**LaGuardia Community College – Last Update**

08

**Fall**

Part 8: Arrays

Dr. Andi Toce

Lecture Notes for MAC 101 (Introduction to Computer Science)

Last updated / viewed: April 23, 15

Table of Contents

1. What are Arrays? 2

2. Initializing Arrays 3

3. Array Size 5

4. Passing Arrays to Functions 6

5. Characters, Strings and Arrays of Strings 7

6. Multi-Dimensional Arrays 10

# What are Arrays?

An array is an ordered collection (potentially very large) of items. Arrays are indexed for an easier access to its elements. Arrays are powerful structures that allow the user to store multiple data values without having to declare as many variables.

Array declaration: ***data\_type array\_identifier[number\_of\_items]***

Assume we want to store 5 double data values for further processing (example find their average). One way is to declare 5 different integer variables, one for each value. This is not an efficient way of storing data. Imagine we have to do this for a lot more data values. The following program uses an array to store the data values and calculate the average of all these values.

|  |  |
| --- | --- |
| FirstArray.cpp | Output |
| #include <iostream>  using namespace std;  int main() {  double myArray[5]; // Declare an array of 5 values  cout << "Enter value: " << endl;  //Enter all values  for (int i = 0; i < 5; i++) {  cin >> myArray[i];  }  //Calculate Average  double sum;  for (int i = 0; i < 5; i++) {  sum += myArray[i];  }  cout << "The Average is: " << sum / 5 << endl;  return 0;  } | Enter value:  5  6  8  2.3  3.2  The Average is: 4.9 |

# Initializing Arrays

Arrays follow the same principles regarding initialization as variables. Global numerical variables (arrays) are all initialized to zero by default if the user does not specify any values. Local variables must be initialized by the user otherwise they will be possibly assigned random values.

|  |  |
| --- | --- |
| InitializingArrays.cpp | Output |
| #include <iostream>  using namespace std;  int GlobalArray[6];  int main() {  int LocalArray[8];  cout << "GlobalArray values before initializing." << endl;  for (int i=0; i<6; i++)  cout << GlobalArray[i] << " ";  cout << endl << endl;  cout << "LocalArray values before initializing." << endl;  for (int i=0; i<8; i++)  cout << LocalArray[i] << " ";  cout << endl <<endl;  int InitializedLocalArray[5] = {3,4,6,2,4};  cout << "InitializedLocalArray values." << endl;  for (int i=0; i<5; i++)  cout << InitializedLocalArray[i] << " ";  return 0;  } | GlobalArray values before initializing.  0 0 0 0 0 0  LocalArray values before initializing.  4751328 0 0 6 2293280 0 4255955 0  InitializedLocalArray values.  3 4 6 2 4 |

**Try now:** Modify the following code to use an array instead of the individually declared variables. Does this approach improve code writing and code readability?

|  |
| --- |
| RandomDiceSimulator.cpp |
| #include <iostream>  #include <cstdlib>  #include <ctime>  using namespace std;  void keepRecord(int);  void displayStatistics();  int n1, n2, n3, n4, n5, n6;  int main() {  int n, i, r;  srand(time(NULL)); // Set seed for random numbers.  cout << "Enter number of dice to roll: ";  cin >> n;  for (i = 1; i <= n; i++) {  r = rand() % 6 + 1; // Get a number 1 to 6  cout << r << " ";  keepRecord(r);  }  cout << endl << endl;  displayStatistics();  return 0;  }  void keepRecord(int value){  switch (value){  case 1 : n1++; break;  case 2 : n2++; break;  case 3 : n3++; break;  case 4 : n4++; break;  case 5 : n5++; break;  case 6 : n6++; break;  }  }  void displayStatistics(){  cout << "Value \t" << "Freq \t" << "Graph" << endl << endl;  cout << "1 \t" << n1 << "\t" ;  for(int i=0; i<n1;i++) cout << "\*";  cout << endl;  cout << "2 \t" << n2 << "\t" ;  for(int i=0; i<n2;i++) cout << "\*";  cout << endl;  cout << "3 \t" << n3 << "\t" ;  for(int i=0; i<n3;i++) cout << "\*";  cout << endl;  cout << "4 \t" << n4 << "\t" ;  for(int i=0; i<n4;i++) cout << "\*";  cout << endl;  cout << "5 \t" << n5 << "\t" ;  for(int i=0; i<n1;i++) cout << "\*";  cout << endl;  cout << "6 \t" << n6 << "\t" ;  for(int i=0; i<n6;i++) cout << "\*";  cout << endl;  } |

# Array Size

**Keeping track of array size.** It is a good practice to declare a constant array size integer to be used when needed.

|  |  |
| --- | --- |
| ArraySize.cpp | Output |
| #include <iostream>  using namespace std;  int main() {  const int myArraySize = 5;  int myArray[myArraySize] = {3,4,6,2,4};  cout << "The Size of myArray is: " << myArraySize << endl;  cout << "Elements in the arrays are: ";  for (int i=0; i<myArraySize; i++)  cout << myArray[i] << " ";  return 0;  } | The Size of myArray is: 5  Elements in the arrays are: 3 4 6 2 4 |

Another way of getting array size.

|  |
| --- |
| ArraySize2.cpp |
| #include <iostream>  using namespace std;  int main() {  int myArray[5] = {3,4,6,2,4};  cout << "The Size in bytes of MyArray is: " << sizeof(myArray) << endl;  cout << "The Size in bytes of the first element is: " << sizeof(myArray[0]) << endl;  int size = sizeof(myArray)/sizeof(myArray[0]);  cout << "The Total number of elements in MyArray is: " << size << endl;  cout << "Elements in the arrays are: ";  for (int i=0; i<size; i++)  cout << myArray[i] << " ";  return 0;  } |
| Output |
| The Size in bytes of MyArray is: 20  The Size in bytes of the first element is: 4  The Total number of elements in MyArray is: 5  Elements in the arrays are: 3 4 6 2 4 |

# Passing Arrays to Functions

In C++ arrays are passed to functions by reference. Individual elements from the array are passed by value. See example below.

|  |  |
| --- | --- |
| PassingArraysToFunctions.cpp | Output |
| #include <iostream>  #include <iomanip>  using namespace std;  void modifyArray(int [], int);  int main() {  const int arraySize = 5;  int myArray[arraySize] = {3,4,6,2,8};  cout << "Elements in the original array are: " << endl;  for (int i=0; i<arraySize; i++)  cout << myArray[i] << " ";  cout << endl;  modifyArray(myArray, arraySize); // modify the array  cout << "Elements in the modified array are: " << endl;  for (int i=0; i<arraySize; i++)  cout << myArray[i] << " ";  cout << endl;  return 0;  } // end main  void modifyArray(int a[], int sizeOfArray){  // add one to each array element  for (int i=0; i<sizeOfArray; i++)  a[i]++;  } | Elements in the original array are:  3 4 6 2 8  Elements in the modified array are:  4 5 7 3 9 |

**Try now:** Add another value to the number set {3,4,6,2,8}, in *PassingArraysToFunctions.cpp,* compile and run the program. Does the program run? What do you observe? What can you conclude?

**Try now:** Change the function signature to read **void** **modifyArray**(const **int** a[], **int** sizeOfArray). Compile and run the program. What do you notice?

**Try now:** Add a cout statement that prints myArray[5]. Does it work? What do you think is happening?

# Characters, Strings and Arrays of Strings

Example illustrating different char arrays.

|  |  |
| --- | --- |
| CharArrayTest.cpp | Output |
| #include <iostream>  using namespace std;  int main(){  char firstArray[8];  for(int i=0; i<8;i++){  firstArray[i]='d';  }  cout << "Print firstArray using for loop: " << endl;  for(int i=0; i<7;i++){  cout << firstArray[i];  }  firstArray[7]='\0';  cout << endl;  cout << "Print firstArray using identifier: " << endl;  cout << firstArray << endl;  char secondArray[] = { 'H', 'e', 'l', 'l', 'o', '\0'};  cout << "Print secondArray using for loop: " << endl;  for(int i=0; i<6;i++){  cout << secondArray[i];  }  cout << endl;  char thirdArray[] = "Test";  cout << "Print thirdArray using for loop: " << endl;  for(int i=0; i<4;i++){  cout << thirdArray[i];  }  cout << endl;  cout << "Changing one digit in third array: " << endl;  thirdArray[0]='N';  cout << "Print thirdArray using for loop: " << endl;  for(int i=0; i<4;i++){  cout << thirdArray[i];  }  return 0;  } | Print firstArray using for loop:  ddddddd  Print firstArray using identifier:  ddddddd  Print secondArray using for loop:  Hello\_  Print thirdArray using for loop:  Test  Changing one digit in third array:  Print thirdArray using for loop:  Nest |

**Try now:** Comment out the statement firstArray[7]='\0'; . What changes do you see in the output?

Arrays of Strings.

