# Introduction to C++ Programming

## Its Applications in Finance

Thanh Hoang

Claremont Graduate University

September 29, 2012

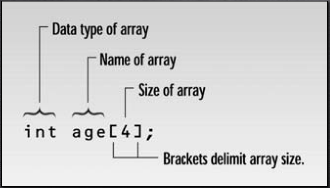
# Today Agenda

1. Arrays
   - One-Dimensional (1D) Array
   - Two-Dimensional (2D) Array
   - Multi-Dimensional Array
2. Strings
   - String Functions
   - Array of Strings
3. Pointers
   - Pointer Operators
   - Pointer Expression
   - Pointers and Arrays
4. File Input and Output
5. Summary

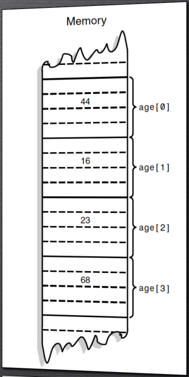
# One-Dimensional Array

## Definition

A one-dimensional array is a list of related variables.



```
int age[4];
```



# Display An Array

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

int main()
{
    int t, data[5];

    // Inputs values in an array
    for(t=0; t<=4; t++) {
        cout << "Enter a number [" << t
                                  << "] in an array: ";
        cin >> data[t];
    }
    cout << endl;

    // Displays the array
    for(t=0; t<=4; t++)
        cout << "Data [" << t << "] in the Data array: "
        << data[t] << endl;

    cout << endl;

    return 0;
}
```

Output

```
Enter a number [0] in an array: 5
Enter a number [1] in an array: 6
Enter a number [2] in an array: 7
Enter a number [3] in an array: 8
Enter a number [4] in an array: 9

Data [0] in the Data array: 5
Data [1] in the Data array: 6
Data [2] in the Data array: 7
Data [3] in the Data array: 8
Data [4] in the Data array: 9
```

# Array of Integers

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

int main()
{
    int yams[3]; // creates array with three elements
    yams[0] = 7; // assigns value to first element
    yams[1] = 8;
    yams[2] = 6;

    int yamcosts[3] = {20, 30, 5}; // initializes an integer array

    cout << "Total yams = " << yams[0] + yams[1] + yams[2] << endl;

    cout << "The package with " << yams[1] << " yams costs "
            << yamcosts[1] << " cents per yam.\n";

    int total = yams[0] * yamcosts[0] + yams[1] * yamcosts[1]
                  + yams[2] * yamcosts[2];
    cout << "The total yam expense is " << total << " cents.\n";

    cout << "\nSize of yams array = " << sizeof(yams) << " bytes.\n";
    cout << "Size of one element = " << sizeof(yams[0]) << " bytes.\n";

    return 0;
}
```

Output

```
Total yams = 21
The package with 8 yams
  costs 30 cents per yam.
The total yam expense is 410 cents.

Size of yams array = 12 bytes.
Size of one element = 4 bytes.
```

# Compute the Average, Minimum and Maximum

## Question

Write a C++ program to compute the average and find the minimum and maximum of a series of six values representing sales for each day of the week, excluding Sunday.

```
int size = 6;
for(t=0; t < size; t++) {
    switch(t) {
        case 0: cout << "Enter the sales on Monday: ";
                break;
        case 1: cout << "Enter the sales on Tuesday: ";
                break;
        case 2: cout << "Enter the sales on Wednesday: ";
                break;
        case 3: cout << "Enter the sales on Thursday: ";
                break;
        case 4: cout << "Enter the sales on Friday: ";
                break;
        case 5: cout << "Enter the sales on Saturday: ";
                break;
    }
    cin >> data[t];
    avg += data[t] / size;
}
```

```
// Finds the minimum and maximum values
min_value = max_value = data[0];
for (t=0; t < size; t++) {
    if (data[t] < min_value)
            min_value = data[t];

    if (data[t] > max_value)
            max_value = data[t];
}
```

# How to Transfer Data

## How to transfer the contents of one array into another?

Let's see this way. Does it work?

```
int a[5], b[5];
a = b;
```

How about this method?

```
int t, a[5], b[5];
for (t = 0; t < 5; t + +)
    a[t] = b[t];
```

# Initialization Rules for Arrays

## Some C++ Rules for Arrays

Let's see this way. Does it work?

```
data[4] = {5, 6, 7, 8};
data1 = data2;
data[4] = {1, 2, 3, 4, 5};
data[5] = {1; 2; 3; 4; 5};
data[ ] = {1.2, 3.4, 5.6, 7.8, 9.1};
```

How do we correct these mistakes?

```
int data[4] = {1, 2, 3, 4};
int data[5] = {1, 2, 3, 4, 5};
float data[ ] = {1.2, 3.4, 5.6, 7.8, 9.1};
```

# C++ Rules for Arrays

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

int main ()
{
    float data[] = {1.1,2.2,3.3,4.4,5.5};

    for (int i=0; i<=4; i++)
        cout << data[i] << endl;

    return 0;
}
```

Output 1

```
1.1
2.2
3.3
4.4
5.5
```

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

int main ()
{
    float data[] = {1.1,2.2,3.3,4.4,5.5};

    for (int i=0; i<=6; i++)
        cout << data[i] << endl;

    return 0;
}
```

Output 2

```
1.1
2.2
3.3
4.4
5.5
4.59163e-41
2.76793e+19
```

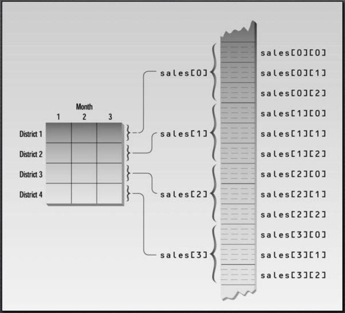
# Two-Dimensional Arrays

## Definition

A two-dimensional (2D) array is a list of one-dimensional arrays.

## General Form

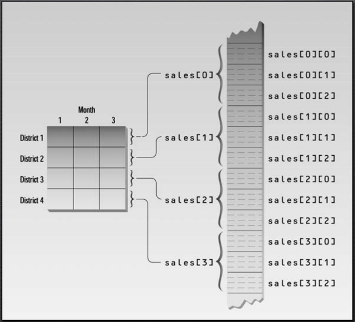
type data[*a*][*b*];



# Two-Dimensional Arrays

## A Simple Program

Write a C++ program to use a 2D array in order to store sales figures for several districts and in several months, which the user inputs.



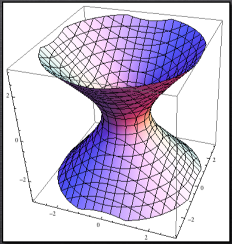
# Multi-Dimensional Arrays

## General Form

```
type data[a][b][c];
```

E.g. *char* threeDims[7][7][7] would require $7 * 7 * 7$, equaled to 343 bytes. If each array dimension is increased by 10 times, then the required memory increases to 343, 000 bytes!

Figure : $f(x, y, z) = x^2 + y^2 - z^2, \ x, y, z \in [-3, 3]$



# String Fundamentals

## Definition

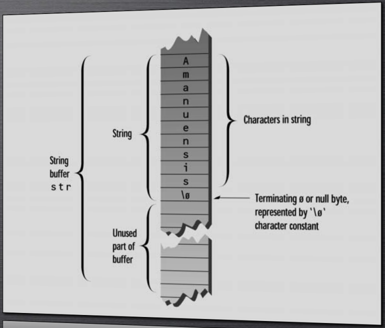
The most common use for one- dimensional arrays is to create a character string.

## General Form

```
char mystring[size];
```

## Two Types of Strings

1. C-style string (*null-terminated*)
2. The string class



# Display a String

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

int main()
{
    const int size = 80; // max characters in string
    char str[size]; // declares a string variable str

    cout << "Enter a string: ";
    cin >> str; // inputs string in str

    // Displays string from str
    cout << "You entered: " << str << endl;

    return 0;
}
```

## Output

```
Enter a string: Hello World!
You entered: Hello
```

# cin.getline()

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

int main()
{
    const int size = 80; // max characters in string
    char str[size]; // declares a string variable str

    cout << "Enter a string: ";
    cin.getline(str,80); // inputs string in str

    // Displays string from str
    cout << "You entered: " << str << endl;

    return 0;
}
```

