

Programming Fundamentals for Engineers - 0702113

## 6. Arrays

**MUNTASER ABULAFI**

**DR. LABIB ARAFEH**

**YACOUB SABATIN**

1

### One-Dimensional Arrays

- There are times when we need to store a complete list of numbers or other data items!
- We could do that by:
  - Creating as many *individual variables* as would be needed for the job, **BUT** this is a hard and tedious process.
  - Suppose we want to **read in five numbers** and **print them out in reverse order**:

```
int a1,a2,a3,a4,a5;  
scanf("%d %d %d %d %d",&a1,&a2,&a3,&a4,&a5);  
printf("%d %d %d %d %d",a5,a4,a3,a2,a1);
```

2

- *Comments:*
  - *Doesn't look very pretty, and*
  - What if the problem was to read in 100 or more values and print them in reverse order?
- What we would really like to do is to use a name like **a[i]** where **i** is a **variable** representing an **index** which specifies which particular value we are working with;
- This is the basic idea of an **array**.

3

- An **array** is a data structure containing a number of data values, all of which have the **same type**.
- As usual, in **C**, we have to declare an array before using it. → Similar to any variable declaration;
  - int a[5];** → Declares an array called **a** with **five** elements, **a[0] to a[4]**
- **a**

--	--	--	--	--

  - a[0] a[1] a[2] a[3] a[4]
- This is called array **subscripting** or **indexing**.

4

- Defining Arrays:

***datatype arrayName[size];***

- Since array lengths *may* need to be adjusted when program changes, it is a good idea to use **MACRO** to define length of array:

**#define ARRAY\_LENGTH 10**

**int a[ARRAY\_LENGTH]**

→ Declares array of 10 elements;

- All the elements of an array must be the **same data type (float, char, int)**

5

Examples:

***int ID[30];*** // Stores ID numbers of students

***float temperatures[31];*** // Store daily temperatures

***char name[20];*** // Stores a *character string*, character Strings in C are terminated by the null character '**\0**' in the last element.

***unsigned short int a[52];*** // Holds 52 unsigned short integer values;

6

C: Name of array (All elements of array have the same name)

c[0]  
c[1]  
c[2]  
c[3]  
c[4]  
c[5]  
c[6]  
c[7]  
c[8]  
c[9]  
c[10]  
c[11]

-45
6
0
72
1543
-89
0
62
-3
1
6453
78

Position number of the element within array c

Note: `i = 0;`

While (`i < N`)

`a[i++] = 0;` // `a[i]` will be initialized with 0,

`a[++i] = 0;` //  $\otimes$ : 0 would be assigned to `a[1]`!

7

C\_Prog. Spring 2011

## Array Initialization

- Arrays can be initialized when declared:  
`int a[5] = {1,2,3,4,5};`
- If **initializer** shorter than array, remaining elements are given value 0:  
`int a[5] = {1,2,3}; /* initial value {1,2,3,0,0} */`
- Length of array may be omitted, if **initializer** is present:  
`int a[] = {1,2,3,4,5};`  
`int a[6] = {0}; // Initial value of a is: {0, 0, 0, 0, 0, 0}`
- To find out the **size** of an array in **Bytes**, we can use the *sizeof operator*:
  - *For an array of 10 integers, sizeof(a) = 20 OR 40!*

8

Why???

```

/* Reverses a series of N numbers */
#define N 10 // Defines length of array to be 10
int a[N], i; // Declares an array a of N length
          of type integer
printf("Enter %d numbers: ", N);
for (i = 0 ; i < N ; i++)
    scanf("%d", &a[i]); // Reads data into array a
printf("In reverse order:");
for (i = N-1 ; i >= 0 ; i--) // Array is indexed 0 to N-1
    printf(" %d\n", a[i]);

```

9

Examples:

```

for (i = 0; i < N; i++)
    a[i] = 0; // Clears array a;
for (i = 0; i < N; i++)
    scanf("%d", &a[i]); // Reads data into array a;
for (i = 0; i < N; i++)
    Sum+= a[i]; // Sums the elements of array a;
for (i = 0; i < N; i++)
    a[i] = i+1; // Sets the array elements of a
                  certain value (i+1);
for (i = 0; i <N; i++)
    printf(" %d\n", a[i]); // Prints out all elements
                          of array a;

