

Arrays

Lecture 28

Sections 8.1, 8.2, 8.4

Robb T. Koether

Hampden-Sydney College

Fri, Nov 8, 2019

- 1 Arrays
- 2 Array Declarations
- 3 Array Elements
- 4 Array Initialization
- 5 Assignment

Outline

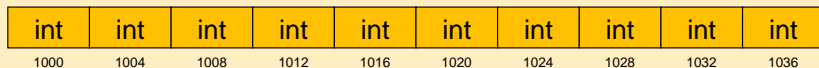
- 1 Arrays
- 2 Array Declarations
- 3 Array Elements
- 4 Array Initialization
- 5 Assignment

Arrays

- An **array** is an indexed collection of objects, all of a single type.
- An array is used to store a list of objects, such as a list of names.
- The array consists of individual array elements.
- The elements are stored in contiguous memory locations.
- The number of elements in the array is the **size** of the array.

Arrays

An Array of 10 `ints`



An array of 10 ints
(memory addresses)

Arrays

An Array of 10 `ints`

int	int	int	int	int	int	int	int	int	int
0	1	2	3	4	5	6	7	8	9

An array of 10 ints
(indexes)

Outline

- 1 Arrays
- 2 Array Declarations**
- 3 Array Elements
- 4 Array Initialization
- 5 Assignment

Array Declarations

Array Declaration

```
int arr[10];
```

- An array declaration must indicate
 - The name of the array (`arr`).
 - The fact that the object is an array (`[]`).
 - The type of element in the array (`int`).
 - The number of elements in the array (`10`).
- Given that information, the compiler will allocate the necessary memory and, perhaps, initialize the array.

Array Declarations

Array Declaration

```
type name[size];
```

- In the array declaration
 - *type* is the type of element in the array.
 - *name* is the name chosen for the array.
 - `[]` indicates that *name* is an array.
 - *size* is the number of elements in the array.

Examples of Array Declarations

Array Declarations

```
string name[30];      // 30 strings
int test_score[25];  // 25 ints
Point pentagon[5];    // 5 Point objects
Date birthdays[3]:    // 3 Date objects
```

Outline

- 1 Arrays
- 2 Array Declarations
- 3 Array Elements**
- 4 Array Initialization
- 5 Assignment

Accessing Array Elements

- An array is given a single name.
- Individual **elements** may be accessed through an **index**, written within square brackets `[]` (the **subscript operator**).
- The indexing begins with 0.
- In an array of size n , the valid indexes range from 0 to $n - 1$.
- Using an index outside that range may cause the program to crash.

Arrays

An Array of 10 `ints`

int	int	int	int	int	int	int	int	int	int
0	1	2	3	4	5	6	7	8	9

An array of 10 ints
(indexes)

Arrays

An Array of 10 `ints`

<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>	<code>int</code>
<code>arr[0]</code>	<code>arr[1]</code>	<code>arr[2]</code>	<code>arr[3]</code>	<code>arr[4]</code>	<code>arr[5]</code>	<code>arr[6]</code>	<code>arr[7]</code>	<code>arr[8]</code>	<code>arr[9]</code>

An array of 10 ints
(element names)

Example of an Array

- The `string` class stores characters in an array.

Example of an Array

- The `string` class stores characters in an array.
- The expression `str[i]` represents the i^{th} character (starting at 0) of the string `str`.

Example of an Array

- The `string` class stores characters in an array.
- The expression `str[i]` represents the i^{th} character (starting at 0) of the string `str`.
- If the size of the string is `n`, then the characters are `str[0]` through `str[n - 1]`.

Example of an Array

- The `string` class stores characters in an array.
- The expression `str[i]` represents the i^{th} character (starting at 0) of the string `str`.
- If the size of the string is `n`, then the characters are `str[0]` through `str[n - 1]`.
- In the same manner, if `arr` is an array of 10 integers, then the individual integers are accessed through the names `arr[0]`, `arr[1]`, ..., `arr[9]`.

Outline

- 1 Arrays
- 2 Array Declarations
- 3 Array Elements
- 4 Array Initialization**
- 5 Assignment

Array Initialization

- If the array elements are a fundamental type (e.g., `int`, `float`, `bool`), then the array is **uninitialized**.
- If the array elements are a created type (e.g., `Point`, `Rational`, `Date`, `string`) then the array is **initialized** using the default constructor of that type.

Array Initialization

Array Initialization

```
int a[10];           // Unitialized ints
double d[10];       // Unitialized doubles
string str[10];      // Initialized to ""
Rational r[10];      // Initialized to 0/1
Point pentagon[5];   // Initialized to (0, 0)
Date birthdays[3];   // Initialized to Jan 1, 1601
```

Array Initialization

Array Initialization

```
int a[10];  
int b[10] = {2, 4, 6, 8, 10, 1, 3, 5, 7, 9};  
int c[] = {2, 4, 6, 8, 10, 1, 3, 5, 7, 9};  
int d[10] = {2, 4, 6};
```

- The above example shows
 - An uninitialized array of size 10.
 - An initialized array of size 10.
 - An initialized array of implied size 10.
 - A partially initialized array of size 10.

Array Initialization

Array Initialization

```
int a[10];           // Uninitialized ints
for (int i = 0; i < 10; i++)
    a[i] = 10*i + 5;
```

- Use a **for** loop to initialize to 5, 15, 25, ..., 95.

Examples of Array Initialization

- Examples

- `ArrayInit.cpp`
- `CardHand.cpp`

Outline

- 1 Arrays
- 2 Array Declarations
- 3 Array Elements
- 4 Array Initialization
- 5 Assignment**

Assignment

Assignment

- Read Sections 8.1, 8.2, 8.4.