CPNM Lecture 6 - Arrays

Mridul Sankar Barik

Jadavpur University

2022

Introduction

- Scalar variables hold a single data item
- Aggregate variables can store collections of values
  - Two types of aggregates in C: arrays and structures

One Dimensional Array

- An array is an ordered collection of data values, all of which have the same type
- These values, known as elements, can be individually selected by their position within the array
- One dimensional array: The elements are conceptually arranged one after another in a single row
- To declare an array, we must specify the type of the array's elements and the number of elements

```
int a[10];
```

- The length of the array can be specified by any (integer) constant expression

```
#define N 10
...
int a[N];
float b[N*2];
```

Array Subscripts

- To access a particular element of an array, we write the array name followed by an integer value in square brackets ⇒ known as subscripting or indexing
- Array elements are always numbered starting from 0, so the elements of an array of length `n` are indexed from 0 to `n-1`
- Expressions of the form `a[i]` are lvalues, so they can be used in the same way as ordinary variables:

```
a[0] = 1;
printf("%d\n", a[5]);
++a[i];
```

- An array subscript may be any integer expression: `a[i+j*10] = 0;`

Arrays and Loops

- Example: clear all elements of an array a

```
for (i = 0; i < N ; i++)
    a[i] = 0;
```

- Example: read data into array a

```
for (i = 0; i < N; i++)
    scanf("%d", &a[i]);
```

- Example: sum the elements of an array a

```
sum = 0;
for (i = 0; i < N; i++)
    sum += a[i];
```

Subscript Bounds

- C doesn't check subscript bounds; if a subscript goes out of range, the program's behavior is undefined
- Example:

```
int a[10], i;
for (i = 1; i <= 10; i++)
    a[i] = 0;
```

- With some compilers, this statement causes an infinite loop
  - When `i` reaches 10, the program stores 0 into `a[10]`
  - But `a[10]` doesn't exist; So, 0 goes into memory immediately after `a[9]`
  - If the variable `i` happens to follow `a[9]` in memory, then `i` will be reset to 0, causing the loop to start over

Array Initialization I

- An array can be given an initial value at the time it's declaration
- Array initializer: a list of constant expressions enclosed in braces and separated by commas

  ```
  int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  ```
- If the initializer is shorter than the array, the remaining elements of the array are given the value 0
- Initialize an array to all zeros: `int a[10] = {0};`
- It's illegal for an initializer to be completely empty
- If an initializer is present, the length of the array may be omitted

  ```
  int a[] = {1, 2, 3, 4, 5, 6, 7, 6, 9, 10};
  ```
  - Compiler uses the length of the initializer to determine the length the array

Array Initialization II

- **Designated Initializers**: supported by C99 standard
  `int a[15] = {0, 0, 29, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 48};`
  This can be rewritten as
  `int a[15] = {[2] = 29, [9] = 7, [14] = 48};`
  - The order in which the elements are listed, does not matter
  - If the array being initialized has length n, each designator must be between 0 and `n - 1`
  - If the length of the array is omitted, a designator can be any non-negative integer. In the latter case, the compiler will deduce the length or the array from the largest designator

Example I

- Example: To check whether a number contains repeated digits

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

int main(void){
    int digit_seen[N] = {0};
    int digit;
    long n;

    printf("Enter a number: ");
    scanf("%ld", &n);

    while(n>0){
        digit=n%10;
        if(digit_seen[digit])
            break;
        digit_seen[digit]=1;
        n=n/10;
    }
    if(n>0)
        printf("Repeated digit \n");
    else
        printf("No repeated digit \n");
}
```

Example

```
/*Read 10 integers and find the maximum*/
#include<stdio.h>
#define N 10

int main(void){
    int a[N], i, max;

    for(i=0; i<N; i++)
        scanf("%d", &a[i]);
    max = a[0];
    for(i=1; i<N; i++)
        if(a[i]>max)
            max=a[i];
    printf("The maximum number is %d\n", max);
    return(0);
}
```

Flow Chart

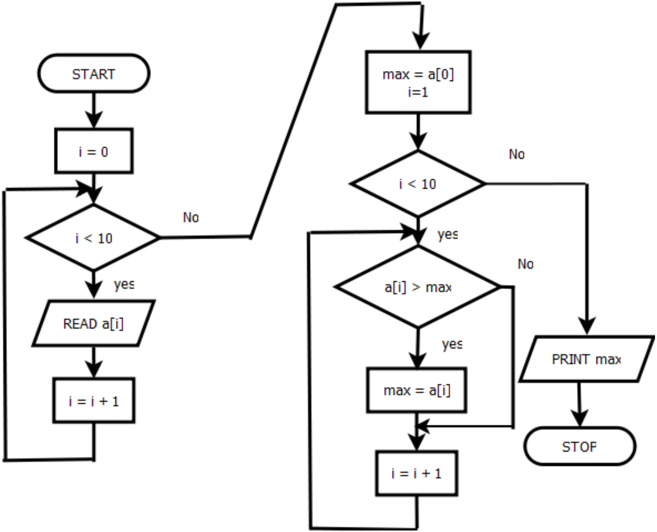

Figure 1: Flow Chart for Finding Maximum of 10 Numbers

Using the sizeof Operator with Arrays

- `sizeof(a)` gives size of array in number of bytes
- Applying `sizeof` on array int a[10] gives 40
- Applying `sizeof` on any single element gives 4
- The number of element can be obtained by `sizeof(a)/sizeof(a[0]) = 10`
- Example: loop need not be modified if array length is changed

```
int a[10];
int i;
...
...

for(i=0; i<(sizeof(a)/sizeof(a[0])); i++)
    a[i]=0;
```

