C Arrays

Initializing Arrays, Accessing Elements By Index, Matrices

SoftUni Team

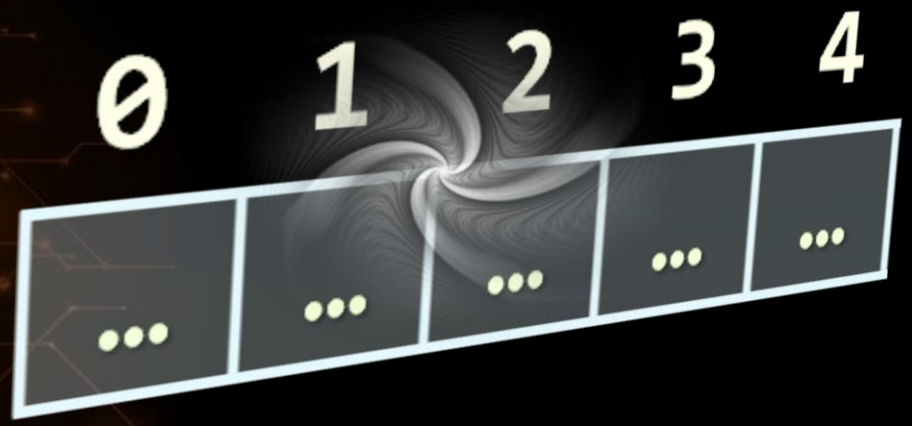
Technical Trainers

Software University

http://softuni.bg

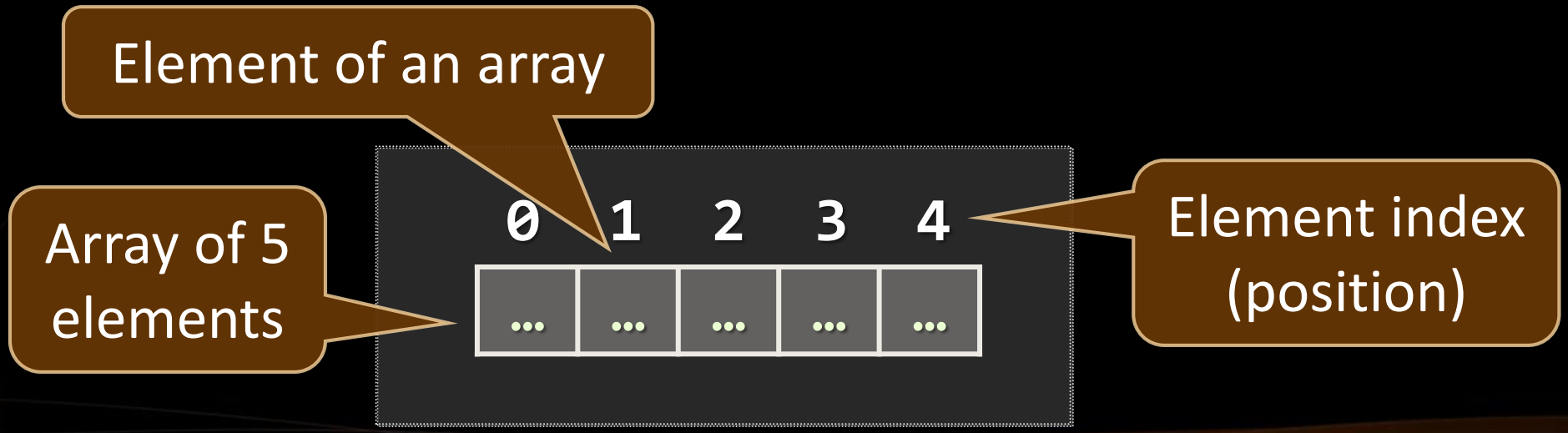
Table of Contents

Declaring and Creating Arrays



What are Arrays?

