Computer Programming

An-Najah N. University
Computer Engineering Department
Luai Malhis, Ph.D,

Arrays: One and Two dimensional

Arrays

- So far we have only used scalar variables:

  variable with single value
- But many things require set of related values: student grades in exam, letters in a aname.
- Arrays are used to store a collection of related values. Example 100 int values.
- Instead of declaring an individual variable to to each element in the array. We declare one special variable to all to all the elements.

Array Declaration

- Array declaration has the format:

  data type arrayname[size];

  where size is an integer constant > 0 that represents the maximum number of values (elements), that you want to store in the array.
- Each element of the array is like little varaible
- All elements of the array are of the same type.
- To reach an element use arrayname[index]

  where index in the range 0 to size -1

Example

- Suppose I want to store grades for 100 students.

  Then I need to do either:
- Declare 100 scalar variables score1, score2, …….. score100.

  In this case I need 100 cin statements to read the graders from the keyboard:

```
cin >> score1
cin >> score2
  ……..
cin >> score100
```

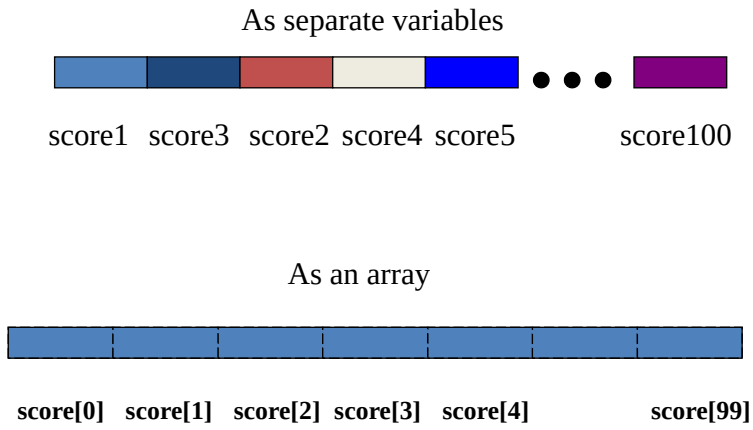
This is a tedious process and makes programming boring and difficult.

Example continue

- Instead: I could declare the array: int score[100];
  which allocates space for 100 integer values.
- Using a loop and one cin statement I can read the 100 scores into the array as follows:
- for (int I =0; I < 100; i++)

  cin >> score[i];
- This technique allows me to declare and array of very large size and be able to process the array using loops

Schematic of Memory



Some Constraints

- Array name must be valid variable name.
- Array size must be constant value > 0.

- Examples of valid array declarations:
  double salaries[100]; char name[30];

- Examples of invalid array declarations:

```
int x;  double salaries[x]; // not constant
 char name[-5];  // constant < 0
int players[];   // must have a constant
double average[5.5];  // must be integer constant
```


Accessing Array Elements

- The individual elements of an array may be accessed by the array name and the subscript or index.

  score[3], score[78], score[99]
- Subscripts in C++ begin with zero, so the first element of the array with size n has the index or subscript 0 and the last element has the index or subscript n-1.
- Any subscript outside this range (< 0 or >= n) is an invalid access to array element.

Array Types

- All array elements are of the same type.
- We can declare an array of integers
  int IA[40];
- We can have an array of doubles
  double DA[50];
- We can have array of characters
  char CA[30];

Special case: strings array of characters that ends with special characters called null '\0'

- You can have an array of any other data type:
  float, short, long,….

Storing Elements in an Array

- Array elements may be input one at a time

```
score[0] = 78; // assignment
score[5] = 80;
cin >> score[81];
 cin >> score[66];
```

- Or use a loop:

```
for (int j = 0; j < n; j ++)
    cin>>score[j];    // from the keyboard
for (int j = 0; j < n; j++)
      score[j] = j *2;  // using an expression
for (int j = 0; j < n; j++)
      score[j] = 70; // same value in all locations
```


Initialization at Declaration

Array elements can be initialized at time of declaration.

```
float temps[4] = {78.5, 79.8, 85.4, 86.2};
```

If you are initializing the elements at the point of declaration the size declarator may be omitted.

```
float temps[ ] = {78.5, 79.8, 85.4, 86.2};
```

the size will implicitly be set equal to the number of elements specified (four here).

float temps[]; // is invalid must give size if not initialized

float temp[4] = {1,6.5,88.2,19,72,31}; // invalid size is too small

Initialization at declaration continues

- int x[10] = {4,5,6}; Store 4,5,and 6 in the first 3 locations respectively and 0 in the rest
- char name[3] = {'M','A','Y'}; array of characters
- Char name[4] ={'M','A','Y','\0'}; string
- char name[10] = {'M','A','Y'}; // string all elements at locations 3,4..9 are filled with '\0'
- char name[10] = "MAY"; string like above
- double values[5] = {1.2, 4,5, 6.2}; locations 3, 4 are filled with 0.0
- char name[] = "Ahmad"; // size is 6
- char name[5] "Ahmad"; // invalid size is small

Declarations and Initializations Continue

- int x[3] = {10,5,13,4,6}; invalid size is too small for specified elements.
- int x[10] ={4,5,,6,7}; is invalid only last values in the array may be omitted. The correct declaration in this case is int x[10] = {4,5,0,6,7};
- int X[10]; x = {4,5,6,7}; is invalid because declaration and the initialization are done in two separate statements .
- char x ="abc"; // x is a single char not array

Strings

- Strings are arrays of characters.
- Strings have some special properties that numeric arrays do not have.
  - The size should allow for the null character '\0'
  - Input may be an entire array at a time

```
cout<<"Please enter a filename"<<endl;
cin>>filename
char name[6] = "Ahmad";   //valid
char name[10] = "Ahmad";  //valid
char name[5] = "Ahmad"; // invalid  size is small
```


Outputting Data from an Array

- Array elements can be outputted individually

```
cout<<score[0]<<" "<<score[5];
```

- Or using a loop

```
for (int j = 0; j < n; j ++)
        cout<<score[j]<< endl;
```

- All elements of arrays cannot be outputted by just using the array name.

```
cout<<score;    // will not work!
```

exception to this if the array is a string

```
char B[10] = "ahmad"; cout << B; // output
    entire string in one cout statement
```


Array Declaration Examples

- Define arrays of types int, char, double and floats each of size 10 elements.

  Solution: int A[10]; char B[10];
  double C[10]; float D[10];

- Declare and initialize an integer array of 100 elements that contains {6,12,4,9,15,0,0….0}.

  Solution: int IA[100] = {6,12,4,9,15};

- Declare and initialize an array of 100 doubles that contain {0.0, 2.0, 4.0, 6.0,…198.0}.

  Solution: double DA[100];

  for (int i= 0; I < 100; i++) DA[i] = I * 2;

Examples Continue

- Define an arrays of characters size 26 elements. Store all small letters in the array.

  solution 1: char CA1[26];

```
for (int i = 0; i < 26; i++)
    CA1[i] = (char) ('a'+i);
```

  solution 2: char CA2[26];

```
for (char ch ='a'; ch <= 'z'; ch++)
    CA2[ch – 'a'] = ch;
```

- Declare an array of 10 characters store "World" in it.
  solution: char st[10] = "world";

Examples Continue

- Declare and initialize a string with the word "Palestine". Array size is selected automatically
  solution char st2[] = "palestine";
- Declare an Array of 100 doubles Then read values into the array from the keyboard. Then multiply each read value by 2 and print the final result in the array starting at location 99 then 98 … down to 0.

  Solution:

```
double A[100];
for (int I =0; I < 100; I++) cin >> A[i];    //read data
for ( I = 0; I < 100; i++) A[i] *= 2;        //process data
for (I = 99; I >=0; i--)  cout << A[i] << endl; //output data
```


Caution!

- C/C++ has no checking on the bounds of the array. Outside the range 0 to size – 1
- Therefore you can store elements with subscripts larger than the size declarator.
- However, these values could be corrupted because the memory spaces have not been officially reserved.
- Retrieving data stored in out-of-bounds elements is compiler and/or platform dependent.

Array Processing

Array elements can be used in assignment statement, output statement, and to perform operations on them, in a similar manner as individual variables.

```
Int  IA[10] = {1,2,4,};  int x, y =12;
x = y * IA[2];
IA[4]= sqrt(IA[3]) + x;
for ( IA[0] =5, int j =1; j < 10; j++)
    IA[j] = IA[j-1] * 2;
for (int sum =0, int j=0; j <100;j++)
    sum += IA[j];
```

Examples

Declare an array of grades for 50 students. Ask the user to enter grade for each student compute and display the average.

```
int grades[50];     int i, sum =0;
for (i=0;i<50;i++) {
    cout << "Enter grade:";      cin >> grades[i];
    sum = sum + grades[i]; }
cout << "Average is:" << sum/50.0 ;
```

More processing:

```
int fail =0; int pass =0;
for ( i=0; I < 50; i++)  if (grade[i] >= 60) pass++;
cout <<pass<<" passed";
for ( i=0; I < 50; i++)  if (grade[i] < 60) fail++;
cout << fail << " failed";
```


Examples 2

Given an array of 100 int values called A, compute the smallest, and the largest values in the array.

