**Arrays**

Why do we need arrays?

1. Write a program that inputs three test scores and computes the average.
2. Write a program that inputs test scores until -1 and computes the average
3. Can you print all the test scores now?
4. Can you find the highest?
5. What if you want to input 100 test scores?

An array is a collection of objects of the same type.

To declare an array:

**type name[SIZE];**

Where type is int, float, char, string, boolean

name is the name of the array(use lowercase)

SIZE is a constant or number

Examples:

int testScores[10];

char letters[60];

const int MAX\_TEMPS = 100;

float temps[MAX\_TEMPS];

6. To save test scores, use an array. Declare an array of up to 100 test scores.

**Declaring an array allocates space in memory for SIZE values.**

Example:

int testScores[10];

Yields:

|  |  |
| --- | --- |
| testScores[0] |  |
| testScores[1] |  |
| testScores[2] |  |
| testScores[3] |  |
| testScores[4] |  |
| testScores[5] |  |
| testScores[6] |  |
| testScores[7] |  |
| testScores[8] |  |
| testScores[9] |  |

Write array declarations for the following: type name[SIZE];

1. 200 floating point weights *float weights[200];*
2. an array named k of 20 characters *char k[20];*
3. an array of 50 int temperatures
4. an array of 25 names
5. an array of 10 boolean flags
6. an array of 30 int test scores
7. an array of 70 zip codes
8. an array of 1000 song titles
9. an array of 100 t or f test answers
10. an array of 100 class names

We are assuming zero-based arrays, which means array indexes begin at 0 and end at SIZE-1

**To access one value in the array:**

arrayName[index]

where index may be a number, variable, or expression

Example:

cin >> testScores[5];

testScores[3] = 15;

cout << testScores[9];

Using the definitions on the previous page, write the appropriate statements:

1. assign 45.5 to the first element of the weight array *weights[0] =45.5 ;*
2. assign ‘X’ to the last element of the array k *k[19] = ‘X’;*
3. cout << the 2nd element of the temps array
4. Input the 5th name in the names array
5. Set the 4th flag in the flags array to True
6. Add 1 to the 7th test score in the test scores array
7. Input the 1st zip code in the zip codes array
8. cout << the last song title in the song titles array
9. Set the first 2 answers to ‘T’ in the test answer array
10. set the last class name to “COSC175”

You cannot access the array as a unit, **the following operations are invalid**:

cout << testScores;

cin >> testScores;

testScores = 0;

**To access the entire array, use the counted loop.**

int testScores[10];

**for (int ix = 0; ix < 10; ix++)**

**testScores[ix] = 0;**

**Hint: *One way to figure out how to do operations on an entire array is to think how you would do the operation to one element of the array.*** For example, to output the first element of the testScores array:

cout << testScores[0;

To cout << the entire testScores array use a counted loop:

for (ix = 0;ix < 10;ix++)

cout << testScores[ix];

Given the array testScores, int testScores[10];

write the code to do the following:

1. Set all the values to zero
2. Print all the values
3. Input all the values
4. Double all elements in the array
5. Increment all the values
6. Print the elements in reverse order
7. Sum all the values
8. Compute the average
9. Determine the largest value in the array

**Using while loops with arrays:**

* If we are not working with the entire array, for example a search or reading from a file, we must use a while loop.
* The while loop must include a check for the maximum number of elements as well as any other checks.

Examples:

Given: float temps[10]

Searching the temps array:

ix = 0;

found = FALSE;

while (ix < 10 && !found)

{

if (temps[ix] = searchElement)

found = TRUE;

else

ix = ix + 1

}

Reading from a file:

ix = 0;

infile >> temp; //read into a temporary variable

while (ix < 10 && infile)

{

temps[ix] = temp;

ix = ix + 1;

infile >> temp;

}

cout << “There were “ << ix << “ values read in.”;

Exercises:

1. Write a solution to read in up to 100 numbers from the user and store in an array. Entering 0 causes input to stop. Count the number of positive and negative ints. Print the numbers back to the screen.
2. Write a solution that will read up to 10 letters into an array and write the letters back to the screen in the reverse order. For example, if the input is **abcde** then the output should be **edcba**. Use a period as a sentinel value to mark the end of the input.
3. Create an array that contains a list of 100 valid charge numbers. Assume the array contains the charge numbers. Let the user enter a charge number. The program should determine if the number is valid by checking for it in the array you created. A simple linear search should be used to locate the number entered by the user. If the user enters a number that is in the array, the program should cout << a message saying the number is valid. If the user enters a number that is not in the array, the program should cout << a message indicating the number is invalid.

**Two dimensional arrays:**

To create a two dimensional array:

d**ataType arrayName[SIZE1][SIZE2];**

where SIZE1 is the number of rows and SIZE2 is the number of columns

Example:

float matrix[4][3];

|  |  |  |  |
| --- | --- | --- | --- |
|  | [0] | [1] | [2] |
| [0] | 6.7 | 81 | 9.5 |
| [1] | 56.7 | 9.8 | -45 |
| [2] | 54.8 | 7.8 | -5.6 |
| [3] | -93.2 | 117.4 | 98.6 |

1. What is the value of matrix[0][2]? *9.5*
2. What is the value of matrix[2][1]?
3. What is the value of matrix[1][2]?
4. What is the value of matrix[3][1]?

**To access the entire array, use nested counted loops:**

Example:

//print the above matrix in table form

int row, col;

for ( row = 0; row < 4; row++)

{

for ( col = 0; col < 3; col++)

cout << matrix[row][col] << " ";

cout << endl;

}

1. Create a character table called junk with 4 rows and 5 columns.
2. Set every value in junk to all blanks.