Arrays

القوائم والمصفوفات
One-Dimensional Arrays
قوائم أحادية البعد

• If you wanted to read in 1000 ints and print them in reverse order, it would take a program that’s over 3000 lines long.
• There’s got to be a better way.
• Don’t worry there is.
• An array of size 1000 would do the trick

إذا أردنا قراءة 1000 عدد صحيح وطباعتهم بطريقة عكسية فإن ذلك يتطلب منك برنامجا مكونا من 3000 سطر.
Single-Dimensional Arrays

- **Generic declaration:**
  
  ```
  typename variablename[size]
  
  - typename is any type
  - variablename is any legal variable name
  - size is a number the compiler can figure out
  ```

  ```
  int  a[10];
  ```

  - Defines an array of ints with subscripts ranging from 0 to 9

  ```
  ```

  - There are 10*`sizeof(int)` bytes of memory reserved for this array.
```cpp
int main()
{
    int values[1000];
    for ( int i=0; i<1000; i++ )
        cin >> values[i];
    for ( int i=999; i>=0; i-- )
        cout << values[i];
    return 124;
}
```
Array Declaration

• First give the **type** of array such as: 
  - int, double, float, bool, etc.
• Then the **name** of the array
• Finally, the **size** of the array inside of square brackets
• Ex-
  - bool TruthArray[12];
  - The size of the array must be a constant int
• One dimensional array
  – A structured collection of components all of the same type, that is given a single name. Each component is accessed by an index that indicates the component’s position within the collection.

• Array position is always started at 0 and goes up to one less then the size.
• You can access any one particular element if you need or want
  – float angle[4];
  – angle[0] = 1.2;
  – angle[1] = 3.4;
  – angle[2] = 0.0;
  – angle[3] = 45.6;
Things You Can Do with an Array Element

• angle[2] = 9.6;
• cin >> angle[2];
• cout << angle[3];
• y = sqrt( angle[1] );
• x = 6.8 * angle[0] + 7.5;
Things You Shouldn’t Do with an Array

• Out-of-bounds array index:
  – An index value that, in C++, is either less than 0 or greater than the array size minus 1.
• You can declare and initialize an array all at one time
• int age[5] = { 23, 10, 16, 37, 12 };
• You can also omit the size of the array and do something like this
• float temperature[] = { 0.0, 112.37, 98.6 };
<table>
<thead>
<tr>
<th>Operation</th>
<th>Allowed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>No</td>
</tr>
<tr>
<td>Assignment</td>
<td>No</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>No</td>
</tr>
<tr>
<td>Comparison</td>
<td>No</td>
</tr>
<tr>
<td>Argument Passage</td>
<td>Reference only</td>
</tr>
<tr>
<td>Return from a function</td>
<td>No</td>
</tr>
</tbody>
</table>
You can write functions that handle all of those operations for you.

For example:

```c
void CopyArray( const int x[], int y[], int size )
{
    for (int i=0; i<size; i++)
        y[i] = x[i];
}
```
It’s always better to declare a const int as the size for the array.

```
const int BUILDING_SIZE = 350;
int occupants[BUILDING_SIZE];
```

You can use BUILDING_SIZE in your for loops as the stop condition.

Now if you need to change the size of your building, you change it in one place.

You can make this a global constant and so all scopes will have access to the constant.
You can pass arrays as arguments to functions.

You will do this for your last two projects.

Arrays are pass-by-reference by default.

You cannot get pass-by-value.

You can pass-by-const-reference by putting const before the variable type in the definition and prototype.
Arrays and Functions

- Function invocation
  
  CopyArray( MyArray, MyOtherArray, 10 );

- Function definition
  
  void CopyArray( const int x[], int y[], int size )
  {
      for (int i=0; i<size; i++)
          y[i] = x[i];
  }
• It is possible to pass just one location of an array to a function.
• This is pass-by-value by default.
• It can also be pass-by-reference and pass-by-const-reference just like every other variable.
Example

Function invocation

Swap( MyArray[4], MyArray[132] );

Function

void Swap( int &x, int &y )
{
    int temp = x;
    x = y;
    y = temp;
}
When thinking about arrays, when ever you type just the name of the array without any brackets you mean the entire array.

When you type the array name with brackets and the number inside, you mean just that particular location.
Two-Dimensional Arrays

• Two dimensional arrays are the same in use except you need an extra set of brackets to indicate the second dimension.

