

Computer Programming 143 – Lecture 16

Arrays IV

row 

column 

Faculty of Engineering



Student Grades $[i][j]$

	0	1	2	3
Student Grades 0	77	68		
Student Grades 1			89	
Student Grades 2				81

6.11 Multiple-Subscripted Arrays

Multiple-Subscripted Arrays

- Tables with rows and columns (**m** by **n** array)
- Like matrices: specify row, then column

	col 0	col 1	col 2	col 3
row 0	c[0][0]	c[0][1]	c[0][2]	c[0][3]
row 1	c[1][0]	c[1][1]	c[1][2]	c[1][3]
row 2	c[2][0]	c[2][1]	c[2][2]	c[2][3]
row 3	c[3][0]	c[3][1]	c[3][2]	c[3][3]

Diagram illustrating the structure of a multiple-subscripted array (matrix) with 4 rows and 4 columns. The array is represented as a table with cells containing indices like `c[0][0]`, `c[0][1]`, etc.

Annotations below the table:

- Array name: `c` (indicated by an arrow pointing to the first character of the first cell).
- Row subscript: `[0]` (indicated by an arrow pointing to the first bracketed index of the first cell).
- Column subscript: `[1]` (indicated by an arrow pointing to the second bracketed index of the first cell).

6.11 Multiple-Subscripted Arrays

Initialisation

#rows
#columns

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

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

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

c[0][1]

1	2
3	4

1	0
3	4

6.11 Multiple-Subscripted Arrays

Referencing elements

- Specify row, then column

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

Refer to Fig. 6.16 and 6.17 in Deitel & Deitel for examples

Array recap

- Declaring arrays (1D and 2D): lectures 13 and 16
- Using arrays to process strings: lecture 14
- Searching and sorting arrays: lectures 14 and 15
- Passing arrays to functions: call-by-reference (lecture 14)

```
int apples[5];  
apples[4] = 55;  
'\0'      "apples"
```

Debugging methods

Using printf () statements

- Print out the data in a particular variable to see if it is as expected
- To see if a certain line is executed:
`printf( "%s:%d\n", __FILE__, __LINE__ );`
- To interrupt the program at specific point:
`exit(-1);` // *defined in stdlib.h library*

Use /*.....*/ comment blocks

- Block out large sections that may be leading to an error
- Unblock single statements / small sections until the error occurs

A1 questions

Structure: 2 hours, 7 questions , 74 marks

1. Short questions: variables, compiling, rand
2. Loops
3. Functions
4. Logic in loops; enums
5. Logic, psuedo code and flow diagrams
6. Recursive function
7. Arrays