- An array is a sequence of elements
  - All elements are of the same type
  - The order of the elements is fixed
  - Has fixed size (must be stored in a separate variable)



Initializing Arrays

- An array can be defined in 3 ways:

1. Stack initialization with specified size

```
int arr[10];
arr[0] = 5;
arr[1] = 2;
```

2. Stack initializer list

```
int arr[] = { 1, 2, 3 };
printf("%d", arr[0]);
```

Initializing Arrays (2)

3. Heap initialization

```
#define SIZE 50
int main()
{
    int *arr = malloc(SIZE * sizeof(int));
    if (arr != NULL)
    {
        for (int i = 0; i < SIZE; i++)
            arr[i] = i * i;
        ...
        free(arr);
    }
}
```

Allocate 50 * 4 = 200 bytes on the heap

Free the memory after we're done using it

SOFTWARE UNIVERSITY FOUNDATION

Accessing Array Elements

Read and Modify Elements by Index

Accessing Array Elements by Index

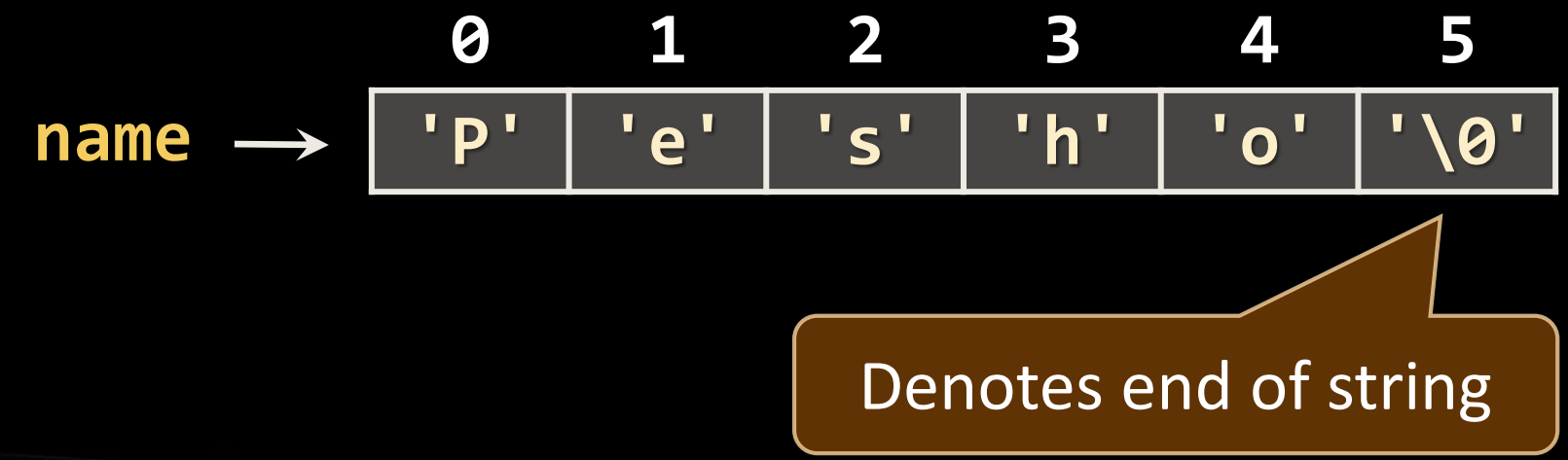
- Array elements are accessed
  - Using the square brackets operator `[]` (indexer)
- Array indexer takes element's index as parameter
  - The first element has index `0`
  - The last element has index `Length-1`
- Elements can be retrieved and changed by the `[]` operator

```
int arr[] = { 1, 2, 3 };
arr[0] = 10;
arr[1] = arr[2];
```

Char Arrays

- Strings in C are represented as character arrays
  - Size should be string length + 1
  - The last character should be reserved for null terminator (`'\0'`)

```
char name[] = { 'P', 'e', 's', 'h', 'o', '\0' };
```



