

## Arrays

- Structured Data Type: Collection of data whose elements are organized in such a way that individual elements can be stored and retrieved
- Array: Grouping of similarly named variables, which are grouped sequentially in memory and accessed by their element number.
- Dimension: The total number of elements for an array is specified in the declaration by its dimension.

Copyright © 2001 R.M. Laurie 1

## One Dimensional Arrays

- Declaration  

```
int nArray[5];
```

  - Reserves array memory nArray[0] to nArray[4]
- For an array dimension of n, float fArray[n]
- Elements are always numbered fArray[0] through fArray[n-1]

nArray[0]	34B3
nArray[1]	94AF
nArray[2]	043F
nArray[3]	F348
nArray[4]	34D1

Copyright © 2001 R.M. Laurie 2

## Initializing Arrays

- Initialization  

```
int nArray[5] = {0, 0, 0, 0, 0};
int nArray[ ] = {0, 0, 0, 0, 0};
```
- for(nl=0; nl<5; nl++) nArray[nl] = 0;
- Array Elements always begin with nArray[0]

nArray[0]	0
nArray[1]	0
nArray[2]	0
nArray[3]	0
nArray[4]	0

Copyright © 2001 R.M. Laurie 3

## Array Bounds

- In C++ there is NO out of bounds array checking.
- If you assign a value outside of the array it will write to an unknown memory location.
- Common Catastrophic ERROR

nArray[0]	0
nArray[1]	0
nArray[2]	0
nArray[3]	0
nArray[4]	0

Copyright © 2001 R.M. Laurie 4

## Passing Arrays Parameters 1

- Arrays Must use Pass-by-Reference
- Cannot use Pass-by-Value for Arrays
- The following example passes the X and Y coordinates to a function that draws a polygon
- Function Prototype with Array Parameters  

```
void DrawPolygon(float[ ], float[ ], int);
```
- Function Definition with Array Parameters  

```
void FuncName(float fXCoord[ ], float fYCoord[ ], int nPoints);
```

Copyright © 2001 R.M. Laurie 5

## Passing Arrays Parameters 2

- Function Call with Array Parameter

```
int main()
{
    float fXTriangle[ ] = {-10, 20.5, 0};
    float fYTriangle[ ] = {-23.3, 16, 9.75};
    ...
    DrawPolygon(fXTriangle, fYTriangle, 3);
    ...
}
```

Copyright © 2001 R.M. Laurie 6

### Passing Arrays Parameters 3

- To Prevent the Array from being modified by function, declare as constant in formal parameter list.
- Example:
  - Function Prototype with Array Parameters  
`void DrawPolygon(const float[ ],  
 const float[ ], int);`
  - Function Definition with Array Parameters  
`void FuncName(const float fXCoord[ ],  
 const float fYCoord[ ], int nPoints);`

Copyright © 2001 R.M. Laurie 7

### Passing Arrays Parameters 3

- To Prevent the Array from being modified by function, declare as constant in formal parameter list.
- Example:
  - Function Prototype with Array Parameters  
`void DrawPolygon(const float[ ],  
 const float[ ], int);`
  - Function Definition with Array Parameters  
`void FuncName(const float fXCoord[ ],  
 const float fYCoord[ ], int nPoints);`

Copyright © 2001 R.M. Laurie 8

### Passing Arrays Parameters 4

- Common Mistake: Attempt to Pass Array Dimension which passes-by-value the out of bounds element

```
DrawPolygon(fXTriangle[3], fYTriangle[3]); // Error
```

- Note: You can pass-by-value individual array elements.  
`OriginPoint(fXTriangle[0], fYTriangle[0]);`

Copyright © 2001 R.M. Laurie 9

### Strings

- An array of characters terminated with the null character '\0'
- Single quotes used to represent characters
- Double quotes used to represent null terminated string
- String Initialization:
 

szAnswer[0]	'Y'
szAnswer[1]	'e'
szAnswer[2]	's'
szAnswer[3]	'\0'
szAnswer[4]	unknown

  - char szAnswer[5] = {'Y','e','s','\0'};
  - char szAnswer[5] = "Yes";
  - char szAnswer[ ] = "Yes"; (Note: Actually has 4 elements)

Copyright © 2001 R.M. Laurie 10

### String Output

- Strings are the only type of array which can be used directly with cout
- Example  
`char szAnswer[6] = "Yes";  
 cout << szAnswer;`
- Will output characters until NULL is found

szAnswer[0]	'Y'
szAnswer[1]	'e'
szAnswer[2]	's'
szAnswer[3]	'\0'
szAnswer[4]	unknown
szAnswer[5]	unknown

Copyright © 2001 R.M. Laurie 11

### String Input

- Using cin >> has the following drawbacks:
  - No bounds checking
  - Can't input strings with white space characters
- cin.get(szAnswer, sizeof(szAnswer));
  - szAnswer is String to store characters
  - sizeof(szAnswer) max string size including Null
- cin.get(cAnswer);
  - Retrieves a single character
- cin.ignore(100, '\n')

szAnswer[0]	'Y'
szAnswer[1]	'e'
szAnswer[2]	's'
szAnswer[3]	'\0'
szAnswer[4]	'r'

Copyright © 2001 R.M. Laurie 12

### cstring Library Functions

- `int strlen(szString);`
  - String Length excluding NULL
- `strcmp(szString1, szString2);`
  - Compares two strings
- `strcpy(szString1, szString2);`
  - Copies szString2 to szString1
- `strcat(szString1, szString2);`
  - Appends szString2 to szString1

Copyright © 2001 R.M. Laurie 13

### MultiDimensional Arrays

- Two dimensional Arrays
  - `float fArray[4][2];`
  - rows, columns
  - Like a Table
- Three dimensional Arrays
  - `int nArray[6][4][2]`
  - rows, columns, depth
  - Like a 3D brick
- There is no limit to the dimensions of an Array in C++

Copyright © 2001 R.M. Laurie 14

**EXAMPLE** -- To keep monthly high temperatures for all 50 states in one array.

```

const int NUM_STATES = 50;
const int NUM_MONTHS = 12;
int stateHighs [ NUM_STATES ] [ NUM_MONTHS ];
    
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
[0]												
[1]												
[2]	66	64	72	78	85	90	99	105	98	90	88	80
[3]												
[4]												
[48]												
[49]												

row 2, col 7 might be Arizona's high for August

stateHighs [2] [7]

Copyright © 2001 R.M. Laurie 15