

Programming with C I

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

Basic Terminology

➤ **data structure**

- a composite of related data items stored under the same name

➤ **array**

- a collection of data items of the same type

Declaring and Referencing Arrays

➤ **array element**

- a data item that is part of an array

➤ **array subscript**

- a value or expression enclosed in brackets after the array name, specifying which array element to access

Table Statements That Manipulate Array x

Statement	Explanation
<code>printf("%.1f, x[0]);</code>	Displays the value of <code>x[0]</code> , which is 16.0.
<code>x[3] = 25.0;</code>	Stores the value 25.0 in <code>x[3]</code> .
<code>sum = x[0] + x[1];</code>	Stores the sum of <code>x[0]</code> and <code>x[1]</code> , which is 28.0 in the variable <code>sum</code> .
<code>sum += x[2]</code>	Adds <code>x[2]</code> to <code>sum</code> . The new <code>sum</code> is 34.0.
<code>x[3] += 1.0;</code>	Adds 1.0 to <code>x[3]</code> . The new <code>x[3]</code> is 26.0;
<code>x[2] = x[0] + x[1];</code>	Stores the sum of <code>x[0]</code> and <code>x[1]</code> in <code>x[2]</code> . The new <code>x[2]</code> is 28.0.

Array x

<code>x[0]</code>	<code>x[1]</code>	<code>x[2]</code>	<code>x[3]</code>	<code>x[4]</code>	<code>x[5]</code>	<code>x[6]</code>	<code>x[7]</code>
16.0	12.0	28.0	26.0	2.5	12.0	14.0	-54.5

Using **for** Loops for Sequential Access


```
for (i = 0; i < SIZE; ++i)  
    scores[i] = i * i;
```

Array scores

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
0	1	4	9	16	25	36	49	64	81	100

Sizeof and Arrays

 Operator sizeof returns the total bytes in the argument

 Total elements = sizeof(array) / sizeof(data-type)

```
int scores[MAX_SCORES];  
int scoresBytes = sizeof(scores); // MAX_SCORES * 4  
int scoresElements = sizeof(scores) / sizeof(int); // MAX_SCORES
```

 Sizeof does not return total bytes being used



You cannot use sizeof to determine the number of elements being used in a partially filled array.



Loading an Array



Be careful not to overflow

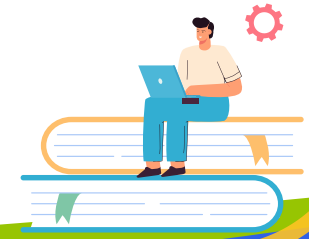


Do not read directly into array elements



```
// Example: Load array of scores checking for overflow
const int MAX_SCORES = 50;
int scores[MAX_SCORES];
int score, scoreCount;

// Load into array, check for too many
for (scoreCount=0; scanf("%d", &score) == 1; scoreCount++) {
    // scoreCount here is one less than actual scores read
    if (scoreCount >= MAX_SCORES) {
        printf("Unable to store more than %d scores. \n", MAX_SCORES);
        exit(1); // stdlib: exit program even in nested function
    }
    scores[scoreCount] = score;
}
```



Multidimensional Arrays

- 💡 Arrays with more than one dimension
 - 💡 Declaration: Additional sizes each enclosed in brackets
- 💡 Two dimensions
 - 💡 Table or 'array of arrays' `int a[3][4];`
 - 💡 Requires two subscripts – row and column

	Column 0	Column 1	Column 2	Column 3	
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	Column index
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	

Array name Row index

Initializing Multidimensional

- » Nested lists
 - » Unspecified values set to zero
- » 2D Example:

```
int nums[4][5] = { {10, 6, -7, 13, 28},  
                  {10, 5, 44, 8},  
                  {33, 20, 1, 0, 14},  
                  {2, 66, 25, 37, 1}  
                };
```



Loading a Two-dimensional Array

// assumes data matches table dimensions

```
int row, col, value;
```

```
for (row=0; row<rows; row++)
```

```
    for (col=0; col<cols; col++) {
```

```
        scanf("%d", &value);
```

```
        a[row] [col] = value;
```

```
    }
```

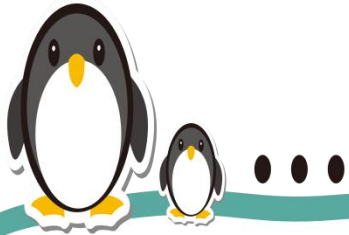


for-loops with Arrays

- 👉 Natural counting loop
 - 👉 Naturally works well 'counting thru' elements of an array
- 👉 General form for forward direction
 - 👉 `for (subscript = 0; subscript < size; subscript++)`
- 👉 General form for reverse direction
 - 👉 `for (subscript = size-1; subscript >= 0; subscript--)`



for-loops with Arrays Examples



```
int scoreSub;  
// Print forward  
for (scoreSub = 0; scoreSub<12; scoreSub++)  
    printf("Score %d is %d\n", scoreSub+1,  
          scores[scoreSub]);  
// Print backward, in reverse  
for (scoreSub = 11; scoreSub >= 0;  scoreSub--)  
    printf("Score %d is %d\n", scoreSub+1,  
          scores[scoreSub]);
```

```
Score 1 is 56  
Score 2 is 52  
Score 3 is 80  
Score 4 is 74  
...  
Score 12 is 87
```

```
Score 12 is 87  
Score 11 is 97  
Score 10 is 86  
Score 9 is 80  
...  
Score 1 is 56
```

56
52
80
74
70
95
92
94
80
86
97
87

Uses of Defined Constant

» Use everywhere size of array is needed

» In for-loop for traversal:

```
int score;  
for (score=0; score<NUMBER_OF_STUDENTS; score++)  
    printf("%d\n", scores[score]);
```

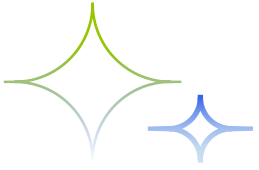
» In calculations involving size:

```
lastIndex = NUMBER_OF_STUDENTS - 1;  
lastScore = scores[NUMBER_OF_STUDENTS - 1];
```

» When passing array a function:

```
total = sum_scores(scores, NUMBER_OF_STUDENTS);
```





THE END

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu