

# The C++ Language

Arrays

## Structure Data Types

- Collections of component items
  - Each item can be accessed individually
- In C++
  - Arrays
  - Structs
  - Unions
  - Classes

## Motivation

- Suppose you wanted to read in 100 temperatures. How would you store them all?

```
double temp00 = 0.0,  
       temp01 = 0.0,  
       temp02 = 0.0,  
       ...  
       temp99 = 0.0;
```

- What if you had to print all of them out? How would you do it?

3

Struble - Arrays

## Arrays

- A collection of elements of the same type
- Each element is accessed by *indexing*
- Mathematics

$temp_0 \ temp_1 \ temp_2 \ \dots \ temp_{99}$

- C++

`temp[0] temp[1] temp[2] ... temp[99]`

4

Struble - Arrays

## Arrays

- Memory

temp[0]	
temp[1]	
temp[2]	
temp[3]	
temp[4]	
...	...
temp[98]	
temp[99]	

5

Struble - Arrays

## Array Declarations

- An array must be declared before use
- Syntax

```
DataType Identifier [ IntegerConstant ] ;
```

- *DataType* is the type of each element
  - Can be simple or complex type, doesn't matter
- *Identifier* is the name used to access the array
- *IntegerConstant* is a literal or named constant
  - Defines the number of elements in the array.

6

Struble - Arrays

## Declaration Examples

```
// Function prototypes
...
// Constants
const int MAX_NAMES = 50;
const int MAX_GRADES = 100;

// Main program
void main() {
    int grades[MAX_GRADES];
    string names[MAX_NAMES];

    // computations in main()
}

// Function definitions and documentation
...

```

Array sizes  
should always  
be named  
constants!

7

Struble - Arrays

## Element Access

- Individual elements in an array are access by *indexing*
  - An *index* is an integer expression
- Syntax

`ArrayIdentifier [ IntegerExpression ]`
- Indices always start at 0 in C++!

8

Struble - Arrays

## Element Access Examples

```
void main() {
    int grades[MAX_GRADES];
    string names[MAX_NAMES];
    int i;

    // read in grades
    for (i = 0; i < MAX_GRADES; i++) {
        cin >> grades[i];
    }
    // Exercise: read in names
    // Print the 20th name
    cout << names[19] << endl;
}
```

9

Struble - Arrays

## Element Access Examples

```
void main() {
    int grades[MAX_GRADES];
    string names[MAX_NAMES];
    int i;

    // Initialize in reverse order
    for (i = 1; i <= MAX_GRADES; i++) {
        grades[MAX_GRADES - i] = i;
    }
}
```

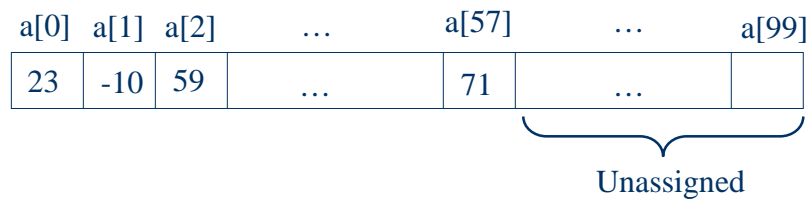
Any kind of integer  
expression.

10

Struble - Arrays

## Array Definitions

- Capacity, the largest number of elements the array can legally store.
- Usage, the largest number of elements actually in use.



11

Struble - Arrays

## Aggregate Operations

- Although you really want some things to work, they don't.

```
const int MAX_SIZE = 50;
int x[MAX_SIZE];
int y[MAX_SIZE];

x = y;           // No, does not copy arrays
x == y;         // No, does not compare elements in arrays
cout << x;      // No, you'll get an error (most of the time)
x + y;         // No, doesn't add elements in arrays
return x;       // No, does not return a copy of the array
someFunc(x);    // YES, we can pass arrays as parameters
```

12

Struble - Arrays

## Aggregate Operations

- Compare two arrays for equality

```
bool areEqual = (xUsage == yUsage);
for (int i = 0; areEqual && i < xUsage; i++) {
    areEqual = (areEqual && x[i] == y[i]);
}
```

- What about printing an array?
- Great candidates for turning into functions!
  - We will return to arrays and functions later

13

Struble - Arrays

## Common Mistake

- The most common mistake is to access an array out of bounds...

```
const int MAX_SCORES = 100;
int scores[MAX_SCORES];

for (int i = 0; i <= MAX_SCORES; i++) {
    scores[i] = 0;
}
```

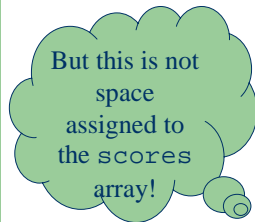
- Assigns 0 to `scores[100]`! What happens?

14

Struble - Arrays

## Common Mistake

- C++ allows you to index an array out of bounds
  - This is evil, but checking array bounds can be costly



scores[0]	
scores[1]	
scores[2]	
...	...
scores[99]	
scores[100]	

15

Struble - Arrays

## Common Mistake

- The results of accessing arrays out of bounds are unpredictable
  - May appear to run just fine
  - If you're lucky, will cause a runtime error
  - May corrupt another variable/value in your program
    - Unexpected infinite loops
    - Unexpected changes in values output
- If your program runs on your machine but not on the Curator, look for array out of bounds errors!!!

