(As this topic is slightly beyond the basics, your DL may choose to skip this part if he/she prefers to focus on the basics and problem-solving. At this point, this offers an alternative to the syntax that you have learned, so no great loss if you skip this.)

3. In Week 7 lecture, we mentioned the relationship between arrays and pointers briefly. Slide 32 reveals that an array name is the address of its first element. This is illustrated in slides 40 and 46 during lecture. Slide 37 shows the alternative syntax for array parameter in a function.

   The following program, **arraySum.c**, is adapted from **Week7_SumArray.c** (slide 35).

```
#include <stdio.h>

int sumArray(int [], int);

int main(void) {
   int foo[] = {44, 9, 17, 1, -4, 22};

   printf("sum is %d\n", sumArray(foo, 4));

   return 0;
}

// To sum up values in array 'arr' with 'size' elements
int sumArray(int arr[], int size) {
   int i, total=0;

   for (i=0; i<size; i++)
      total += arr[i];

   return total;
}
```

   The following program, **arraySumUsingPointer.c**, uses pointer syntax for the array parameter, as well as pointer operation in the **sumArray()** function. The **main()** function is omitted below as it is the same as the one in **arraySum.c**.

```
int sumArray(int *, int);

int main(void) { ... } // omitted for brevity

// To sum up values in array 'arr' with 'size' elements
int sumArray(int *arr, int size) {
   int *p, total=0;

   for (p = arr; p < arr + size; p++)
      total += *p;

   return total;
}
```

You DL may explain this code if he/she chooses to discuss it.

III. Arrays

Important notes:

As we are basing on ANSI C (or C90), we do NOT allow variable-length arrays. Hence the following code is not permitted, because at compilation time the system wouldn't know what value **n** contains and hence the size of the array is unknown:

```
int n, a[n];
printf("Enter n: ");
scanf("%d", &n);
. . .
```

However, as gcc is C99-compliant (C99 is a newer standard for C, which permits variable-length arrays), the code above can be compiled without warning, even if –Wall option is used.

You may compile with the –pedantic option instead, in which case a warning message will be issued.

For all problems on arrays, we will specify the maximum size of an array, so that you can declare the array with the right size. We will not accept the use of variable-length arrays.

4. This is a true story. On 25 August 2010 we received an email from a lecturer about the following program **q4.c**, which when run, gives an infinite loop! Give a possible explanation. What is the moral of the story?

```
#include <stdio.h>
int main(void) {
   double arr[] = { 1.1, 2.2, 3.3, 4.4 };
   int i;

   for (i=0; i<=4; i++) {
      printf("%d\n", i);
      arr[i] = 0;
   }
   return 0;
}
```

5. **Logical thinking.**
   An array is a collect of data. It is very common to ask the following two questions about a collection: (a) do all the data in the collection share a certain property? (b) does there exist one datum that has a certain property? The former is a *universal* question, and the later an *existential* question.

   For example, "are all the values in the array non-negative?" is a universal question; "is there one value in the array that is negative?" is an existential question. In this case, the two questions are actually the same, hence we can transform one into another.

   Write a function **nonNegative(int arr[], int size)** that returns 1 (true) if all the elements in arr[0]… arr[size-1] are non-negative; or returns 0 (false) otherwise. You may assume that the array has at least one element.

6. **Logical thinking.**
   Given an array of integers, write a function **isSorted(int arr[], size)** that returns 1 (true) if the array **arr** is sorted in non-decreasing order, or returns 0 (false) otherwise. You may assume that the array has at least one element.

   For example, 3, 12, 15, 18 and -5, 8, 8, 10 are in non-decreasing order, but 4, 6, 9, 7, 12 is not.

   Do you see any similarity between this question and Q5? For this question, what is the property you have "abstracted" out to check?

IV. Unpublished Questions

Your DL may email you some questions for you to prepare before coming to this discussion session.