• Example
  – `const int NUM_ROWS = 100;`
  – `const int NUM_COLS = 9;`
  – `float alpha[NUM_ROWS][NUM_COLS];`

• In C++, typically the rows come first.
• Assuming the array declaration from the previous slide, we can:

• Initialize the array

```c
for ( int row=0; row<NUM_ROWS; row++ )
  for ( int col=0; col<NUM_COLS; col++ )
    alpha[row][col] = 0.0;
```
• **Sum Columns**

```cpp
for ( int col=0; col<NUM_COLS; col++ )
{
    total = 0;
    for ( int row=0; row<NUM_ROWS; row++ )
        total += alpha[row][col];
    cout << "Column sum: " << total << '\n';
}
```
To pass a 2-D array into a function you need to specify the size of the second dimension in the prototype and function definition.

For example:

```c
void Copy( int X[][MAX.COL],
           const int Y[][MAX.COL],
           int MAX_ROW);
```
• Initialization of arrays can be done by a comma separated list following its definition.

• For example:

```java
int array [4] = { 100, 200, 300, 400 };
```

– This is equivalent to:

```java
int array [4];
array[0] = 100;
array[1] = 200;
array[2] = 300;
array[3] = 400;
```

• `int array[] = { 100, 200, 300, 400 };`
```c
#include <stdio.h>
int main() {
    float expenses[12]={10.3, 9, 7.5, 4.3, 10.5, 7.5, 7.5, 8, 9.9, 10.2, 11.5, 7.8};
    int count, month;
    float total;
    for (month=0, total=0.0; month < 12; month++)
        total+=expenses[month];
    for (count=0; count < 12; count++)
        printf("Month %d = %.2f K$\n", count+1, expenses[count]);
    printf("Total = %.2f K$, Average = %.2f K$\n", total, total/12);
    return 0;
}
```
Multidimensional Arrays

• For example:
    • defines a two dimensional array
    • a is an array of int[3];

• In memory:
Also can be done by:

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

is equivalent to

```c
a[0][0] = 1;
a[0][1] = 2;
a[0][2] = 3;
a[1][0] = 4;
a[3][2] = 12;
```
```c
#include <stdio.h>
#include <stdlib.h>

int main () {
    int random1[8][8];
    int a, b;
    for (a = 0; a < 8; a++)
        for (b = 0; b < 8; b++)
            random1[a][b] = rand()%2;
    for (a = 0; a < 8; a++)
    {
        for (b = 0; b < 8; b++)
            printf ("%c ", random1[a][b] ? 'x' : 'o');
        printf("\n");
    }
    return 0;
}
```
void mysort(int a[], int size)
{
    int i, j, x;
    for (i = 0; i < size; i++)
    {
        for (j = i; j > 0; j--)
        {
            if (a[j] < a[j - 1])
            {
                /* Change the order of a[j] and a[j-1] */
                x = a[j]; a[j] = a[j - 1]; a[j - 1] = x;
            }
        }
    }
}

int main()
{
    int i;
    int tab[10] = {3, 6, 3, 5, 9, 2, 4, 5, 6, 0};
    for (i = 0; i < 10; i++)
    {
        printf("%d ", tab[i]);
    }
    printf("\n");
    mysort(tab, 10);
    for (i = 0; i < 10; i++)
    {
        printf("%d ", tab[i]);
    }
    printf("\n");
    return 0;
}
// arrays example

#include <iostream>

using namespace std;

int billy[] = {16, 2, 77, 40, 12071};

int n, result = 0;

int main()
{
    for (n = 0; n < 5; n++)
    {
        result += billy[n];
    }
    cout << result;
    return 0;
}
```c
#define WIDTH 5
#define HEIGHT 3
int jimmy [HEIGHT][WIDTH];
int n,m; int main ()
{
    for (n=0;n<HEIGHT;n++)
    for (m=0;m<WIDTH;m++)
    {
        jimmy[n][m]=(n+1)*(m+1);
    }
    return 0; }
```
• Write a function that takes two arrays of ints, of size Size, and swaps them.

• You can use this prototype if you wish:
  
  ```c
  void SwapArray( int ArrayOne[], int ArrayTwo[], int Size );
  ```
```cpp
#include <iostream>

using namespace std;

void printarray (int arg[], int length)
{
    for (int n=0; n<length; n++)
    {
        cout << arg[n] << " ";
    }
}

int main ()
{
    int firstarray[] = {5, 10, 15};
    int secondarray[] = {2, 4, 6, 8, 10};
    printarray (firstarray,3);
    printarray (secondarray,5);
    return 0;
}
```
• `char jenny [20];`
• `Char myword[]={‘h’,’e’,’l’,’l’,’o’,’\0’};`
• `Char myword[]="hello’;`
```cpp
#include <iostream>

using namespace std;

int main ()
{
    char question[] = "Please, enter your first name: ";
    char greeting[] = "Hello, ";
    char yourname [80];
    cout << question;
    cin >> yourname;
    cout << greeting << yourname << "!";
    return 0;
}
```
string mystring;

char myntcs[]="some text";

mystring = myntcs;
Questions

1. An array element is accessed using
   a. a first-in-first-out approach.
   b. the dot operator.
   c. a member name.
   d. an index number.