Multidimensional Arrays I

- An array may have any number of dimensions
- To create a two dimensional array
  `int m[5][9];`
- To access the element of m in row i, column j, we write
  `m[i][j]`
- C stores arrays in **row-major** order, with row 0 first, then row 1, ...
- Nested for loops are ideal for processing multidimensional arrays
- An array in C can have maximum 32 dimensions
- Example: Initialize an array for use as an identity matrix

Multidimensional Arrays II

```
#define N 10

double ident[N][N];
int row, col;

for(row = 0; row < N; row++)
    for (col = 0; col < N; col++)
        if (row == col)
            ident[row] [col] = 1.0;
        else
            ident[row] [col] = 0.0;
```

Initializing a Multidimensional Array

- We can create an initializer for a two dimensional array by nesting one-dimensional initializers:

  ```
  int a[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
  int a[3][3] = {{1,2,3}}; /*last two rows contain 0's*/
  ```

- If an inner list isn't long enough to fill a row, the remaining elements in the row are initialized to 0:

  ```
  int a[3][3] = {{1,2}, {5}, {9,10}};
  ```

- We can even omit the inner braces

  ```
  int a[3][3] = {1,2,5,9,10};
  ```

- C99's designated initializers work with multidimensional arrays

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

Address of an Element in an Array

- Single dimensional array
  - Address of `a[i] = Base Address + i * sizeof(element)`
- Double dimensional array
  - Array dimension is `R`×`C`
  - Address of `a[i][j] = Base Address + (i * C + j) * sizeof(element)` for row major storage
  - Address of `a[i][j] = Base Address + (j * R + i) * sizeof(element)` for column major storage

Constant Arrays

- Any array can be made ”constant” by starting its declaration with the word canst

  ```
  const char hex_chars[] = {’0’, ’1’, ’2’, ’3’, ’4’, ’5’,
  ’6’, ’7’, ’8’, ’9’, ’A’, ’8’, ’e’, ’D’, ’E’, ’F’};
  ```

- A constant array should not be modified by the program

Example - Matrix Operations I

- Generate Matrix

```
int i, j, k;
srand((unsigned int) time (NULL)); //Seed generator

for(i=0;i<MAX;i++)
for(j=0;j<MAX;j++)
     a[i][j]=(double)(rand()%k);
     //Generate values
```

- Print Matrix

```
for(i=0;i<MAX;i++){
    for(j=0;j<MAX;j++)
        printf("%g\t", a[i][j]);
        //Print a row in one line
    printf("\n");
}
```

- Multiply Matrices

Example - Matrix Operations II

```
int i, j, k;

for(i=0; i<MAX; i++)
    for(j=0; j<MAX; j++){
        c[i][j] = 0.0;
        //Initialize elements of product matrix
        for(k=0; k<MAX; k++)
            c[i][j] += a[i][k] * b[k][j];
            //Inner product
    }
```

Example I

- A program to accept roll numbers of ten students and also marks obtained by them in three subjects. Program should print total marks of each students.

```
#include<stdio.h>

#define N 10

int main(void){

  int roll[N], marks[N][3], i, j, sum;

  for(i=0; i<N; i++){
    printf("Enter roll number of %dth student: ", i+1);
    scanf("%d", &roll[i]);
    for(j=0; j<3; j++){
      printf("Enter Marks %d of Student %d: ", j+1, i+1);
      scanf("%d", &marks[i][j]);
    }
  }

  for(i=0; i<N; i++){
    printf("Marks obtained by Student with roll %d: ", roll[i]);
    for(j=0; j<3; j++){
      printf("%d ", marks[i][j]);
    }
    printf("\n");
  }
  printf("\n");
```

Example II

```
  for(i=0; i<N; i++){
    printf("Total Marks obtained by Student with roll %d: ", roll[i]);
    sum = 0;
    for(j=0; j<3; j++){
      sum = sum + marks[i][j];
    }
    printf("%d\n", sum);
  }
}
```

Another Example I

- A program to accept names of ten students and marks obtained by them in five subjects and to print the names of the students who have obtained highest marks subject wise

```
#include<stdio.h>

#define STU_SIZE 10
#define SUB_SIZE 5

int main(void){
    char name[STU_SIZE][40];
    int marks[STU_SIZE][SUB_SIZE], i, j, high_index, high_marks;

    for(i=0; i<STU_SIZE; i++){
        printf("Enter name of %dth student: ", i+1);
        scanf("%[^\n]s", name[i]);
        for(j=0; j<SUB_SIZE; j++){
            printf("Enter marks obtained by %s in Subject %d: ",
                name[i], j+1);
            scanf("%d", &marks[i][j]);
        }
        getchar();
    }
```

Another Example II

```
    for(j=0; j<SUB_SIZE; j++){
        high_index=0;
        high_marks=marks[0][j];
        for(i=1; i<STU_SIZE; i++){
            if(marks[i][j]>high_marks)
                high_index=i;
        }
        printf("Subject %d: Highest marks %d obtained by %s\n",
            j+1, marks[high_index][j], name[high_index]);
    }
}
```