```

10



```

char arrayChar[] = {'A','r','r','a','y','\0'};
int arrayInt[5] = {1,2,4,8,16};
float arrayFloat[3] = { 1.24 , 2 , 4.68756 };
double arrayDouble[2];
int arrayInt2D[ ][4] = {1,6,3,7, 0,3,8,9, 2,5,2,3};
arrayDouble[0] = 23.23456532;
arrayDouble[1] = 2.3422267;

printf("The size of arrayChar is
      %d\n", sizeof(arrayChar));
printf("The size of arrayInt is
      %d\n", sizeof(arrayInt));
printf("The size of arrayFloat is
      %d\n", sizeof(arrayFloat));

```

**Output:**

```

The size of arrayChar is 6
The size of arrayInt is 20
The size of arrayFloat is 12
The size of arrayDouble is 16
The size of arrayInt2D is 48
The size of arrayInt2D[0] is 16

```

13

**Example:**

```

/* Checks numbers
for repeated digits */
#include <stdio.h>
#define TRUE 1
#define FALSE 0
typedef int Bool;

main()
{
    //all false
    Bool digit_seen[10] = {0};
    //holds the digit and the index
    int digit;
    long int n; //here is the number
    printf("Enter a number: ");
    scanf("%ld", &n);

    while (n > 0)
    {
        digit = n % 10; //extract the last digit
        if (digit_seen[digit])
            break;
        digit_seen[digit] = TRUE;
        n /= 10; /*move to the next digit by
        getting rid of the last digit
        from the right */
    }

    if (n > 0) //means the digit was seen
        printf("Repeated digit\n\n");
    else
        printf("No repeated digit\n\n");

    return 0;
}

```

14

Operators						Associativity	Type
[]	()					left to right	highest
++	--	!	(type)			right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical and
						left to right	logical or
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma
Operator precedence.							

15

C\_Prog. Spring 2011

## Multidimensional Arrays

- An array can have 2 or more dimensions

Example:

```
double x[8][3];    /* 8 Rows, 3 Columns */
```

→ This declares an 8 x 3 array, or an array of 24 elements

- A multidimensional array can be initialized
- ```
int x[3][3] = { {1, 4, 3}, {4, 8, 2}, {7, 9, 11} };
OR
```

```
int x[3][3] = {1, 4, 3,
               4, 8, 2,
               7, 9, 11};
```

- Defining multiple arrays of same type:
  - Example: `int b[ 100 ], x[ 27 ];`
- **Tables (Matrices)** with rows and columns (**m** by **n** array):

|       | Column 0             | Column 1             | Column 2             | Column 3             |
|-------|----------------------|----------------------|----------------------|----------------------|
| Row 0 | <code>a[0][0]</code> | <code>a[0][1]</code> | <code>a[0][2]</code> | <code>a[0][3]</code> |
| Row 1 | <code>a[1][0]</code> | <code>a[1][1]</code> | <code>a[1][2]</code> | <code>a[1][3]</code> |
| Row 2 | <code>a[2][0]</code> | <code>a[2][1]</code> | <code>a[2][2]</code> | <code>a[2][3]</code> |

Array name      Row subscript      Column subscript

17

C\_Prog. Spring 2011

- **Example:**

```
double x[8][3]; //Declares a 8 x 3 array: 8 Rows, 3 Columns
```
- A multidimensional array can be initialized:
 

```
int x[3][3] = { {1, 4, 3}, {4, 8, 2}, {7, 9, 11} };
int x[3][3] = {1, 4, 3,
               4, 8, 2,
               7, 9, 11};
```

|   |   |
|---|---|
| 1 | 2 |
| 3 | 4 |

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

  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero
 

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

|   |   |
|---|---|
| 1 | 0 |
| 3 | 4 |
- Referencing elements: Specify row, then column as
 

```
printf( "%d", b[ 0 ][ 1 ] );
```

18

C\_Prog. Spring 2011

## sizeof() again

- Find out the maximum of "rows" there are in a **2D array**:

How many **rows** are there in `arrayInt2D[][4]` mentioned before?

`int arrayInt2D[ ][4] = {1,6,3,7, 0,3,8,9, 2,5,2,3};`  
12 elements here.

`sizeof(arrayInt2D) / sizeof(arrayInt2D[0])`

→ `48/16 = 3`

Think of this calculation as:

"size of entire array / size of a column"

- So, there are 3 **rows** & 4 **columns** in the 2D array.