Initializing Strings – Example

```
int main()
{
    char firstName[] = "pesho";
    firstName[0] = 'P';

    char title[4] = "Mr.";
    title[3] = '\0';

    char *lastName = "Goshov";
    printf("%s %s %s\n", title, firstName, lastName);

    lastName[0] = 'P'; // Error!

    return 0;
}
```

*lastName resides in read-only memory

String Array

- Strings are character arrays
  - String arrays are arrays of character arrays

```
int main()
{
    char *names[3];
    names[0] = "Penka";
    names[1] = "John";
    names[2] = "Ahmed";

    for (int i = 0; i < sizeof(names); i++)
        printf("%s\n", names[i]);

    return 0;
}
```

Buffer Overflow

- A buffer overflow occurs when the program writes outside the designated memory block, e.g.:

```
int arr[5];
arr[5] = 10; // Buffer overflow
```

  - Corrupts adjacent memory blocks
  - A very serious problem when developing in C
- Rule: Always stick to the bounds of the memory you work with
  - Validate you are not working outside of it

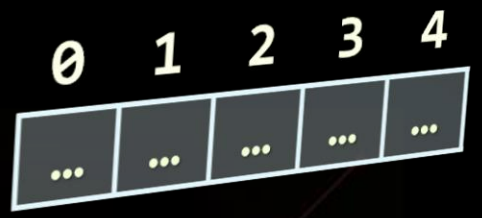
Accessing Array Elements – Examples

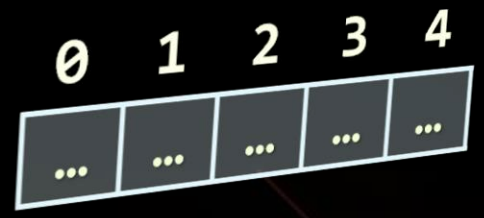

```
char *towns[] = { "Sofia", "Varna", "Bourgas" };
printf("%d\n", sizeof(towns)); // 24
printf("%s\n", towns[0]); // Sofia

towns[2] = "Pernik";
towns[2][0] = 'G'; // Gernik

// Buffer overflow! Will damage adjacent data
// towns[3] = "Plovdiv";
```

SOFTWARE UNIVERSITY FOUNDATION

Accessing Elements By Index

Live Demo





Arrays: Input and Output

Reading and Printing Arrays on the Console

Reading Arrays From the Console

- First, read from the console the length of the array

```
int n;
scanf("%d", &n);
```

- Next, create the array of given size **n** and read its elements:

```
int arr[n];
for (int i = 0; i < n; i++)
{
   scanf("%d", &arr[i]);
}
```

Last Digit as String – Example


```
int main()
{
    int n;
    scanf("%d", &n);
    int lastDigit = n % 10;

    char *digits[] = { "zero", "one", "two", "three", "four",
                       "five", "six", "seven", "eight", "nine" };

    printf("%s\n", digits[lastDigit]);

    return 0;
}
```

SOFTWARE UNIVERSITY FOUNDATION

Printing Arrays

Live Demo

Processing Arrays: for Statement

- Use `for` loop to process an array when
  - Need to keep track of the index
  - Processing is not strictly sequential from the first to the last
- In the loop body use the element at the loop index (`array[index]`):

```
for (int index = 0; index < array.Length; index++)
{
    squares[index] = array[index] * array[index];
}
```

Processing Arrays Using for Loop – Examples

- Printing array of integers in reversed order:

```
int arr[] = { 1, 2, 3, 4, 5 };
int length = sizeof(arr) / sizeof(int);
for (int i = length - 1; i >= 0; i--)
    printf("%d ", arr[i]);
// Result: 5 4 3 2 1
```

- Initialize array elements with their index:

```
for (int index = 0; index < length; index++)
{
    arr[index] = index;
}
```