Solution:

```
int smallest =0; int largest = 0;
for ( int I = 0; I < 100; I++) {
   if (A[i] < smallest)  smallest = A[i];
   if (A[i] > largest) largess = A[i];
}
cout << "The smallest value is " << smallest << endl;
cout <<  "The largrest value is " << largest;
```


Examples 3

Read 120 int values from the keyboard store all negative values in an array called negative and all positive values in an array called positive. Select appropriate array size for negative and positive arrays.

Soultion:

```
int positive[120]; int negative[120]; int num;
int I,j,k; I = j = k =0;
for (; I < 120; i++) {
   cin >> num;
    if (num >= 0) positive[j++] = num;
    else            negative[k++] = num;
}
```


Examples 4

- Given an array of 1000 characters called AC, compute the count small letters in the array.

```
for ( int csmall =0, int I = 0; I < 1000; i++)
     if ( AC[i] >= 'a' && AC <= 'z") csmall++;
```

- Given an array of int values of size 100 (AI) print the average of all 3 digit values in the array.

```
double sum =0; int count =0;
for (int I =0; I < 100; I++)
     if (AI[I]>99  &&  AI[I]<999){sum += AC[i];  count++};
cout << "The average is" << sum/count;
```

Examples 5

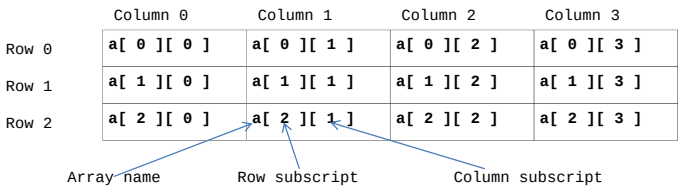
- Define an array of 100 int and store the sequence:
  1 2 3 5 8 13 21 ….
  int A[100]; A[0] = 1; A[1] = 2;
  for (int i =2; i <100; i++)
  A[i] = A[i-1] + A[i-2];
- Write a program to do each of the following:
  Given an array of 15 double values called DA.
  Print the index of the smallest value in the array
  The larges tod value.
  Print the sum of all values that are > 100.0
  Replace all negative values with 0.0 in the array.
  Print only array elements that are even and 2 digits

Examples 5 continue

```
int i =0, locsmall = 0, small = DA[0], sum =0, large = DA[0];
 for (int i =0; i < 15; i++) {
   if (DA[i] < small) { small = DA[i]; locsmall =i;}
   if (DA[i] %2  && DA[i] > large) large = DA[i];
   if (DA[i] > 100) sum+= DA[i];
   if(DA[i] < 0) DA[i] = 0;
   if(DA[i] > 9 and DA[i] < 100 && DA[i]%2 ==0)
      cout << DA[i] << endl;
}
cout << "loc = " << locsmall   <<  " sum = " << sum << endl;
cout << Largest value in the array << "large"  << endl;
```


Two Dimensional Arrays (Matrices)

- **Two subscripts `a[ i ][ j ]` are used to reference an element in the matrix**
- **When declared must define number of rows and columns example: `double B[5][6];`**
  - Matix with rows and columns
  - Specify row, then column
  - "Array of arrays" ex a[3][4];
    - **`a[0]`** is an array of 4 elements
    - **`a[0][0]`** is the first element of that array



Matrix Declaration and Initialization

- Define a matrix of types int, char, double and floats each of size 6X4 elements.
  int im[6][4]; char cm[6][4];
  double dm[6][4]; float fm[6][4];

- Define a 4x4 int matrix where each cell contains the sum of row and column indices

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

Matrix Initialization

- A matrix can be initialized at declaration:

  Default of 0 (int), or 0.0 (double) or '\0' (char) for not specified values like the one dimensional array

  – Initializes grouped by row in braces

```
int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```

Row 0	1	2
Row 1	3	4

```
int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };
```

Row 0	1	0
Row 1	3	4

When initializing a matrix at declaration the number of columns must be specified but number of row may be not specified.

Initialization at Declaration

int M[3][3] = {{1,2,3},{4,5,6},{7,8,9}};

int [][3] = {{1,2},{3},{4,5,6}}; specifies a matrix with three rows and three columns missing values are filled with 0. {1,2,0},{3,0,0},{4,5,6}

int A[][] = {{1,2},{3},{4,5,6}}; is invalid declaration because number of columns not defined

int A[2][] = {{1,2},{3},{4,5,6}}; is invalid declaration because number of rows must be at least three

int A[4][4] = {{1,2},{3},{4,5,6}}; valid declaration fills missing values with 0s.
{{1,2,0,0},{3,0,0,0},{4,5,6,0},{0,0,0,0}}

Reading and Printing Matrix Elements

- When reading elements from the keyboard must read it one item at a time: Int A[10][20];

```
for (int r =0; r < 10; r++)        // for each row
  for (int c =0; c < 20; c++)      // for each column in  a row
      cin>> A[r][c];               // read elements
```

- When printing elements on the screen must print it one item at a time: Int A[10][20];

```
for (int r =0; r < 10; r++) {        // for each row
  for (int c =0; c < 20; c++)        // for each column in  a row
     cout << A[r][c] << " ";         // print element
  cout << endl;                      // print end line
}
```


Read and Print Examples

Give int M2[6][4];
Read elements and store in matrix row0 then row1 ...

