

Advanced Programming COEN 11

Lecture 2

1

Arrays

- An array is a collection of two or more adjacent memory cells, called array elements
- An array is associated with a symbolic name

1-2

Declaring Arrays

- To declare an array, determine
 - The name
 - The type of the elements
 - The size of the array
- Example

```
double x[8];
```

Array with 8 doubles in memory,
which are referenced by the name x

1-3

Referencing Arrays

- To reference the array
 - Use the name of the array
 - Example: x

1-4

Referencing Arrays

- To reference each individual element
 - Use the name of the array and the index of the element
 - The index is given by the subscript in brackets and goes from 0 to size-1
 - Example: $x[0], x[1], x[2], \dots, x[7]$

1-5

Referencing Arrays

- Subscripts are integers
 - Constants, variables, or expressions
 - Examples
 - $x[4] = x[5];$
 - $x[i] = 0;$
 - $b = x[i];$
 - $x[i + j] = a;$

1-6

Referencing Arrays

- Subscripts must be within the right range
 - If the array has size SIZE
 - Subscript range is zero to SIZE - 1
 - Using an out-of-range subscript may
 - Produce wrong results or
 - Crash the program → run-time error

1-7

The Eight Elements of Array x

- Example
 - double array[8];

1-8

The Eight Elements of Array x

- Examples of statements using array x

```
printf ("%lf", x[0]);  
x[3] = 25.0;  
sum = x[0] + x[1];  
sum += x[2];  
x[3] += 1.0;  
x[2] = x[0] + x[1];
```

1-9

Initializing Arrays

- Statically at declaration

```
int x[3] = {10, 2, 3};  
int y[ ] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  
char vowels[ ] = {'a', 'e', 'i', 'o', 'u'};
```

1-10

Initializing Arrays

- Dynamically at run time

➢ Use a loop!

➢ Example:

```
#define SIZE 10  
...  
int x[SIZE];  
...  
for (i = 0; i < SIZE; i++)  
    x[i] = i;
```

1-11

Multidimensional Arrays

- Arrays with 2 or more dimensions

- Used to represent

➢ Tables

➢ Matrices

➢ Any two dimensional objects

1-12

Multidimensional Arrays

❑ Declaration

- Name,
- Type,
- Size of each dimension

❑ Examples

```
double matrix[20][20];  
int multi[10][10][5];
```

1-13

Multidimensional Arrays

❑ Initialization

- Static: Values are grouped by dimension!

```
int matrix[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};  
int matrix[3][3][3] = {{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},  
                      {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},  
                      {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}};
```

1-14

Multidimensional Arrays

❑ Initialization

- Dynamic: Use nested loops!

- Example

...

```
int matrix[3][4];  
...  
for (i = 0; i < 3; i++)  
    for (j = 0; j < 4; j++)  
        matrix[i][j] = i + j;
```

1-15

Functions and Arrays

- ❑ The name of an array represents its address in memory

- ❑ Passing an array as an argument to a function is done by reference

- Example:

```
int array[50];  
int value;  
...  
return_value = search (array, value);
```

Function and Arrays

- ❑ Functions receiving arrays as arguments need to specify all but the first dimension size

- ❑ Example with a 1D array:

```
int  
search (int array [ ], int value)  
{  
    ...  
}
```

Function and Arrays

- ❑ Example with a 2D array:

```
int  
search (int array [ ][NCOLS], int value)  
{  
    ...  
}
```

Character Strings

- ❑ Characters

➢ Type char - 1 byte

```
char c = 'c';  
char d = 65;
```

➢ Input/Ouput - %c

```
printf ("%c", c); or putchar (c);  
scanf ("%c", &c); or c = getchar ();
```

Character Strings

- ❑ A character array is an array in which the individual elements are stored as characters

- ❑ A character string is a character array in which the last element is a character '\0', which has an ASCII integer equivalent to zero.

String Definition

- Character string constants are enclosed in double quotes
 - "info.txt"
 - "r"
 - "15762"

String Definition

- A character string can be initialized using string constants

```
char filename[12] = "info.txt";
char filename[ ] = "info.txt";
char filename[ ] = {'i', 'n', 'f', 'o', ' ', 't', 'x', 't', '\0'};
```

String Initialization

- A string can also be initialized with a word that is read from the keyboard

```
char word[100];
...
scanf ("%s", word);
```

String Output

- Use printf

```
char string[100];
...
printf ("String: %s\n", string);
```

Arrays of Strings

- ❑ A string is an array of characters
- ❑ An array of strings is a 2D array of characters
- ❑ Example, a list of 10 names, each with at most 19 characters:
`char names[10][20];`

String Functions

- ❑ `#include <string.h>`
- `strlen (s)` - lenght of the string *s*
`strcpy (s, t)` - copy *t* to *s*
`strncpy (s, t, n)` - copy *n* characters from *t* to *s*
`strcat (s, t)` - concatenates *t* to the end of *s*
`strncat (s, t, n)` - concatenates *n* characters from *t* to the end of *s*
`strcmp (s, t)` - compares *s* and *t* (*<*: -1, *==*: 0, *>*: 1)
`strncmp (s, t, n)` - compares at most *n* characters of *s* to *t*

Details...

- ❑ String functions need to receive strings, which end with a zero ('\0') character
- ❑ Careful not to overflow the receiving string with `strcpy` and `strcat`

String Conversions

- ❑ **String to Number**
 - Function `scanf` converts a string typed into a number:
- `scanf ("%d", &int_num);`
`scanf ("%f", &float_num);`

String Conversions

□ String to Number

➤ Function atoi converts a string to an integer

```
int      x;  
char    str[ ] = "123";  
...  
x = atoi (str);
```

→ Also, strtoint and strtodouble

String Conversions

□ Number to string

➤ Function sprintf converts number to strings

```
int      x = 123;  
float   y = 45.46;  
char    str[20];  
...  
sprintf (str, "%d %f", x, y);
```