2. All the elements in an array must be the _________ data type.

3. Write a statement that defines a one-dimensional array called doubleArray of type double that holds 100 elements.

4. The elements of a 10-element array are numbered from _______ to _______.

5. Write a statement that takes element j of array doubleArray and writes it to cout with the insertion operator.

6. Element doubleArray[7] is which element of the array?
   a. The sixth
   b. The seventh
   c. The eighth
   d. Impossible to tell
1. d
2. same
3. double doubleArray[100];
4. 0, 9
5. cout << doubleArray[j];
6. c
7. int coins[] = { 1, 5, 10, 25, 50, 100 };
8. d
9. twoD[2][4]
10. true
11. float flarr[3][3] = {{52, 27, 83}, {94, 73, 49}, {3, 6, 1}};
12. memory address
13. a, d
14. an array with 1000 elements of structure or class employee
15. emplist[16].salary
16. d
7. Write a statement that defines an array coins of type int and initializes it to the values of the penny, nickel, dime, quarter, half-dollar, and dollar.

8. When a multidimensional array is accessed, each array index is
   a. separated by commas.
   b. surrounded by brackets and separated by commas.
   c. separated by commas and surrounded by brackets.
   d. surrounded by brackets.

9. Write an expression that accesses element 4 in subarray 2 in a two-dimensional array called twoD.

10. True or false: In C++ there can be an array of four dimensions.

11. For a two-dimensional array of type float, called f1arr, write a statement that declares the array and initializes the first subarray to 52, 27, 83; the second to 94, 73, 49; and the third to 3, 6, 1.

12. An array name, used in the source file, represents the ________ of the array.

13. When an array name is passed to a function, the function
   a. accesses exactly the same array as the calling program.
   b. accesses a copy of the array passed by the program.
   c. refers to the array using the same name as that used by the calling program.
   d. refers to the array using a different name than that used by the calling program.

14. Tell what this statement defines:
    employee emplist[1000];
15. Write an expression that accesses a structure member called salary in a structure variable that is the 17th element in an array called emplist.

16. In a stack, the data item placed on the stack first is
   a. not given an index number.
   b. given the index number 0.
   c. the first data item to be removed.
   d. the last data item to be removed.

17. Write a statement that defines an array called manybirds that holds 50 objects of type bird.

18. True or false: The compiler will complain if you try to access array element 14 in a 10-element array.

19. Write a statement that executes the member function cheep() in an object of class bird that is the 27th element in the array manybirds.
20. A string in C++ is an ________ of type ________.

21. Write a statement that defines a string variable called city that can hold a string of up to 20 characters (this is slightly tricky).

22. Write a statement that defines a string constant, called dextrose, that has the value “C6H12O6-H2O”.

23. True or false: The extraction operator (>>) stops reading a string when it encounters a space.

24. You can read input that consists of multiple lines of text using
   a. the normal cout << combination.
   b. the cin.get() function with one argument.
   c. the cin.get() function with two arguments.
   d. the cin.get() function with three arguments.

25. Write a statement that uses a string library function to copy the string name to the string blank.

26. Write the declaration for a class called dog that contains two data members: a string called breed and an int called age. (Don’t include any member functions.)

27. True or false: You should prefer C-strings to the Standard C++ string class in new programs.

28. Objects of the string class
   a. are zero-terminated.
   b. can be copied with the assignment operator.
   c. do not require memory management.
   d. have no member functions.
17. bird manybirds[50];
18. false
19. manybirds[26].cheep();
20. array, char
21. char city[21] (An extra byte is needed for the null character.)
22. char dextrose[] = "C6H12O6-H2O";
23. true
24. d
25. strcpy(blank, name);
26. class dog
   {
      private:
         char breed[80];
         int age;
   };
27. false
28. b, c
29. int n = s1.find("cat");
30. s1.insert(12, "cat");
1. Write a function called reverseit() that reverses a C-string (an array of char). Use a for loop that swaps the first and last characters, then the second and next-to-last characters, and so on. The string should be passed to reverseit() as an argument.

Write a program to exercise reverseit(). The program should get a string from the user, call reverseit(), and print out the result. Use an input method that allows embedded blanks.

