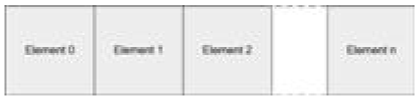
Arrays in C++

Array

- Just as in Matlab, a collection of related data is called an array
- Arranged in rows and/or columns
- Stored in a single variable

One Dimensional Array



- One variable that consists of n elements
- One-dimensional arrays in C++ are similar to the Matlab row array
- Note that element numbers start at 0

Array Declarations

- Array variables must be declared in C++
- Examples
  - int temps[4];

    creates an array called temps that holds four integers
  - float nums[20];

    creates an array called nums that holds 20 floating point numbers

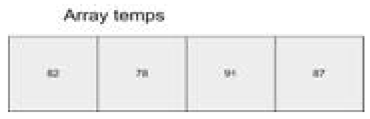
Array Initialization

- Arrays can be initialized as they are declared
  - int temps[5] = {82, 78, 91, 85, 73};
- Do not have to specify array size if initial values are used. Array automatically created large enough to hold initial values
  - int temps[ ] = {82, 78, 91, 85, 73};

Subscripts

- Individual array elements accessed by using subscripts
- Subscript is the element number
- Subscript must be an integer within the range of the array
- In C++, subscripts always start at 0

Examples



```
x = temps [0];      // x is 82
y = temps [2];      // y is 91
temps[3] = 23;      //changes content of last
                    // element from 87 to 23
```

Warning

- C++ will not flag an error if the array subscript goes out of bounds
- Ex:

```
int A[3];       // declare A with 3 elements
A[5] = 2;       // subscript out of range
```

  This is perfectly legal, but will lead to unpredictable behavior
- Beware!

Arrays and For Loops

- C++ is a general purpose programming language that has not be optimized for mathematics as Matlab has
- Cannot do array manipulation as easily
- Almost always need to use a for loop to step through the array elements

Examples

```
//Print out the contents of array A, which has
  //5 elements
     for (int i = 0; i < 5; i++) {
          cout << A[i] << endl;
     }
```

Examples

```
//Set all 100 elements of array B to 0
     for (int j = 0; j < 100; j++) {
          B[j] = 0;
     }
//Copy array B to array C
     for (int k = 0; k < 100; k++) {
          C[k] = B[k];
     }
```

Example

Generate (x, y) points for the equation
$y = mx + b$

x should vary between -10 and +10

m and b should be chosen by the user

Algorithm

- Prompt for and read in m (the slope) and b (the offset)
- Create an array X containing values between -10 and +10
- For each element in X, calculate the corresponding Y using the formula

  $Y = mX + b$

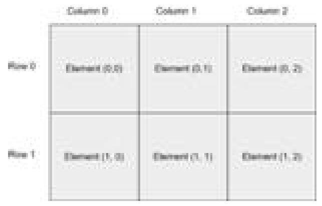
```
#include <iostream>
using namespace std;
int main()
{
   int X[21], Y[21];
   int m, b, x_val;
   // prompt for and read in slope and offset
   cout << "Please enter the slope:  ";
   cin >> m;
   cout << "Please enter the offset: ";
   cin >> b;
```

```
   //x values start at -10 and go to +10
   x_val = -10;

   // Loop creates X[i] and Y[i] and prints out x,y pair
   for (int i = 0; i < 21; i++) {
        X[i] = x_val;
        x_val++;
        Y[i] = (X[i] * m) + b;
        cout << "(" << X[i] << ", " << Y[i] << ")" << endl;
   }
return 0;
}
```

Two-Dimensional Arrays

- Just as in Matlab, C++ supports two dimensional arrays
- A two dimensional array has rows and columns, like a table
- Sometimes called a matrix



	Column 0	Column 1	Column 2
Row 0	Element (0,0)	Element (0,1)	Element (0, 2)
Row 1	Element (1, 0)	Element (1, 1)	Element (1, 2)

Two Dimensional Array Declarations

- Similar to one-dimensional arrays
- Examples
  - int temps[4][4];

    creates an array called temps with 4 rows and 4 columns (holds 16 integers)
  - float nums[20][2];

    creates an array called nums with 20 rows and 2 columns (holds 40 floating point numbers)

Two Dimensional Array Initialization

- Arrays can be initialized as they are declared
  - int temps[2][2] = {{82, 78}, {91, 85}};
  - Values for each row held in { }

Subscripts

- Individual array elements accessed by using subscripts
- Subscript is row number, column number
- Both parts of the subscript must be integers within the range of the array
- Subscripts start at 0

Examples

```
A is 1 1 1
     2 2 2
     3 3 3
m = A[0,0];     // m is 1
n = A[2, 1];    // n is  3
```

Examples, con't

```
A[1, 1] = 5;        A is  1 1 1
                          2 5 2
                          3 3 3
A[0, 2] = 8;        A is  1 1 8
                          2 5 2
                          3 3 3
```

Nested For Loops

- Use nested for loops to step through two dimensional arrays
- Usual format is:

```
for each row
  for each column
     do something to array[row][column]
```

Example

```
// Initialize array to all 1's
int ex[3][2];
for (int row = 0; row < 3; row++){
  for (int col = 0; col < 2; col++) {
     ex[row][col] = 1;
  }
}
```

Example

```
// Display contents of an array
int ex[3][2] = {{1,1}, {2,2}, {3,3}};
for (int row = 0; row < 3; row++){
   for (int col = 0; col < 2; col++) {
        cout << ex[row][col] << "  ";
   }
   cout << endl;  // new line at end of each row
}
```