Programming with C I

Fangtian Zhong
CSCI 112

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

2024.02.16

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
- **subscripted variable**
  - a variable followed by a subscript in brackets, designating an array element
- **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
printf("%.1f, x[0]);	Displays the value of x[0], which is 16.0.
x[3] = 25.0;	Stores the value 25.0 in x[3].
sum = x[0] + x[1];	Stores the sum of x[0] and x[1], which is 28.0 in the variable sum.
sum += x[2]	Adds x[2] to sum. The new sum is 34.0.
x[3] += 1.0;	Adds 1.0 to x[3]. The new x[3] is 26.0;
x[2] = x[0] + x[1];	Stores the sum of x[0] and x[1] in x[2]. The new x[2] is 28.0.

Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
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;
```

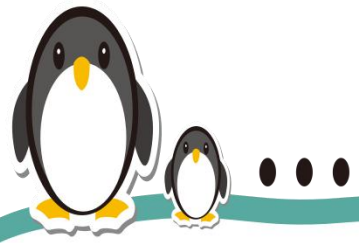
Array scores

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

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);
```

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

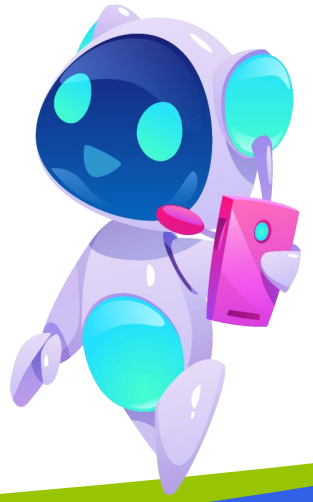
Column index

Array name Row index

Initializing Multidimensional

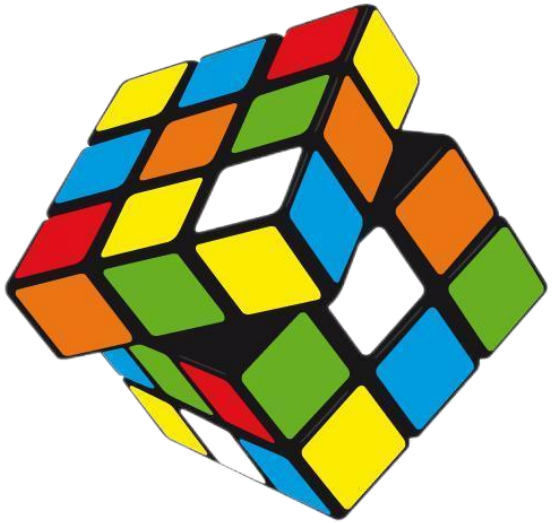
- Nested lists
  - Unspecified values set to zero
- 2D Example:

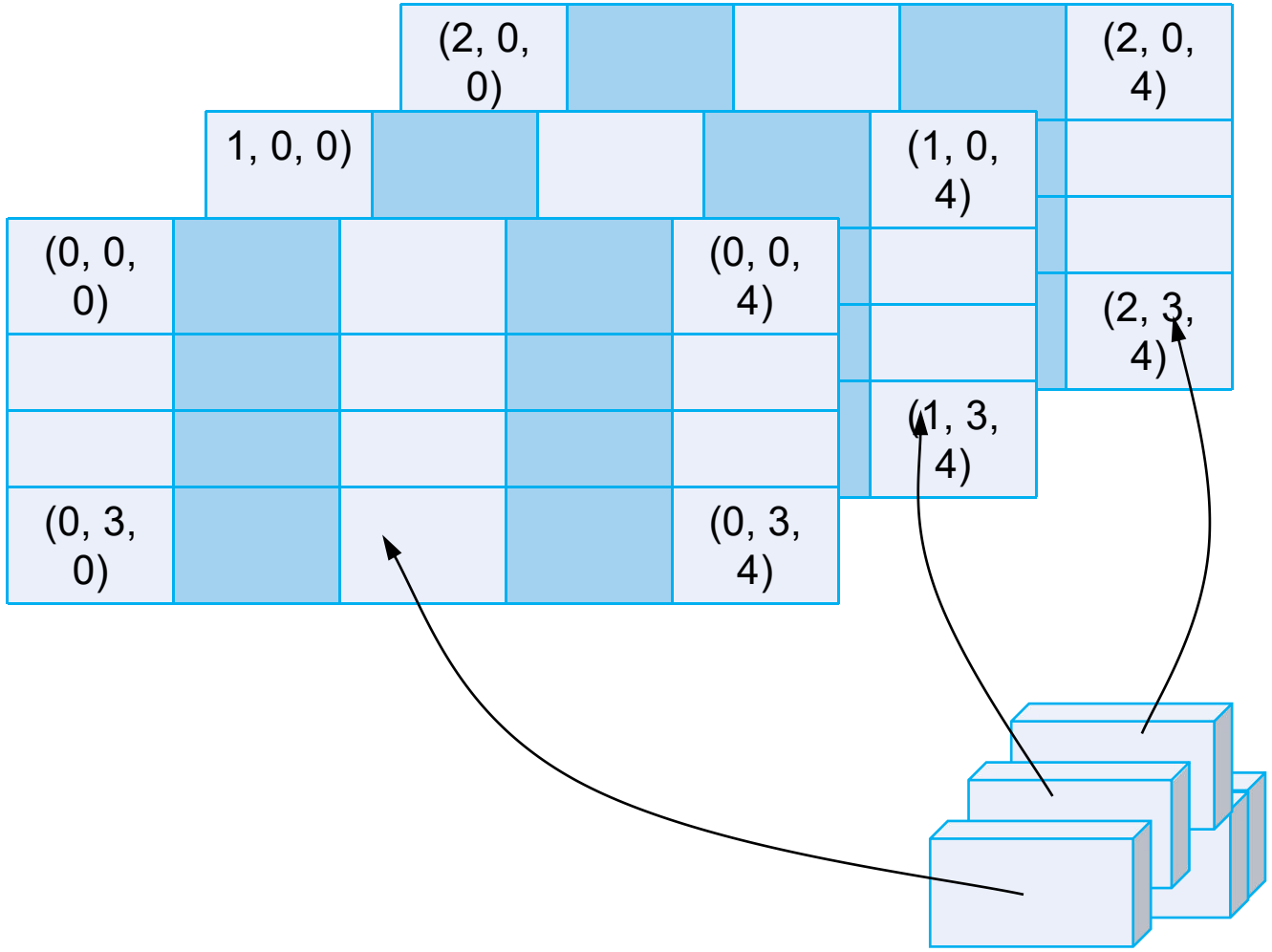
```
int nums[4] [5] = { {10, 6, -7, 13, 28},
                    {10, 5, 44, 8}.
                    {33, 20, 1, 0, 14},
                    {2, 66, 25, 37, 1}
                  }
```



Three-dimensional Visualization

```
int cubes[3] [3] [3];
```

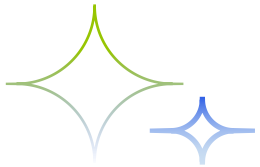




Multidimensional Array Parameters

Must specify size after first dimension

```
void scalar_multiply(int rows, int cols,
                                    int a[ ] [cols], int scalar) {
    // multiplies each element in array by scalar
    int row, col;
    for (row=0; row＜rows; row++)
        for (col=0; col＜cols; col++)
            a[row] [col] *=scalar;
}
```



MONTANA
STATE UNIVERSITY

THE END

Fangtian Zhong
CSCI 112

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

2024.02.16