SOFTWARE UNIVERSITY FOUNDATION

Processing Arrays

Live Demo

Bubble Sort – Example

```
#include <stdbool.h>
void bubble_sort(int array[], int length)
{
    bool hasSwapped = true;
    while (hasSwapped)
    {
        hasSwapped = false;
        for (int i = 0; i < length - 1; i++)
        {
            if (array[i] > array[i + 1])
            {
                int oldValue = array[i];
                array[i] = array[i + 1];
                array[i + 1] = oldValue;
                hasSwapped = true;
            }
        }
    }
}
```

Swap values

(example continues)

Bubble Sort – Example

```
int main()
{
    int arr[] = { 5, 3, 8, 7, 3, 2 };
    int length = sizeof(arr) / sizeof(int);

    bubble_sort(arr, length);

    for (int i = 0; i < length; i++)
    {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

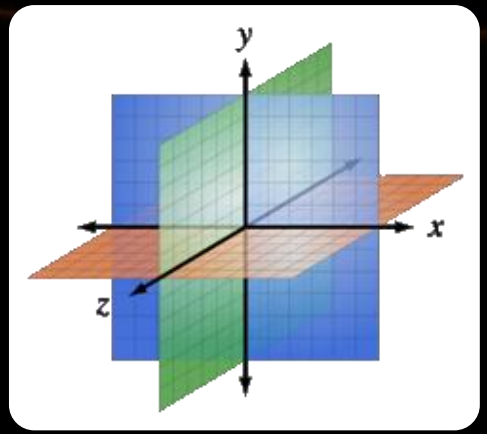
- More: https://en.wikipedia.org/wiki/Bubble_sort

SOFTWARE UNIVERSITY FOUNDATION

Bubble Sort

Live Demo

SOFTWARE UNIVERSITY FOUNDATION



Multidimensional Arrays

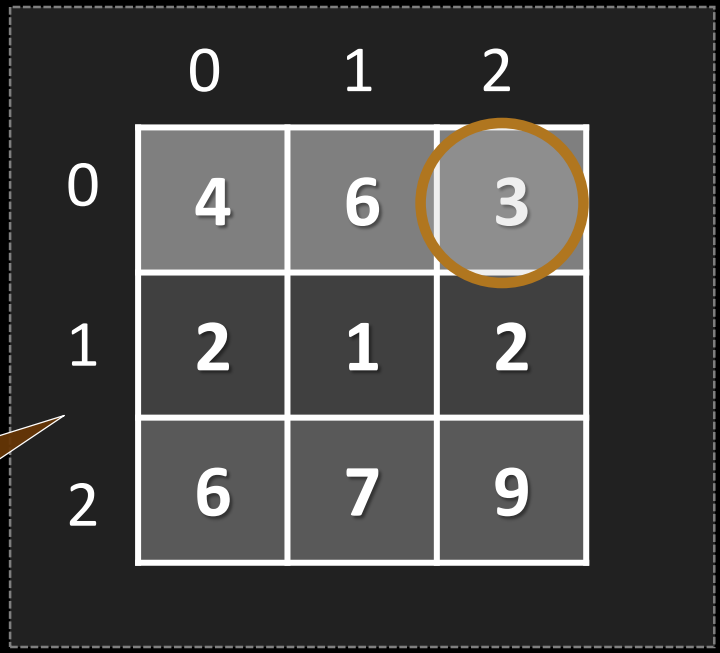
Using Array of Arrays, Matrices and Cubes

What is Multidimensional Array?

- Multidimensional arrays have more than one dimension
  - The most used multidimensional arrays are the 2-dimensional
    - Known as matrices or tables
- Declaring multidimensional arrays:

```
int intMatrix[3];
float floatMatrix[4][4];
char strCube[3][4][5];
```

One main array whose elements are arrays



Initializing Multidimensional Arrays

- Initializing with values multidimensional array:

