Arrays

Lecture 7

4.1 Introduction

- Arrays
  - Structures of related data items
  - Static entity
    - (same number of items throughout program)

4.2 Arrays

- Array
  - Consecutive group of memory locations
  - Same name and type (**int**, **char**, etc.)
- To refer to an element
  - Specify array name and position number (index)
  - Format: arrayname[position number]
  - First element at position 0
- N-element array c

  c[0], **c[1]** … **c[n – 1]**
  - Nth element as position N-1

4.2 Arrays

- Array elements like other variables
  - Assignment, printing for an integer array **c**

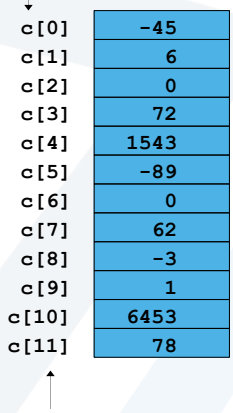
```
c[ 0 ] =  3;
cout << c[ 0 ];
```

- Can perform operations inside subscript

  c[5 – 2] same as **c[3]**

4.2 Arrays

Name of array (Note that all elements of this array have the same name, **c**)



Position number of the element within array **c**

ht

5

4.3 Declaring Arrays

- When declaring arrays, specify
  - Name
  - Type of array
    - Any data type
  - Number of elements
  - *type arrayName* **[** *arraySize* **] ;**

```
int c[ 10 ];  // array of 10 integers
float d[ 3284 ]; // array of 3284 floats
```

- Declaring multiple arrays of same type
  - Use comma separated list, like regular variables

```
int b[ 100 ], x[ 27 ];
```

4.4 Examples Using Arrays

- Initializing arrays
  - For loop
    - Set each element
  - Initializer list
    - Specify each element when array declared

```
int n[ 5 ] = { 1, 2, 3, 4, 5 };
```

    - If not enough initializers, rightmost elements 0
    - If too many syntax error
  - To set every element to same value

```
int n[ 5 ] = { 0 };
```

  - If array size omitted, initializers determine size

```
int n[] = { 1, 2, 3, 4, 5 };
```

    - 5 initializers, therefore 5 element array

fig04_03.cpp (1 of 2)

```
1  // Fig. 4.3: fig04_03.cpp
2  // Initializing an array.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14   int n[ 10 ];  // n is an array of 10 integers
15
16   // initialize elements of array n to 0
17   for ( int i = 0; i < 10; i++ )
18     n[ i ] = 0;  // set element at location i to 0
19
20   cout << "Element" << setw( 13 ) << "Value" << endl;
21
22   // output contents of array n in tabular format
23   for ( int j = 0; j < 10; j++ )
24     cout << setw( 7 ) << j << setw( 13 ) << n[ j ] << endl;
25
```

```
26    return 0;  // indicates successful termination
27
28 } // end main
```

(2 of 2)

fig04_03.cpp
output (1 of 1)

```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
      5             0
      6             0
      7             0
      8             0
      9             0
```


fig04_04.cpp
(1 of 1)

```
1  // Fig. 4.4: fig04_04.cpp
2  // Initializing an array with a declaration.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14   // use initializer list to initialize array n
15   int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
16
17   cout << "Element" << setw( 13 ) << "Value" << endl;
18
19   // output contents of array n in tabular format
20   for ( int i = 0; i < 10; i++ )
21     cout << setw( 7 ) << i << setw( 13 ) << n[ i ] << endl;
22
23   return 0;  // indicates successful termination
24
25 } // end main
```

fig04_04.cpp output (1 of 1)

```
Element         Value
      0            32
      1            27
      2            64
      3            18
      4            95
      5            14
      6            90
      7            70
      8            60
      9            37
```


4.4 Examples Using Arrays

- Array size
  - Can be specified with constant variable (**const**)
    - **const int SIZE = 20;**
  - Constants cannot be changed
  - Constants must be initialized when declared
  - Also called named constants or read-only variables

fig04_05.cpp (1 of 2)

```
1  // Fig. 4.5: fig04_05.cpp
2  // Initialize array s to the even integers from 2 to 20.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14   // constant variable can be used to specify array size
15   const int ARRAYSIZE = 10;
16
17   int s[ ARRAYSIZE ];  // array s has 10 elements
18
19   for ( int i = 0; i < ARRAYSIZE; i++ )  // set the values
20     s[ i ] = 2 + 2 * i;
21
22   cout << "Element" << setw( 13 ) << "Value" << endl;
23
```


(2 of 2)

```
24   // output contents of array s in tabular format
25   for ( int j = 0; j < ARRAYSIZE; j++ )
26     cout << setw( 7 ) << j << setw( 13 ) << s[ j ] << endl;
27
28   return 0;  // indicates successful termination
29
30 } // end main
```

fig04_05.cpp output (1 of 1)

```
Element         Value
      0             2
      1             4
      2             6
      3             8
      4            10
      5            12
      6            14
      7            16
      8            18
      9            20
```

(1 of 1)

fig04_07.cpp
output (1 of 1)

```
1  // Fig. 4.7: fig04_07.cpp
2  // A const object must be initialized.
3
4  int main()
5  {
6     const int X;  // Error: x must be initialized
7
8     X = 7;       // Error: cannot modify a const variable
9
10    return 0;    // indicates successful termination
11
12 } // end main
```

```
d:\cpphtp4_examples\ch04\Fig04_07.cpp(6) : error C2734: 'x' :
      const object must be initialized if not extern
d:\cpphtp4_examples\ch04\Fig04_07.cpp(8) : error C2166:
   l-value specifies const object
```


(1 of 1)

fig04_08.cpp
output (1 of 1)

```
1  // Fig. 4.8: fig04_08.cpp
2  // Compute the sum of the elements of the array.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main()
9  {
10    const int ARRAYSIZE = 10;
11
12    int a[ ARRAYSIZE ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
13
14    int total = 0;
15
16    // sum contents of array a
17    for ( int i = 0; i < ARRAYSIZE; i++ )
18       total += a[ i ];
19
20    cout << "Total of array element values is " << total << endl;
21
22    return 0;  // indicates successful termination
23
24 } // end main
```

```
Total of array element values is 55
```

fig04_09.cpp (1 of 2)

```
1  // Fig. 4.9: fig04_09.cpp
2  // Histogram printing program.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14   const int ARRAYSIZE = 10;
15   int n[ ARRAYSIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
16
17   cout << "Element" << setw( 13 ) << "Value"
18        << setw( 17 ) << "Histogram" << endl;
19
20   // for each element of array n, output a bar in histogram
21   for ( int i = 0; i < ARRAYSIZE; i++ ) {
22     cout << setw( 7 ) << i << setw( 13 )
23          << n[ i ] << setw( 9 );
24
25     for ( int j = 0; j < n[ i ]; j++ )   // print one bar
26       cout << '*';
```


fig04_09.cpp (2 of 2)

```
27
28     cout << endl;  // start next line of output
29
30   } // end outer for structure
31
32   return 0;  // indicates successful termination
33
34 } // end main
```

fig04_09.cpp output (1 of 1)

```
Element         Value         Histogram
      0            19         ********************
      1             3         ***
      2            15         ***************
      3             7         *******
      4            11         ***********
      5             9         *********
      6            13         *************
      7             5         *****
      8            17         *****************
      9             1         *
```

fig04_10.cpp (1 of 2)

```
1  // Fig. 4.10: fig04_10.cpp
2  // Roll a six-sided die 6000 times.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 #include <cstdlib>
13 #include <ctime>
14
15 int main()
16 {
17    const int ARRAYSIZE= 7;
18    int frequency[ ARRAYSIZE ] = { 0 };
19
20    srand( time( 0 ) );  // seed random-number generator
21
22    // roll die 6000 times
23    for ( int roll = 1; roll <= 6000; roll++ )
24       ++frequency[ 1 + rand() % 6 ]; // replaces 20-line switch
25                                      // of Fig. 3.8
```


(2 of 2)

fig04_10.cpp output (1 of 1)

```
26
27    cout << "Face" << setw( 13 ) << "Frequency" << endl;
28
29    // output frequency elements 1-6 in tabular format
30    for ( int face = 1; face < ARRAYSIZE; face++ )
31       cout << setw( 4 ) << face
32            << setw( 13 ) << frequency[ face ] << endl;
33
34    return 0;  // indicates successful termination
35
36 } // end main
```

```
Face     Frequency
   1          1003
   2          1004
   3           999
   4           980
   5          1013
   6          1001
```

fig04_11.cpp (1 of 2)

```
1  // Fig. 4.11: fig04_11.cpp
2  // Student poll program.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 int main()
13 {
14   // define array sizes
15   const int RESPONSESIZE = 40;   // size of array responses
16   const int FREQUENCYSIZE = 11;  // size of array frequency
17
18   // place survey responses in array responses
19   int responses[ RESPONSESIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8,
20     10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7,
21     5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
22
23   // initialize frequency counters to 0
24   int frequency[ FREQUENCYSIZE ] = { 0 };
25
```


fig04_11.cpp (2 of 2)

```
26   // for each answer, select value of an element of array
27   // responses and use that value as subscript in array
28   // frequency to determine element to increment
29   for ( int answer = 0; answer < RESPONSESIZE; answer++ )
30     ++frequency[ responses[answer] ];
31
32   // display results
33   cout << "Rating" << setw( 17 ) << "Frequency" << endl;
34
35   // output frequencies in tabular format
36   for ( int rating = 1; rating < FREQUENCYSIZE; rating++ )
37     cout << setw( 6 ) << rating
38          << setw( 17 ) << frequency[ rating ] << endl;
39
40   return 0;  // indicates successful termination
41
42 } // end main
```

fig04_11.cpp output (1 of 1)

```
Rating        Frequency
     1                2
     2                2
     3                2
     4                2
     5                5
     6               11
     7                5
     8                7
     9                1
    10                3
```


