

## Chapter 8 - Arrays

8.1 Array Basics	8.4 Parallel Arrays
8.2 Sequentially Searching an Array	8.5 Two-Dimensional Arrays
8.3 Processing the Contents of an Array	8.6 Arrays of Three or More Dimensions

*The big idea behind this chapter is ....*

---

*It relates to the previous chapter how ...*

---

*The main purpose of this chapter is ...*

---

*The key questions are ...*

---

Why:

When:

How:

*Why is this material at this point in the class?*

---

*You'll know this material when ...*

---

*Main assumptions are ...*

---

*Opening Thoughts.* Write any thoughts or questions you have before reading this material. See if you can find the answers while you read.

*Key Ideas.* Record major points from the chapter.

[illegible]

## *List of Figures*

1	Array use case.	4
2	Array with loop demo.	5
3	Demo out of bounds error.	5
4	Range based loop example.	6
5	Demo of a variable based array.	6
6	How to process array elements.	6
7	Process string as an array demo.	7
8	Parallel array demo.	7
9	Example of passing an array.	8
10	Pass by reference.	8
11	TypeDef and arrays.	9
12	Multidimensional array demo.	9
13	How to pass a MD array.	10
14	Vector demo.	11
15	Vector size() function.	12
16	Push() and Pop() demo.	13
17	Vector clear() demo..	13
18	Parallel vectors.	14
19	Vector loop with reference variable.	15

```
1 // 8-1.cpp -- Intro to arrays
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     const int NUM_EMPLOYEES = 6;
7     int hours[NUM_EMPLOYEES];
8
9     cout << "Enter hours worked by "
10         << NUM_EMPLOYEES << " employees: ";
11     cin >> hours[0];
12     cin >> hours[1];
13     cin >> hours[2];
14     cin >> hours[3];
15     cin >> hours[4];
16     cin >> hours[5];
17
18     cout << "The hours you entered are: ";
19     cout << " " << hours[0];
20     cout << " " << hours[1];
21     cout << " " << hours[2];
22     cout << " " << hours[3];
23     cout << " " << hours[4];
24     cout << " " << hours[5] << endl;
25
26     return 0;
27 }
```

Figure 1: §8.3 Array use case. Source file: 8-1.cpp

```

1 // 8-2.cpp -- Array with loops
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     const int NUM_EMPLOYEES = 6;
7     int hours[NUM_EMPLOYEES];
8
9     cout << "Enter the hours worked by "
10         << NUM_EMPLOYEES << " employees: ";
11
12     for (int i = 0; i < NUM_EMPLOYEES; i++)
13         cin >> hours[i];
14
15     cout << "The hours you entered are: ";
16
17     for (int i = 0; i < NUM_EMPLOYEES; i++)
18         cout << " " << hours[i];
19
20     cout << endl;
21     return 0;
22 }

```

Figure 2: §8.3 Array with loop demo.  
Source file: 8-2.cpp

```

1 // 8-4.cpp -- Demo of out of bounds errors
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     const int SIZE = 3;
7     int A[SIZE] = { 1, 1, 1 };
8     int B[SIZE];
9
10    cout << "Original data in array A\n";
11    for (int count = 0; count < 3; count++)
12        cout << A[count] << "\t";
13
14    cout << "\n\n Now store 7 elements in the second 3 element array\n";
15    for (int count = 0; count < 7; count++)
16        B[count] = 5;
17
18    cout << "If the program did not crash, try to display the array\n";
19    for (int count = 0; count < 7; count++)
20        cout << B[count] << "\t";
21
22    cout << "\n\n Display array A.\n";
23    for (int count = 0; count < 3; count++)
24        cout << A[count] << "\t";
25
26    cout << "\n\n Array A's values were overwritten by \n"
27        << "the values overflowing array B\n";
28
29    return 0;
30 }

```

Figure 3: §8.3 Demo out of bounds error. Source file: 8-4.cpp

```

1 //8-9.cpp -- range based for loop demo
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main() {
7     string planets[] = { "Mercury", "Venus", "Earth", "Mars",
8                           "Jupiter", "Saturn", "Uranus",
9                           "Neptune", "Pluto"};
10
11     cout << "Here are the planets\n";
12     for (string planet : planets)
13         cout << planet << endl;
14
15     return 0;
16 }
17

```

Figure 4: §8.5 Range based loop example. Source file: 8-9.cpp

```

1 #include <iostream>
2
3 int main() {
4     int num;
5     std::cout << "Enter size: ";
6     std::cin >> num;
7     int quizzes[num];
8     std::cout << "Hey! I compile!" << std::endl;
9 }

```

Figure 5: §8.x1 Demo of a variable based array. Source file: noName.cpp

```

1 // arrStuff.cpp -- stuff to do with an array
2 #include <iostream>
3 using namespace std;
4
5 int main() {
6     const int SIZE = 5;
7     int aArray[] = {10, 20, 30, 40, 50};
8     int bArray[] = {0, 0, 0, 0, 0};
9
10     for (int e : aArray) cout << e << ", ";
11     cout << endl;
12     for (int e : bArray) cout << e << ", ";
13     cout << endl;
14
15     cout << "Array copy\n";
16     // Can't do bArray = aArray
17     for (int i = 0; i < SIZE; i++)
18         bArray[i] = aArray[i];
19

```

Figure 6: §8.6 How to process array elements. Source file: arrStuff.cpp

```

1 // 8-14(mod).cpp -- process a string as an array
2 #include <iostream>
3 #include <string>
4 #include <cctype>
5 using namespace std;
6
7 int main() {
8     char ch;
9     int vowelCount = 0;
10    string sentence;
11
12    cout << "Enter a sentence" << endl;
13    getline(cin, sentence);
14
15    for (int pos = 0; pos < sentence.length(); pos++) {
16        ch = toupper(sentence[pos]);
17
18        switch(ch) {
19            case 'A':
20            case 'E':
21            case 'I':
22            case 'O':
23            case 'U': vowelCount++;
24        }
25    }
26
27    cout << "There are " << vowelCount << " vowels in this sentence.\n";
28
29    return 0;
30 }

```

Figure 7: §8.6 Process string as an array demo. Source file: 8-14(mod).cpp

```

1 // paraMo.cpp -- C++ parallel arrays
2 #include <iostream>
3
4 using namespace std;
5
6 int main() {
7     string names[] = {"BAD", "January", "February", "March", "April", "May", "June",
8                     "July", "August", "September", "October", "November",
9                     "December"};
10    int days[] = {0, 30, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
11
12    int mo = 0;
13    cout << "Enter a month number (1-12): ";
14    cin >> mo;
15
16    cout << names[mo] << " has " << days[mo] << " days in it\n";
17    // cout << names[mo+1] << " has " << days[mo+1] << " days in it\n";
18
19    return 0;
20 }

```

Figure 8: §8.7 Parallel array demo. Source file: paraMo.cpp

```

1 // 8-16.cpp -- pass array elements
2 #include <iostream>
3 using namespace std;
4
5 void ShowValue(int);
6
7 int main() {
8     int collection[]{5, 10, 15, 20, 25, 30, 35, 40};
9
10    for (int c : collection)
11        ShowValue(c);
12    cout << endl;
13
14    return 0;
15 }
16
17 void ShowValue(int num) {
18     cout << num << " ";
19 }

```

Figure 9: §8.9 Example of passing an array. Source file: 8-16.cpp

```

1 // 8-17.cpp -- pass entire array
2 #include <iostream>
3 using namespace std;
4
5 void ShowValue(int intArray[], int size);
6
7 int main() {
8     const int SIZE = 8;
9     int collection[]{5, 10, 15, 20, 25, 30, 35, 40};
10
11    ShowValue(collection, SIZE);
12    cout << endl;
13
14    return 0;
15 }
16
17 void ShowValue(int nums[], int size) {
18     for (int i = 0; i < size; i++)
19         cout << nums[i] << " ";
20     cout << endl;
21 }

```

Figure 10: §8.9 Pass by reference. Source file: 8-17.cpp



```

1 // 8-18.cpp -- array and typedef
2 #include <iostream>
3 using namespace std;
4
5 typedef int arrayType[];
6
7 void ShowValue(arrayType, int);
8
9 int main() {
10     const int SIZE = 8;
11     int collection[] {5, 10, 15, 20, 25, 30, 35, 40};
12
13     ShowValue(collection, SIZE);
14
15     return 0;
16 }
17
18 void ShowValue(arrayType nums, int size) {
19     for (int i = 0; i < size; i++)
20         cout << nums[i] << " ";
21     cout << endl;
22 }

```

Figure 11: §8.9 TypeDef and arrays.  
Source file: 8-18.cpp

```

1 // mdArray.cpp -- MD array demo
2 #include <iostream>
3 #include <iomanip>
4 using namespace std;
5
6 const int NUM_COLS = 4;
7 const int TBL1_ROWS = 3;
8
9
10 int main() {
11     int table1[TBL1_ROWS][NUM_COLS] = { {1, 2, 3, 4},
12                                           {5, 6, 7, 8},
13                                           {9, 10, 11, 12} };
14     cout << "Table 1's contents:\n";
15     for (int row = 0; row < TBL1_ROWS; row++) {
16         for (int col = 0; col < NUM_COLS; col++) {
17             cout << setw(5) << table1[row][col] << " ";
18         }
19         cout << endl;
20     }
21 }
22

```

Figure 12: §8.10 Multidimensional array  
demo. Source file: mdArray.cpp

```

1 // 8-22.cpp -- passing an MD array demo
2 #include <iostream>
3 #include <iomanip>
4 using namespace std;
5
6 const int NUM_COLS = 4;
7 const int TBL1_ROWS = 3;
8 const int TBL2_ROWS = 4;
9
10 void ShowArray(const int[][NUM_COLS], int);
11
12 int main() {
13     int table1[TBL1_ROWS][NUM_COLS] = { {1, 2, 3, 4},
14                                           {5, 6, 7, 8},
15                                           {9, 10, 11, 12} };
16     int table2[TBL2_ROWS][NUM_COLS] = { {10, 20, 30, 40},
17                                           {50, 60, 70, 80},
18                                           {90, 100, 110, 120} };
19     cout << "Table 1's contents:\n";
20     ShowArray(table1, TBL1_ROWS);
21     cout << "Table 2's contents:\n";
22     ShowArray(table2, TBL2_ROWS);
23 }
24
25 void ShowArray(int const array[][NUM_COLS], int numRows) {
26     for (int row = 0; row < numRows; row++) {
27         for (int col = 0; col < NUM_COLS; col++) {
28             cout << setw(5) << array[row][col] << " ";
29         }
30         cout << endl;
31     }
32 }
33

```

Figure 13: §8.10 How to pass a MD array. Source file: 8-22.cpp

```

1 // 8-26(mod).cpp -- Demo adding elements to vector
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5
6 int main() {
7     vector<int> numbers;
8     int num = 0;
9     int tempNum = 0;
10
11     cout << "How many numbers to enter: ";
12     cin >> num;
13
14     for (int i = 0; i < num; i++) {
15         cout << "Enter number " << (i + 1) << ": ";
16         cin >> tempNum;
17         numbers.push_back(tempNum);
18     }
19
20     cout << "Here's the values: \n";
21
22     for (int val : numbers)
23         cout << val << " ";
24     cout << endl;
25     return 0;
26 }

```

Figure 14: §8.12 Vector demo. Source file: 8-26(mod).cpp

```

1  // 8-27.cpp -- Demo vector size function
2  #include <iostream>
3  #include <vector>
4  using namespace std;
5
6  void showValues(vector<int>);
7
8  int main() {
9      vector<int> values;
10
11      for (int i = 0; i < 4; i++)
12          values.push_back(i * 2);
13
14      showValues(values);
15
16      return 0;
17  }
18
19  void showValues(vector<int> vect) {
20      for (int i = 0; i < vect.size(); i++)
21          cout << vect[i] << " ";
22      cout << endl;
23  }

```

Figure 15: §8.12 Vector size() function.  
Source file: 8-27.cpp

```

1 // 8-28(mod).cpp -- Vector demo size, push_back, pop_back
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5
6 int main() {
7     vector<int> values;
8     values.push_back(1);
9     values.push_back(2);
10    values.push_back(3);
11    cout << "The size of values is " << values.size() << endl;
12
13    cout << "Popping a value\n";
14    values.pop_back();
15    for (int i = 0; i < values.size(); i++)
16        cout << values[i] << " ";
17    cout << endl;
18
19    cout << "Popping another value\n";
20    values.pop_back();
21    for (int i : values)
22        cout << i << " ";
23    cout << endl;
24
25    cout << "Popping the last value\n";
26    values.pop_back();
27    if (values.empty())
28        cout << "Vector is empty";
29    else
30        for (int i : values)
31            cout << i << " ";
32    cout << endl;
33
34    return 0;
35 }

```

Figure 16: §8.12 Push() and Pop() demo.  
Source file: 8-28(mod).cpp

```

1 // 8-29.cpp -- Demo of vector clear function
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5
6 int main() {
7     vector<int> values(100);
8
9     cout << "The values vector has "
10         << values.size() << " elements.\n";
11    cout << "I now call the clear method.\n";
12    values.clear();
13    cout << "The values vector has "
14         << values.size() << " elements.\n";
15
16    return 0;
17 }

```

Figure 17: §8.12 Vector clear() demo..  
Source file: 8-29.cpp

```

1 // 8-24.cpp -- Parallel vectors
2 #include <iostream>
3 #include <iomanip>
4 #include <vector>
5 using namespace std;
6
7 int main() {
8     const int NUM_EMPS = 5;
9     vector<int> hours(NUM_EMPS);
10    vector<double> payRate(NUM_EMPS);
11    double grossPay;
12
13    cout << "Enter the hours worked and the hourly pay"
14         << " rates of " << NUM_EMPS << " employees.\n";
15    for (int i = 0; i < NUM_EMPS; i++) {
16        cout << "\nHours worked by employee #"
17             << (i + 1) << ": ";
18        cin >> hours[i];
19        cout << "Hourly pay rate for this employee: $";
20        cin >> payRate[i];
21    }
22
23    cout << "\nGross pay per employee:\n";
24    cout << fixed << showpoint << setprecision(2);
25    for (int i = 0; i < NUM_EMPS; i++) {
26        grossPay = hours[i] * payRate[i];
27        cout << "Employee #" << (i + 1);
28        cout << ": $" << setw(7) << grossPay << endl;
29    }
30    return 0;
31 }

```

Figure 18: §8.12 Parallel vectors. Source file: 8-24.cpp

```

1 // 8-25.cpp -- vector loop & ref variable
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5
6 int main() {
7     vector<int> numbers(3);
8
9     for (int& val: numbers) {
10         cout << "Enter an integer: ";
11         cin >> val;
12     }
13
14     cout << "Here's the values: \n";
15
16     for (int val : numbers)
17         cout << val << " ";
18     cout << endl;
19     return 0;
20 }
21

```

Figure 19: §8.12 Vector loop with reference variable. Source file: 8-25.cpp