```
int matrix[2][4] =
{
    {1, 2, 3, 4}, // row 0 values
    {5, 6, 7, 8}  // row 1 values
};
```

- Matrices are represented by a list of rows
  - Rows consist of list of values
- The first dimension comes first, the second comes next (inside the first)

Accessing Elements

- Accessing N-dimensional array element:

```
nDimensionalArray[index1][...][indexn]
```

- Getting element value example:

```
int array[3][3];
array[1][1] = 5;
int element11 = array[1][1];  // 5
```

- Setting element value example:

```
int array[3][4];
for (int row = 0; row < 3; row++)
  for (int col = 0; col < 4; col++)
    array[row][col] = row + col;
```

Reading a Matrix – Example

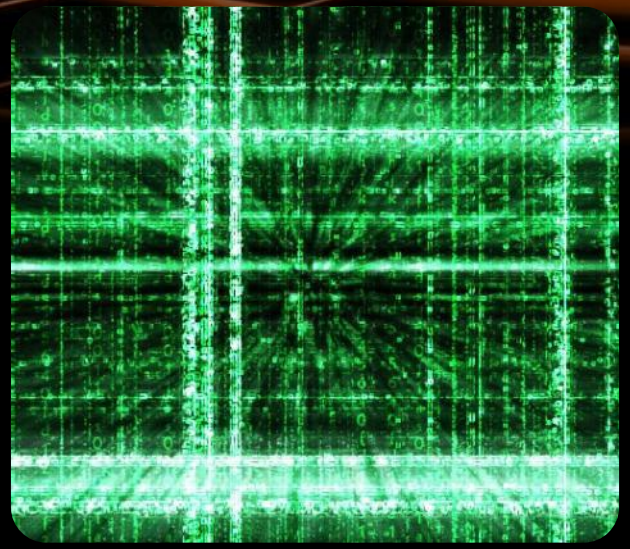

```
#define ROWS 2
#define COLS 2

int main()
{
    int matrix[ROWS][COLS];
    for (int row = 0; row < ROWS; row++)
    {
        for (int col = 0; col < COLS; col++)
        {
            scanf("%d", matrix[row][col]);
        }
    }
    return 0;
}
```

Printing Matrix – Example

```
int array[ROWS][COLS] = {
    {5, 2, 1, 4},
    {2, 10, 33, 1},
    {7, 6, 3, 0},
    {13, 15, 10, 9}
};

for (int row = 0; row < ROWS; row++)
{
    for (int col = 0; col < COLS; col++)
    {
        printf("%-5d", array[row][col]);
    }

    printf("\n");
}
```

List initializer

SOFTWARE UNIVERSITY FOUNDATION



Reading and Printing Matrices

Live Demo

Maximal Platform – Example

- Finding maximal sum of 2x2 platform

```
#include "stdio.h"
#include "limits.h";
#define ROWS 3
#define COLS 6

int main()
{
    int matrix[ROWS][COLS] = {
        {7, 1, 3, 3, 2, 1},
        {1, 3, 9, 8, 5, 6},
        {4, 6, 7, 9, 1, 0}
    };

    int bestSum = INT_MIN;
```

(example continues)

Maximal Platform – Example (2)

```
    for (int row = 0; row < ROWS - 1; row++)
    {
        for (int col = 0; col < COLS - 1; col++)
        {
            int sum = matrix[row][col] + matrix[row][col + 1] +
                matrix[row + 1][col] + matrix[row + 1][col + 1];
            if (sum > bestSum)
            {
                bestSum = sum;
            }
        }
    }

    printf("%d\n", bestSum);
    return 0;
}
```

Maximal Platform

Live Demo

Matrix Multiplication

Live Demo

C Programming – Arrays

Questions?

https://softuni.bg/trainings/1212/C-Programming-October-2015

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

  - Attribution: this work may contain portions from
    - "Programming Basics" course by Software University under CC-BY-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg