CSC 215

Math and Computer Science

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A Legacy of Excellence

2D Arrays

- Sometimes called a matrix
- Holds multiple rows / columns in a single variable name
- Easy to access entire array with nested loops
  - The outer loop is usually the row
  - The inner loop is usually the column

Visual 2D array

- An array with 3 rows and 4 columns

	0	1	2	3
0				
1				
2				

- It is laid out in contiguous memory, no 2d memory

Declaration

- dataType variableName[NUMROWS] [NUMCOLS];
  - Example: int exams [40][4]; // 40 rows, 4 columns – 40 students, 4 exams
    - // NUMROWS and NUMCOLS must be constant integers
- Can use constant variables

```
const int MAXSTUDENT = 40;
const int EXAMCOUNT = 4;
int exams[MAXSTUDENT][EXAMCOUNT];
```

Temperature Example

- Keep track of hourly temperatures for January
- Need a 2D array with 31 rows and 24 columns
  - 31 days – each row holds the temperatures for that day
  - 24 columns – record hourly temperatures for each day

```
int hourlyTemps[ 31 ][24];
```

Usage

- Each element of the 2d array is accessed by giving the row index and the column index
  - To refer to the 10th day in January at the 8th hour
    - Remember, the 10th day would be at index 9
    - The 8th hour would be at index 8, I view midnight as the zero hour

      0 = midnight, 1 = 1:00, 2=2:00…. 8=8:00

```
hourlyTemps[9][8] = currentTemp;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Components of array

int arrayName[10][20];

- arrayName – contains the address of the array in memory
- Subscripts
  - [rowIndex][columnIndex] represents a position in the array
  - arrayName[rowIndex][columnIndex] is a single value stored in the 2d array
  - arrayName[rowIndex] represents a 1d array (the entire array)
    - Can pass a single row from a 2d array to a function that expects a 1d array.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Initializer Lists

- Initialize at declaration
- Initializer list of initializer lists

```
int myArray[3][4] = { {0,3,2,6} , {9,9,23,7} , {8,3,4,5} };
```

	0	1	2	3
0	0	3	2	6
1	9	9	23	7
2	8	3	4	5

- Each initializer list is a row

Initializer List - Continued

- Elements not given will be set to zero

```
int myArray[3][4] = { {0,3,2} , {9,9} , {8} };
```

	0	1	2	3
0	0	3	2	0
1	9	9	0	0
2	8	0	0	0

Initializer List - Continued

- Can give elements in 1 long initializer list, there are 12 integers

```
int myArray[3][4] = { 1,2,3,4,5,6,7,8,9,10,11,12};
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Initializer List - Continued

- Can give elements in 1 long initializer list, there are 12 integers

```
int myArray[3][4] = { 1,2,3,4,5,6};
```

	0	1	2	3
0	1	2	3	4
1	5	6	0	0
2	0	0	0	0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Initializer List - Continued

- Can initialize array to all zeros

```
int myArray[3][4] = { 0 };
```

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Initializer List - Continued

- Can not initialize array to all ones

```
int myArray[3][4] = { 1 };
```

	0	1	2	3
0	1	0	0	0
1	0	0	0	0
2	0	0	0	0

- Must use for loop

```
for(i=0; i<3; i++)
    for(j=0; j<4; j++)
        myArray[i][j] = 1;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Filling array from an Input Stream

```
Int myArray[3][4];

for( i=0; i<3; i++)                             // Go through each row
        for( j=0; j<4; j++)                     // Then each column in the row
                cin >> myArray[i][j];
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Passing 2D Arrays to Functions

- Function prototype

  returnType functionName(dataType arrayName[][#]);

  Example: void printArray2d(int myArray[][4], int rows);

  - Number of rows does not need to be specified in the array variable
  - Number of columns is **REQUIRED** to be passed in the array variable
    - Tells compiler that every 4 integers is the start of a new row
  - Number of rows and Columns can be passed as separate parameters if needed.
  - I highly recommend always passing the rows, column is not required unless you want a variable.

Passing 2d Arrays to Functions

- Function Definition
  - Header is the same as the prototype but without the semicolon
  - The array is passed by reference

SOUTH DAKOTA
SCHOOL OF MINES & TECHNOLOGY

Passing 2d Arrays to Functions

Assume int myArray[3][4];

```
void printArray( int myArray[][4], int rows, int cols);
```

- Function Call

```
printArray( myArray, 3, 4 ) ;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Passing a row to a function

Assume:

```
int myArray[3][4];
void printArray1D( int a[], int size );
```

```
for(i=0; i<3; i++)
        printArray1D( myArray[i], 4 ); // pass a row from 2d.
```

THERE IS NO WAY TO PASS A COLUMN

Sample Code

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

const int NUM_ROWS = 10;
const int NUM_COLS = 4;

void print_matrix( int matrix[][NUM_COLS],
                   int rows );

int main()
{
    int my_2D_array[NUM_ROWS][NUM_COLS];
    int i, j;

    for(i = 0; i < NUM_ROWS; i++)
    {
        for(j = 0; j < NUM_COLS; j++)
        {
            my_2D_array[i][j] = 2*i + j;
        }
    }

    //prints all data
    print_matrix(my_2D_array, NUM_ROWS);
    cout << endl << endl;

    //prints just first 4 rows
    print_matrix(my_2D_array, 4 );
    return 0;
}


void print_matrix( int matrix[][NUM_COLS],
                   int rows )
{
    int row, col;

    for(row = 0; row < rows; row++)
    {
        for(col = 0; col < NUM_COLS; col++)
            cout << setw(6) << matrix[row][col] ;
        cout << endl;
    }
}
```