|  |  |
| --- | --- |
| StringArrayTest.cpp | Output |
| #include <iostream>  using namespace std;  int main(){  char \*weekDays[7]= {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};  for(int i=0; i<7;i++){  cout << weekDays[i] << endl;  }  return 0;  } | Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  Sunday |

|  |  |
| --- | --- |
| DrawCards.cpp | Output |
| #include <iostream>  #include <cstdlib>  #include <ctime>  #include <cmath>  using namespace std;  int rand\_0toN1(int n);  void draw\_a\_card();  int select\_next\_available(int n);  bool card\_drawn[52];  int cards\_remaining = 52;  char \*suits[4] = {"hearts", "diamonds", "spades", "clubs"};  char \*ranks[13] = {"ace", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "jack", "queen", "king"};  int main() {  int n, i;  srand(time(NULL)); // Set seed for randomizing.  while (1) {  cout << "Enter no. of cards to draw (0 to exit): ";  cin >> n;  if (n == 0)  break;  for (i = 1; i <= n; i++)  draw\_a\_card();  }  return 0;  }  // Draw-a-card function  // Perform a card draw by getting a random 0-4 and a random 0-12.  // Use these to index the strings arrays, ranks and suits.  void draw\_a\_card() {  int r; // Random index (0 thru 12) into  // ranks array  int s; // Random index (0 thru 3) into  // suits array  int n, card;  n = rand\_0toN1(cards\_remaining--);  card = select\_next\_available(n);  r = card % 13; // r = random 0 to 12  s = card / 13; // s = random 0 to 3  cout << ranks[r] << " of " << suits[s] << endl;  }  // Select-next-available-card function.  // Find the Nth element of card\_drawn, skipping over  // those elements already set to true.  int select\_next\_available(int n) {  int i = 0;  // At beginning of deck, skip cards already drawn.  while (card\_drawn[i])  i++;  while (n-- > 0) { // Do the following n times:  i++; // Advance to next card  while (card\_drawn[i]) // Skip past cards  i++;  }  card\_drawn[i] = true;  return i;  }  int rand\_0toN1(int n) {  return rand() % n;  } | Enter no. of cards to draw (0 to exit): 10  seven of hearts  six of diamonds  ace of diamonds  six of hearts  four of diamonds  five of clubs  ten of diamonds  queen of hearts  three of hearts  seven of diamonds  Enter no. of cards to draw (0 to exit): |

**Homework:** Write a similar program to *DrawCards.cpp* for a bag that contains the eight objects: Each item has a unique combination of color (red, blue, orange, green) and shape (ball, cube). Every time an object is picked from the bag, it can’t be picked again, so the number of possible choices decreases by one. The logic should be identical to that in *DrawCards.cpp*, but the array settings will differ. You may also want to give your variables different names, such as items\_remaining and (for the integer array) items\_picked.

The program should also deal with the case when we run out of items. Simulate replacing all items in the bag and continue picking items. (Hint: Read the book)

# Multi-Dimensional Arrays

C++ allows creation of arrays with 2 or more dimensions. Below is an example of declaring, initializing and printing a 2-dimensional array of numbers.

|  |  |
| --- | --- |
| Array2D.cpp | Output |
| #include <iostream>  using namespace std;  int main(){  const int columns = 6;  const int rows = 4;  int my2DArray[rows][columns];  for (int i=0;i<rows;i++){  for(int j=0; j<columns;j++)  my2DArray[i][j] = i+j;  }  for (int i=0;i<rows;i++){  for(int j=0; j<columns;j++)  cout << my2DArray[i][j] << " ";  cout << endl;  }  return 0;  } | 0 1 2 3 4 5  1 2 3 4 5 6  2 3 4 5 6 7  3 4 5 6 7 8 |

Another 2D array example:

|  |  |
| --- | --- |
| Array2DExample2.cpp | Output |
| #include <iostream>  using namespace std;  void printArray(const int [][3]);  const int rows = 2;  const int columns = 3;  int main(){  int array1[rows][columns] = {{1,2,3},{4,5,6}};  int array2[rows][columns] = {1,2,3,4,5};  int array3[rows][columns] = {{1,2},{4}};  cout << "Values in array1 are:" << endl;  printArray(array1);  cout << "Values in array2 are:" << endl;  printArray(array2);  cout << "Values in array3 are:" << endl;  printArray(array3);  return 0;  }  void printArray(const int a[][columns]){  for (int i=0;i<rows;i++){  for(int j=0; j<columns;j++)  cout << a[i][j] << " ";  cout << endl;  }  } | Values in array1 are:  1 2 3  4 5 6  Values in array2 are:  1 2 3  4 5 0  Values in array3 are:  1 2 0  4 0 0 |