2. Create a class called employee that contains a name (an object of class string) and an employee number (type long). Include a member function called getdata() to get data from the user for insertion into the object, and another function called putdata() to display the data. Assume the name has no embedded blanks.

Write a main() program to exercise this class. It should create an array of type employee, and then invite the user to input data for up to 100 employees. Finally, it should print out the data for all the employees.

3. Write a program that calculates the average of up to 100 English distances input by the user. Create an array of objects of the Distance class, as in the ENGLARAY example in this chapter. To calculate the average, you can borrow the add_dist() member function from the ENGLCON example in Chapter 6. You’ll also need a member function that divides a Distance value by an integer. Here’s one possibility:

```c++
void Distance::div_dist(Distance d2, int divisor)
{
    float fltfeet = d2.feet + d2.inches/12.0;
    fltfeet /= divisor;
    feet = int(fltfeet);
    inches = (fltfeet-feet) * 12.0;
}
```
```cpp
// ex7_1.cpp
// reverses a C-string
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    void reversit( char[] ); //prototype
    const int MAX = 80; //array size
    char str[MAX]; //string

    cout << "\nEnter a string: "; //get string from user
    cin.get(str, MAX);

    reversit(str); //reverse the string

    cout << "Reversed string is: "; //display it
    cout << str << endl;
    return 0;
}

//reversit()
//function to reverse a string passed to it as an argument
void reversit( char s[] )
{
    int len = strlen(s); //find length of string
    for(int j = 0; j < len/2; j++) //swap each character
    { // in first half
        char temp = s[j]; // with character
        s[j] = s[len-j-1]; // in second half
        s[len-j-1] = temp;
    }
}```
// ex7_2.cpp
// employee object uses a string as data
#include <iostream>
#include <string>
using namespace std;

///employee:

class employee
{
private:
    string name;
    long number;

public:
    void getdata()     //get data from user
    {
        cout << "\nEnter name: "; cin >> name;
        cout << "Enter number: "; cin >> number;
    }

    void putdata()     //display data
    {
        cout << "\n Name: " << name;
        cout << "\n Number: " << number;
    }
};
int main()
{
    employee emparr[100];  // an array of employees
    int n = 0;             // how many employees
    char ch;               // user response

    do {                   // get data from user
        cout << "Enter data for employee number " << n+1;
        emparr[n++].getdata();
        cout << "Enter another (y/n)? "; cin >> ch;
    } while( ch != 'n' );

    for(int j=0; j<n; j++)    // display data in array
    {
        cout << "Employee number " << j+1;
        emparr[j].putdata();
    }

    cout << endl;
    return 0;
}
class Distance
{
  private:
    int feet;
    float inches;

  public:
    Distance() //constructor (no args)
    { feet = 0; inches = 0; }
    Distance(int ft, float in) //constructor (two args)
    { feet = ft; inches = in; }

    void getdist() //get length from user
    {
      cout << "\nEnter feet: "; cin >> feet;
      cout << "Enter inches: "; cin >> inches;
    }

    void showdist() //display distance
    { cout << feet << "\'-" << inches << '"'; }

    void add_dist( Distance, Distance ); //declarations
    void div_dist( Distance, int );
};
void Distance::add_dist(Distance d2, Distance d3)
{
    inches = d2.inches + d3.inches;   // add the inches
    feet = 0;                        // (for possible carry)
    if(inches >= 12.0)               // if total exceeds 12.0,
    {
        inches -= 12.0;               // then decrease inches
        feet++;                       // by 12.0 and
    }                                 // increase feet
    feet += d2.feet + d3.feet;       // add the feet
}
void Distance::div_dist(Distance d2, int divisor)
{
    float fltfeet = d2.feet + d2.inches/12.0;  // convert to float
    fltfeet /= divisor;  // do division
    feet = int(fltfeet);  // get feet part
    inches = (fltfeet-feet) * 12.0;  // get inches part
}

int main()
{
    Distance distarr[100];  // array of 100 Distances
    Distance total(0, 0.0), average;  // other Distances
    int count = 0;  // counts Distances input
    char ch;  // user response character
    do {
        cout << "Enter a Distance";  // get Distances
        distarr[count++].getdist();  // from user, put
        cout << "Do another (y/n)? ";  // in array
        cin >> ch;
    }while( ch != 'n' );

    for(int j=0; j<count; j++)  // add all Distances
        total.add_dist( total, distarr[j] );  // to total
    average.div_dist( total, count );  // divide by number
    cout << "The average is: ";  // display average
    average.showdist();
    cout << endl;
    return 0;
}