4.4 Examples Using Arrays

- Strings (more in ch. 5)
  - Arrays of characters
  - All strings end with **`null`** (**`'\0'`**)
  - Examples
    - **`char string1[] = "hello";`**
      - **`Null`** character implicitly added
      - **`string1`** has 6 elements
    - **`char string1[] = { 'h', 'e', 'l', 'l', 'o', '\0' };`**
  - Subscripting is the same
    `String1[ 0 ]` is **`'h'`**
    `string1[ 2 ]` is **`'l'`**

4.4 Examples Using Arrays

- Input from keyboard

```
char string2[ 10 ];
cin >> string2;
```

  - Puts user input in string
    - Stops at first whitespace character
    - Adds **`null`** character
  - If too much text entered, data written beyond array
    - We want to avoid this (section 5.12 explains how)
- Printing strings
  - **`cout << string2 << endl;`**
    - Does not work for other array types
  - Characters printed until **`null`** found

fig04_12.cpp (1 of 2)

```
1  // Fig. 4_12: fig04_12.cpp
2  // Treating character arrays as strings.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11   char string1[ 20 ],          // reserves 20 characters
12   char string2[] = "string literal"; // reserves 15 characters
13
14   // read string from user into array string2
15   cout << "Enter the string \"hello there\": ";
16   cin >> string1;  // reads "hello" [space terminates input]
17
18   // output strings
19   cout << "string1 is: " << string1
20      << "\nstring2 is: " << string2;
21
22   cout << "\nstring1 with spaces between characters is:\n";
23
```

```
24    // output characters until null character is reached
25    for ( int i = 0; string1[ i ] != '\0'; i++ )
26       cout << string1[ i ] << ' ';
27
28    cin >> string1;  // reads "there"
29    cout << "\nstring1 is: " << string1 << endl;
30
31    return 0;  // indicates successful termination
32
33 } // end main
```

(2 of 2)

fig04_12.cpp
output (1 of 1)

```
Enter the string "hello there": hello there
string1 is: hello
string2 is: string literal
string1 with spaces between characters is:
h e l l o
string1 is: there
```


4.5 Passing Arrays to Functions

- Specify name without brackets
  - To pass array **myArray** to **myFunction**

```
int myArray[ 24 ];
myFunction( myArray, 24 );
```

  - Array size usually passed, but not required
    - Useful to iterate over all elements

4.5 Passing Arrays to Functions

- Functions can modify original array data
- Value of name of array is address of first element
  - Function knows where the array is stored
  - Can change original memory locations
- Individual array elements passed like regular variables
  - Cannot modify calling function's value
  - `square( myArray[3] );`

4.5 Passing Arrays to Functions

- Functions taking arrays
  - Function prototype
    - `void modifyArray( int b[], int arraySize );`
    - `void modifyArray( int [], int );`
      - Names optional in prototype
    - Both take an integer array and a single integer
  - No need for array size between brackets
    - Ignored by compiler

fig04_14.cpp (1 of 3)

```
1  // Fig. 4.14: fig04_14.cpp
2  // Passing arrays and individual array elements to functions.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setw;
11
12 void modifyArray( int [], int );  // appears strange
13 void modifyElement( int );
14
15 int main()
16 {
17    const int ARRAYSIZE = 5;              // size of array a
18    int a[ ARRAYSIZE ] = { 0, 1, 2, 3, 4 };  // initialize a
19
20    cout << "Effects of passing entire array by reference:"
21       << "\n\nThe values of the original array are:\n";
22
23    // output original array
24    for ( int i = 0; i < ARRAYSIZE; i++ )
25       cout << setw( 3 ) << a[ i ];
```


fig04_14.cpp (2 of 3)

```
26
27    cout << endl;
28
29    // pass array a to modifyArray by reference
30    modifyArray( a, ARRAYSIZE );
31
32    cout << "The values of the modified array are:\n";
33
34    // output modified array
35    for ( int j = 0; j < ARRAYSIZE; j++ )
36       cout << setw( 3 ) << a[ j ];
37
38    // output value of a[ 3 ]
39    cout << "\n\n\n"
40       << "Effects of passing array element by value:"
41       << "\n\nThe value of a[3] is " << a[ 3 ] << '\n';
42
43    // pass array element a[ 3 ] by value
44    modifyElement( a[ 3 ] );
45
46    // output value of a[ 3 ]
47    cout << "The value of a[3] is " << a[ 3 ] << endl;
48
49    return 0;  // indicates successful termination
50
51 } // end main
```

fig04_14.cpp (3 of 3)

```
52
53 // in function modifyArray, "b" points to
54 // the original array "a" in memory
55 void modifyArray( int b[], int sizeOfArray )
56 {
57    // multiply each array element by 2
58    for ( int k = 0; k < sizeOfArray; k++ )
59       b[ k ] *= 2;
60
61 } // end function modifyArray
62
63 // in function modifyElement, "e" is a local copy of
64 // array element a[ 3 ] passed from main
65 void modifyElement( int e )
66 {
67    // multiply parameter by 2
68    cout << "Value in modifyElement is "
69       << ( e *= 2 ) << endl;
70
71 } // end function modifyElement
```


fig04_14.cpp output (1 of 1)

```
Effects of passing entire array by reference:

The values of the original array are:
 0 1 2 3 4
The values of the modified array are:
 0 2 4 6 8


Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
```