```
for (int r= 0; r< 6; r++)
    for (int c = 0; c < 4; c++)
        cin >> M2[r][c]
```

Print the elements in the matrix row0 then row1 …

```
for (int r = 0; r < 6; r++) {
    for (int c = 0; c < 4; c++)
        cout << M2[r][c];
    cout << endl; }
```

Read and Print Examples 2

Read Elements and store column0 then column1 ….

```
int M2[6][4];
for (int c = 0; c < 4; c++)
      for (int r = 0; r < 6; r++)
          cin >> M2[r][c];
```

Print the Matrix elements column0 then column1 ….

```
for (int c = 0; c < 4; c++) {
    for (int r = 0; i < 6; r++)
          cout << M2[r][c];
    cout << endl;
}
```


Important Notes

- int A[][]; or int A[][4]; or int A[5][]; are invalid declarations; must specify both rows and columns when declared only (not initialized at declaration)
- Given int A[Row][Col]; Each element is specified using row and column indices. Range 0..Row -1 , 0 .. Col -1
- Example: Given int B[3][5]; Then B[1][3] references the element in 2nd row and 4th column
- To traverse all elements in the matrix need two loops. One loop traverse the rows the other loop traverse the columns.

Matrix Examples

Given 10X10 matrix M of type int.

Print the values of first and last elements in M.

```
cout << "first " << M[0][0];
cout << "last " << M[9][9];
```

Print the value of the first element in row 3 the matrix.

```
cout << M[3] [0];
```

Print the value of row 2 column 4

```
cout << M[2][4];
```

Store 110 in element at 4th row and 5th column

```
M[3][4] = 110;
```

Add two locations and and store result in another location

```
M[2][4] = M[1][2] + M[6][3];
```


Matrix Example 2

Store 0 in the diagonal elements in the matrix.

```
for (int i = 0; i < 10; i++) M[i][i] =0;
```

Store 5 in the last element of every row // last column

```
for (int i = 0; i < 10; i++) M[i][9] = 5;
```

Print in the first element of every column // first row

```
for (int i = 0; i < 10; i++)  cout << M[0][i] << endl;
```

Print the elements in row 5.

```
for (int i = 0; i < 10; i++)  cout << M[5][0] << endl;
```

Find the sum all the elements in column 6

```
for (int i = 0, int sum =0; i < 10; i++) sum+=M[i][6];
```

Matrix Examples 3

Compute smallest, largest and average value in M.

```
int small = M[0][0]; int large = M[0][0]; int sum =0;
 for (int r =0; r < 10; r++)
      for (int c= 0; c < 10; c++) {
           if(M[r][c] < small) small = M[r][c];
           if(M[r][c] > large) large = M[r][c];
           sum + = M[r][c]; }
cout << "smallest - largest" << small << "-" << large;
cout << "the average value " << sum/100.0;
```


Matrix Examples 4

Compute the count of negative values, the sum of all even values and product all values > 10.

```
int countn =0,  sume =0; int product10 =1;
  for (int r =0; r < 10; r++)
     for (int c= 0; c < 10; c++) {
          if (M[r][c] < 0) countn++;
          if(M[r][c] %2 == 0) sume+= M[r][c];
          if(M[r][c] > 10) product *= M[r][c]; }
 cout << "count negative =" << countn << endl;
 cout << "sum even =" << sume << endl;
 cout << "product greater than 10 =" << product10;
```


Matrix Examples 5

- Swap elements in row 2 with elements in row 5

```
for (int c =0, int temp; c < 10; c++) {
      temp = M[2][c];
      M[2][c] = M[5][c];
      M[5][c] = temp; }
```

- Compute the sum of each row and store it in array

```
int Asums[10] = {};
for (int r =0; r < 10; r++)
      for (int c = 0; c < 10; c++)
          Asums[r] += M[r][c];
```


Arrays With Higher Dimensions

- Same arguments is extended to arrays of higher diminutions:
- int A[3][4][5]; is an array of 3 dimensions
- int A[4][5][6][7]; is an array of 4 dimensions.
- When referencing an element in k dimensional array k subscripts must be used.
- For this course we only deal with one and two dimensional arrays.