16

Struble - Arrays

## Array Initialization

- Most commonly use a loop to initialize arrays

```
const int MAX_SCORES = 100; // Capacity of scores array
int scores[MAX_SCORES];

for (int i = 0; i < MAX_SCORES; i++) {
    scores[i] = 0; // change 0 to whatever initial value
}
```

- Make this a function!!!!

17

Struble - Arrays

## Initialization at Declaration

- You can also initialize arrays at declaration
  - When you know size and values of array beforehand

```
const int MAX_HEIGHTS = 3;
int heights[MAX_HEIGHTS] = {5, 7, 8};
```

- Unassigned elements are initialized to 0

```
int heights[MAX_HEIGHTS] = {3}; // heights[1] == 0
                                // heights[2] == 0
```

18

Struble - Arrays

## Initialization at Declaration

- Providing too many initial values is a compile-time error.

```
// ERROR!  
int heights[MAX_HEIGHTS] = {3, 7, 8, 9};
```

19

Struble - Arrays

## Properties of Array Elements

- Once you specify an array element, it can be used just like any other variable of the specified data type:

```
int scores[MAX_SCORES];  
  
scores[0] = 90;  
total = total + scores[i];  
cout << "Your score is " << scores[38] << endl;  
Swap(scores[3], scores[10]);
```

20

Struble - Arrays

## Properties of Array Elements

```
void Swap(int& x, int& y) {  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

21

Struble - Arrays

## Arrays As Parameters

- Recall that entire arrays can be passed as parameters.
- Arrays are always pass by reference
  - Can use the `const` keyword to enforce input only parameter
- Array name without brackets is used as the actual parameter (argument)
- Formal parameter uses brackets to specify passing an array

22

Struble - Arrays

## Arrays As Parameters Example

```
const string INIT_STRING = "";
void main() {
    string names[MAX_NAMES];
    InitStringArray(names, MAX_NAMES, INIT_STRING);
    ...
}

// Initialize an array of strings to the specified value
void InitStringArray(string array[], int size,
                    const string& value)
{
    for (int i = 0; i < size; i++) {
        array[i] = value;
    }
}
```

23

Struble - Arrays

## Arrays As Parameters Example 2

```
void AddIntArrays(const int x[], const int y[],
                int x[], int usage);
void PrintIntArray(ostream& Out, const int x[], int usage);
void main() {
    int x[MAX_ELEM], y[MAX_ELEM];
    int i = 0;
    // read until input failure or max elements
    cin >> x[i] >> y[i];
    while (cin && i < MAX_ELEM) {
        i++;
        if (i < MAX_ELEM) {
            cin >> x[i] >> y[i];
        }
    }
    AddIntArrays(x, y, x, i);
    PrintIntArray(cout, x, i);
}
```

24

Struble - Arrays

## Arrays As Parameters Example 2

```
// Add two arrays together and store in third parameter
void AddIntArrays(const int x[], const int y[],
                 int z[], int usage)
{
    for (int i = 0; i < usage; i++) {
        z[i] = x[i] + y[i];
    }
}

// Print an integer array
void PrintIntArray(ostream& Out, const int x[], int usage) {
    for (int i = 0; i < usage; i++) {
        Out << x[i] << " ";
    }
    Out << endl;
}
}
```

25

Struble - Arrays

## Parallel Arrays

- Sometimes you want to associate different types of data to one logical item
- Example
  - Automobile parts in a store
    - Price and name
- But, arrays can only store one type of data
- Use multiple arrays
  - The index refers to the same logical item in each array

26

Struble - Arrays

## Parallel Arrays Example

```
const int MAX_ITEMS = 100;
void main() {
    string ItemName[MAX_ITEMS];
    double ItemPrice[MAX_ITEMS];
    int item = 0;
    // Make sure to initialize your arrays (code not included)
    getline(cin, ItemName[item], '\t');
    cin >> ItemPrice[item];
    while (cin && item < MAX_ITEMS) {
        item++;
        if (item < MAX_ITEMS) {
            getline(cin, ItemName[item], '\t');
            cin >> ItemPrice[item];
        }
    }
}
```

27

Struble - Arrays

## Parallel Arrays Example

Item (Index)	ItemName	ItemPrice
0	"Shocks"	59.99
1	"Brake Pads"	79.95
2	"Wipers"	2.38
3		
4		

28

Struble - Arrays

## Parallel Arrays

- To access a logical item
  - Access each parallel array with the same index
  - Index identifies the logical item of interest
- Write functions to encapsulate parallel operations
  - Initialization
  - Printing
  - Access

29

Struble - Arrays

## Multidimensional Arrays

- Arrays are often used to store data for two or more dimension
  - Matrices
  - Images
  - Graphs
  - etc.

30

Struble - Arrays

## Multidimensional Arrays Example

```
const int MAX_ROWS = 5;
const int MAX_COLS = 10;
void InitMatrix(int matrix[][MAX_COLS], int rows,
               int cols, int value);

void main() {
    int matrix[MAX_ROWS][MAX_COLS];

    InitMatrix(matrix, MAX_ROWS, MAX_COLS, 0);

    // Access fifth row, third column
    matrix[4][2] = 10;
}
```

31

Struble - Arrays

## Multidimensional Arrays Example

```
void InitMatrix(int matrix[][MAX_COLS], int rows,
               int cols, int value)
{
    for (int row = 0; row < rows; row++) {
        for (int col = 0; col < cols; col++) {
            matrix[row][col] = value;
        }
    }
}
```

32

Struble - Arrays

## Multidimensional Arrays Example

	0	1	2	...					9
0									
1									
2									
3									
4									