19

- In C, a **string** is simply an array of type **char**
- *Declaring a string:*  
`char str_var[20];` /\* Creates a variable that can hold a string from 0 to 19 characters long \*/
- *Initializing a string*  
`char str_var[20] = "Text goes here";` /\* Creates the following in memory:  
`Text goes here\0?????` (20 characters)  
→ '0' is a null character, indicates the end of a string  
→ The content following the null character is **ignored** by the **string functions** in C

20

- Array of strings....

```
char names[7][6] = { "Ryan",
                    "Tom",
                    "Chad",
                    "Jackie",
                    "Tara",
                    "Lori",
                    "Kelly" };
```

How many  
strings

Amount of characters any  
string may have.  
If this number is left out,  
compiler will use longest name  
length as maximum length

```
printf("%s", names[0]);    /* print Ryan */
printf("%s", names[3]);    /* print ????? */
```

21

## Constant Arrays

- Arrays can be defined as constants using the keyword *const*
- Used if array is not supposed to be modified by program

```
const int age [ ] = {20,30,40,50};
```

22

# Examples

23

```
// Try this code that initializes an array with an initializing list
#include <stdio.h>
int main()
{
    int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
    int i; /* counter */
    printf( "%s%13s\n", "Element", "Value" );
    /* output contents of array in tabular format */
    for ( i = 0; i < 10; i++ ) {
        printf( "%7d%13d\n", i, n[ i ] );
    }
    return 0;
}
```

24

| Element | Value |
|---------|-------|
| 0       | 32    |
| 1       | 27    |
| 2       | 64    |
| 3       | 18    |
| 4       | 95    |
| 5       | 14    |
| 6       | 90    |
| 7       | 70    |
| 8       | 60    |
| 9       | 37    |

25

```
// Try this code that initializes elements of array s to even integers from 2 – 20
#include <stdio.h>
#define SIZE 10
int main()
{
    int s[ SIZE ]; /* array s has 10 elements */
    int j;          /* counter */
    for ( j = 0; j < SIZE; j++ ) /* set the values */
    {
        s[ j ] = 2 + 2 * j;
    }
    printf( "%s%13s\n", "Element", "Value" );
    for ( j = 0; j < SIZE; j++ ) {
        printf( "%7d%13d\n", j, s[ j ] ); }
    return 0; }
```

| Element | Value |
|---------|-------|
| 0       | 2     |
| 1       | 4     |
| 2       | 6     |
| 3       | 8     |
| 4       | 10    |
| 5       | 12    |
| 6       | 14    |
| 7       | 16    |
| 8       | 18    |
| 9       | 20    |

27

```
// Try the code that Computes the sum of the elements of the array
#include <stdio.h>
#define SIZE 12
int main()
{
    int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
    int i;          /* counter */
    int total = 0; /* sum of array */
    for ( i = 0; i < SIZE; i++ )
        total += a[ i ];
    printf( "Total of array element values is %d\n", total );
    return 0;
}
```

Total of array element values is 383

28

```

// Try the code that demonstrates Student Poll
#include <stdio.h>
#define RESPONSE_SIZE 40 /* define array sizes */
#define FREQUENCY_SIZE 11 /* to save expected
                           answers (0-10) */

/* function main begins program execution */
int main()
{
    int answer; /* counter */
    int rating; /* counter */
    int frequency[ FREQUENCY_SIZE ] = { 0 }; //Reset the
   11 elements

    int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9,
    7, 8, 10, 1, 6, 3, 8, 6, 10, 0, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6,
    6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
}

```

```

/* for each answer, select value of an element of array
responses and use that value as subscript in array
frequency to calculate the frequency of each answer. */
for ( answer = 0; answer < RESPONSE_SIZE; answer++ )
{
    ++frequency[ responses [ answer ] ]; //add this response
} /* end for */
printf( "%s%17s\n", "Rating", "Frequency" );
for ( rating = 0; rating < FREQUENCY_SIZE; rating++ )
{ printf( "%6d%17d\n", rating, frequency[ rating ] ); }

return 0;
}

```

| Rating | Frequency |
|--------|-----------|
| 0      | 1         |
| 1      | 2         |
| 2      | 2         |
| 3      | 1         |
| 4      | 2         |
| 5      | 5         |
| 6      | 11        |
| 7      | 5         |
| 8      | 7         |
| 9      | 1         |
| 10     | 3         |

```

// Try the code that prints a Histogram
#include <stdio.h>
#define SIZE 10
int main()
{
    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
    int i; /* outer counter */
    int j; /* inner counter */
    printf( "%s%13s%17s\n", "Element", "Value",
            "Histogram" );
    for ( i = 0; i < SIZE; i++ )
    {
        printf( "%7d%13d      ", i, n[ i ] );
        for ( j = 1; j <= n[ i ]; j++ )
        { /* print one bar */
            printf( "%c", '*' );
        } // end of the inner for loop
    }
}

