

# **Arrays**

Spring 2017

Victor Eijkhout and Charlie Dey

# Arrays

a definition

a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

# One dimensional arrays

declaration

//Examples:

```
{
    int numbers[] = {5,4,3,2,1};
    cout << numbers[3] << endl;
}
{
    int numbers[5]{5,4,3,2,1};
    cout << numbers[3] << endl;
}
```

# One dimensional arrays

declaration and iterating through

//Examples:

```
{  
    int numbers[] = {5,4,3,2,1};  
    cout << numbers[3] << endl;  
    for (int i=0; i<5; i++)  
        cout << numbers[i] << endl;  
}
```

# One dimensional arrays

Declaring, reading and writing

//Examples:

```
{  
    int numbers[5];  
    for (int i=0; i<5; i++)  
        numbers[i] = 2*i;  
  
    for (int i=0; i<5; i++)  
        cout << numbers[i] << endl;  
}
```

# Exercise 1.

Write a program that creates an array of 100 random integers  
to submit:

name your program: 'arrays1\_<your\_eid>.cc'

cp this to the submission folder: /home/charlie/submission/Arrays

```
// this code generates a random number between
// 0 and 10;
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    srand((unsigned)time(0));
    int random_integer = rand()%10;
    cout << random_integer << endl;
}
```

# One dimensional arrays

as an arguments to a function

Note: Arrays are passed by reference!

```
#include <iostream>
using namespace std;

double getAverage(int arr[],
    int size)
{
    int i, sum = 0;
    double avg;
    for (i = 0; i < size; i++)
    {
        sum += arr[i];
    }
    avg = double(sum) / size;
    return avg;
}
```

```
int main () {

    double avg;
    int numbers[5];
    for (int i=0; i<5; i++)
        numbers[i] = 2*i;

    avg = getAverage(numbers, 5) ;

    cout << "Average value is: " <<
    avg << endl;

    return 0;
}
```

# Exercise 2.

Using your random array generators,

Write 2 functions that take an array as an argument

- one function that finds the maximum value and the index of the maximum value
- one function that finds the minimum value and the index of minimum value

to submit:

name your program: 'arrays2\_<your\_eid>.cc'

cp this to the submission folder: /home/charlie/submission/Arrays



# Exercise 3.

Using your random array generator,

Write a function that takes an array and 2 index locations and swaps the values of the array at the 2 index locations.

to submit:

name your program: 'arrays3\_<your\_eid>.cc'

cp this to the submission folder: /home/charlie/submission/Arrays

# Exercise 4.

Using your random array generator,

Write a function that will sort your randomly generated array from smallest to largest, by traversing your array and swapping values of adjacent indices if  $a[i] > a[i+1]$

How can you test that your array is sorted?

to submit:

name your program: 'arrays4\_<your\_eid>.cc'

cp this to the submission folder: /home/charlie/submission/Arrays

# Multi-dimensional arrays

a definition

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is basically a list of one-dimensional arrays.

To declare a two-dimensional integer array of x rows and y columns

```
type arrayName [ x ][ y ];
```

# Multi-dimensional arrays

a definition

```
int a [ 3 ] [ 4 ];
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[ 0 ][ 0 ]	a[ 0 ][ 1 ]	a[ 0 ][ 2 ]	a[ 0 ][ 3 ]
Row 1	a[ 1 ][ 0 ]	a[ 1 ][ 1 ]	a[ 1 ][ 2 ]	a[ 1 ][ 3 ]
Row 2	a[ 2 ][ 0 ]	a[ 2 ][ 1 ]	a[ 2 ][ 2 ]	a[ 2 ][ 3 ]

# Multi-dimensional arrays

declaration

//Examples:

```
int a[3][4] = {  
    {0, 1, 2, 3} ,    /* initializers for row indexed by 0 */  
    {4, 5, 6, 7} ,    /* initializers for row indexed by 1 */  
    {8, 9, 10, 11}    /* initializers for row indexed by 2 */  
};
```

```
int b[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

# Multi-dimensional arrays

declaration, reading

```
#include <iostream>
using namespace std;

int main () {
int a[3][4] = {
    {0, 1, 2, 3} ,    /* initializers for row indexed by 0 */
    {4, 5, 6, 7} ,    /* initializers for row indexed by 1 */
    {8, 9, 10, 11}    /* initializers for row indexed by 2 */
};

    // output each array element's value
    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ ) {
            cout << "a[" << i << "][" << j << "]: ";
            cout << a[i][j]<< endl;
        }

    return 0;
}
```

# Multi-dimensional arrays

declaration, writing and reading

```
#include <iostream>
using namespace std;

int main () {
    int a[3][4];

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ ) {
            a[i][j] = i+j;

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ ) {
            cout << "a[" << i << "][" << j << "]: ";
            cout << a[i][j]<< endl;
        }

    return 0;
}
```

## Exercise 5.

Write a program that creates a 2 dimensional array of 100 rows and 100 columns of random integers

```
// this code generates a random number between
// 0 and 10;
#include <iostream>
#include <cstdlib>
using namespace :: std;
int main()
{
    srand((unsigned)time(0));
    int random_integer = rand()%10;
    cout << random_integer << endl;
}
```



# Multi-dimensional arrays

as an arguments to a function

Note: Arrays are passed by reference!

```
#include <iostream>

using namespace std;

int multiplyByC(int arr[][4], int rows,
int cols, int C)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            arr[i][j] *= C;
        }
    }
    return 0;
}
```

```
int main ()
{
    int a[3][4];

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ )
            a[i][j] = i+j;

    multiplyByC(a, 3, 4, 5);

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ ) {
            cout << a[i][j]<< endl;
        }

    return 0;
}
```

# Multi-dimensional arrays

as an arguments to a function

Note: Arrays are passed by reference!

```
#include <iostream>

using namespace std;

int multiplyByC(int arr[][4],
               int ans[][4], int rows, int cols,
               int C)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            ans[i][j] = arr[i][j]*C;
        }
    }
    return 0;
}
```

```
int main ()
{
    int a[3][4], c[3][4];

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ )
            a[i][j] = i+j;

    multiplyByC(a, c, 3, 4, 5);

    for ( int i = 0; i < 3; i++ )
        for ( int j = 0; j < 4; j++ ) {
            cout << c[i][j]<< endl;
        }

    return 0;
}
```

# Exercise 6.

Write a function that creates a 100x100 identity matrix , a matrix where the diagonal values are 1's and the rest of the values - the upper and lower triangles - are 0's

to submit:

name your program: 'arrays6\_<your\_eid>.cc'

cp this to the submission folder: /home/charlie/submission/Arrays

# Exercise 7.

- Using your random number generator, create 2 random 100x100 matrices.
- Write a function that multiplies the 2 matrices together and puts the result in a third matrix.
- Test your matrix multiplication function by multiplying your random matrix with the same size identity matrix, the result will be the same as the original matrix.

Matrix Multiplication Algorithm:

- Input: matrices A and B
- Let C be a new matrix of the appropriate size
- For i from 1 to n:
  - For j from 1 to p:
    - Let sum = 0
    - For k from 1 to m:
      - Set  $\text{sum} \leftarrow \text{sum} + A[i][k] \times B[k][j]$
    - Set  $C[i][j] \leftarrow \text{sum}$
- Return C

to submit:

name your program: 'arrays7\_<your\_eid>.cc'  
cp this to the submission folder:  
/home/charlie/submission/Arrays

# Multi-dimensional arrays

a definition

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is basically a list of one-dimensional arrays.

To declare a two-dimensional integer array of x rows and y columns

```
type arrayName [ x ][ y ];
```