Intro. Computing with the C Programming Language

Arrays

Shin Hong

4 October 2023

Motivation

Write a program that receives an arbitrary number of integers, and then prints them in ascending order

Write a program that receives 20 integers and then prints out all unique integers with the number of repetitions

Array

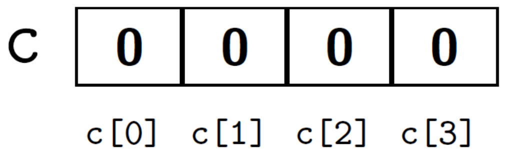
- A array is a set of values where each value is identified by a number
  - each element of a array is referenced by its index
  - the indices of an array starts from 0 to length – 1.
  - declaration

    <type> array-name `[` <length> `]` `;`

    ```
    int c[4] ;
    ```

  - initialization

    ```
    int c[4] = {0, 1, 2, 3} ;
    ```



Examples

- array.c
- sort.c

Increment and Decrement Operators

- An increment/decrement operator returns the current value of a variable and immediately increases/decreases it by 1
  - Ex.

```
int a = 3, b ;

a++ ;      // a = a + 1 ;
a-- ;      // a = a - 1 ;
b = a++ ; // b = a ; a = a + 1 ;
```

Compound Assignment

- A compound assignment updates the left-hand side as the result of an arithmetic operation with the right-hand side
- Examples

```
a += b ;      // a = a + b ;
a -= b ;      // a = a - b ;
a *= b ;      // a = a * b ;
```

Accessing Array Elements

- The [] operator allows us to read and write an element of an array at a specific index
  - ex.

```
c[0] = 7;
c[1] = c[0] * 2;
c[2]++;
c[3] -= 60;
```

- The result of accessing an array beyond its length is undefined, and often such an access results a crash
  - ex. array-out-of-bound.c

Passing Array to Function

- A function may have an array type argument
  - ex. arr_inc.c

```
void arr_inc (int a[], int length) {
  int i ;
  for (i = 0 ; i < length ; i++) {
    a[i]++ ;
  }
}
```

- An array type argument does not receive copied values from the caller, but receives the address of the array variables
  - updating the array at a callee function does change the original array at the caller function
  - note that the mention about ``call-by-reference`` in the textbook may be misled. I will give better explanation on this concept after we study pointer.

Array Length

- The elements of an array is consecutively placed in memory

- `sizeof(a)` gives the number of bytes allocated for a variable `a` or a type `a`
  - `sizeof` is a C operator, not a function

- `sizeof(arr)` retries the bytes allocated for `arr`, thus `sizeof(arr)/sizeof(arr[0])` gives the number of elements of `arr`

Random Numbers

- Function `rand()` returns an integer between 0 and `RAND_MAX`
  - to use `rand()`, `stdlib.h` must be included
  - `rand()` uses pseudo-random number generation algorithm
- The sequence of numbers generated from `rand()` is determined by a seed
  - the random seed can be defined by `srand()`
  - in most cases, a seed is obtained from the current time, for example by calling `time()`
- Examples
  - `random.c`

Two-dimensional Array

- A two-dimensional array is an array of single-dimensional arrays
  - e.g.

    ```
    int a[5][4] ;
    ```

    an array of 5 single-dimensional array with 4 integers

- The initial values of a two-dimensional array are formed as a list of value lists guarded by {}
  - e.g.,

    ```
    int a[3][2] = {{1,2}, {3,4}, {5,6}} ;
    ```

- Example
  - `histogram.c`