```

```

        printf( "\n" ); /* start next line of output */
    } /* end outer for */
    return 0;
}

```

| Element | Value | Histogram |
|---------|-------|-----------|
| 0       | 19    | *****     |
| 1       | 3     | ***       |
| 2       | 15    | *****     |
| 3       | 7     | *****     |
| 4       | 11    | *****     |
| 5       | 9     | *****     |
| 6       | 13    | *****     |
| 7       | 5     | *****     |
| 8       | 17    | *****     |
| 9       | 1     | *         |

```

// Try the code that deals with strings
#include <stdio.h>
int main()
{
    char string1[ 20 ]; /* reserves 20 characters */
    char string2[] = "string literal"; //reserves 15 characters
    int i; /* counter */
    printf("Enter a string: ");
    scanf( "%s", string1 ); /* no need for & sign when we deal with
                             strings */
    printf("string1 is: %s\nstring2 is: %s\n", string1, string2 );
    printf("string1 with spaces between characters is:\n");
    for ( i = 0; string1[ i ] != '\0'; i++ ) //Loop until a null character
    { printf( "%c ", string1[ i ] ); } //notice the " "

```

33

```

    printf( "\n" );
    return 0;
}

```

```

Enter a string: Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o

```

34

## Notes about previous example 1/2

- No need for & sign when we deal with strings in scanf: `scanf( "%s", string1 );`
- since `string1` is an array name, it's treated as a pointer automatically (more about pointers later).
- A string read using `scanf` will never contain white spaces.
- Instead we can use `gets(string1);` from `string.h` header file.

35

## Notes about previous example 2/2

- `gets` doesn't skip white spaces.
- `gets` reads until it finds a *new-line character*, `gets` discards the new-line character and the *null character* takes the place of it.
- Another function in `string.h` header file is `puts` which can be used to print strings.
- `puts` prints the string contents and then writes an additional new-line character after it.

36

## Example:

```

#include <stdio.h>
#include <string.h>
int main()
{
    char FullName[ 20 ];
    printf("Please enter your full name ");
    gets(FullName); /* Will read the user input until the user
                    presses enter, and
                    then store it in FullName. */
    printf("Welcome ");
    puts(FullName); /* Will print a new-line automatically
                    AFTER the FullName. */

    return 0;
}

```

37

- **Problem:** Create 3 arrays, Read data into the first two, Subtract each element in the first array from the corresponding element in the second array. Store the differences in the third array. Print all the 3 arrays. The relationships between the arrays are as bellow:

$$\text{ArrayC}[i] = \text{ArrayB}[i] - \text{ArrayA}[i]$$

|        |    |    |    |     |     |
|--------|----|----|----|-----|-----|
| ArrayA | 10 | 21 | 13 | 17  | 19  |
| ArrayB | 19 | 44 | 72 | 3   | 0   |
| ArrayC | 9  | 23 | 59 | -14 | -19 |

38

**Solution:**

1. LOOP all array elements
  1. INPUT ArrayA[i];
2. LOOP all array elements
  1. INPUT ArrayB[i];
3. LOOP all array elements
  1.  $\text{ArrayC}[i] = \text{ArrayB}[i] - \text{ArrayA}[i]$ ;
4. LOOP all array elements
  1. Output ArrayA[i], ArrayB[i], ArrayC[i]
5. STOP

39

**Make your programs efficient, interactive, and concise as possible**

- Q1) Develop a C program for the previous problem.
- Q2) Develop a C program to output the odd elements of a read 10-element array?
- Q3) Develop a C program that inputs a list of numbers, letting the user decide how many. Output the list backwards, but only output numbers greater than 20?

40

**Q4) Develop the a C program for a luxury-car dealer who wants to keep a monthly log of the number of sales made for each of six different luxury cars. The dealer wants a printout at the end of the year showing the total number of sales of each car and the best sales month for the year: a) for each car, and b) for the total sales of all cars in each month?**

**Submission Date: \_\_\_\_\_**

**Documentation is very important**

**Include screen shots of the output!**

**Submit softcopy + hardcopy**