**Output**

```
Enter a string: Hello World!
You entered: Hello World!
```

# cin.getline()

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

int main()
{
    char name[20], dessert[20];

    cout << "Enter a name: ";
    cin.getline(name, 20);

    cout << "Enter his/her favorite dessert: ";
    cin.getline(dessert, 20);

    cout << "I have some delicious " << dessert
              << " for " << name << endl;

    return 0;
}
```

# String Functions

### *strcpy*

The *strcpy* (str1, str2) function copies the contents from a string to another.

*strcpy* (to, from);

** The size of the destination string must be large enough to hold the original string.*

### *strcat*

The *strcat* (str1, str2) function appends *str2* to the end of *str1*; while *str2* is unchanged.

*strcat* (str1, str2);

*** The size of the str1 string must be large enough to hold its original string and the str2.*

### *strcmp*

The *strcmp* (str1, str2) function compares two strings and returns zero if they are the same. If *str1* is greater than *str2* according to dictionary order, a positive value will be returned; otherwise, a negative value will be returned.

*strcmp* (str1, str2);

**** strcmp function returns false when the strings match.*

### *strlen*

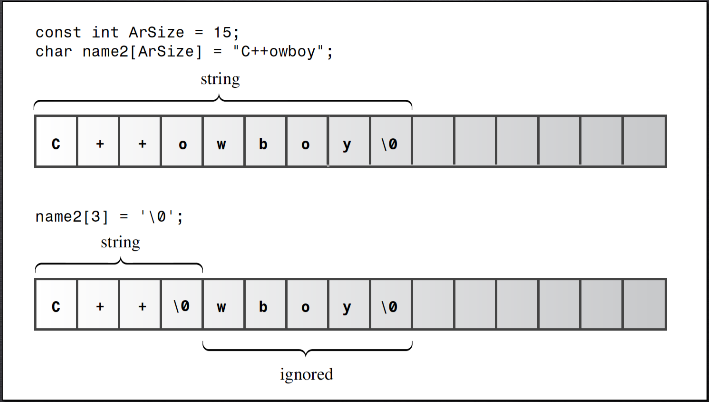
*strlen* function returns the length of the string.

*strlen* (str);

# Another Example of String

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

int main()
{
    const int Size = 15;
    char name1[Size]; // desclares an empty array
    char name2[Size] = "C++owboy"; // initializes an array

    // NOTE: some implementations may require the static keyword
    // to initialize the array name2
    cout << "Howdy! I m " << name2;
    cout << "! What s your first name? ";
    cin >> name1;

    cout << "Well, " << name1 << ", your name has ";
    cout << strlen(name1) << " letters and is stored" << endl;
    cout << "in an array of " << sizeof(name1) << " bytes." << endl;
    cout << "Your initial is " << name1[0] << "." << endl;

    name2[3] = \0 ; // null character

    cout << "Here are the first 3 characters of my name: ";
    cout << name2 << endl;

    return 0;
}
```

# Shortening a String with a Null Terminator



```
const int ArSize = 15;
char name2[ArSize] = "C++owboy";
```

```
name2[3] = '\0';
```

# Using the Null Terminator

```cpp
#include <iostream>
#include <cstring>
#include <cctype>
using namespace std;

int main()
{
    char str[80];

    strcpy(str, "my name is cplusplus");

    for (int i=0; str[i]; i++)
        str[i] = toupper(str[i]);

    cout << str << endl;

    return 0;
}
```

Output

```
MY NAME IS CPLUSPLUS
```

# Arrays of Strings

## Definition

An array of strings is a special form of a two-dimensional array. In order to create an array of strings, a two-dimensional character array is used.

## General Form

char data[*number of strings*][*maximum length of each string*];

E.g. *char* data[20][80] declares an array of 20 strings, in which each string has a maximum length of 80, including 79 characters and 1 null terminator.

# Telephone directory

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

int main()
{
    int t;
    char str[80];
    char numbers[10][80] = {
        "Adam", "(909) 678-3322",
        "Mary", "(909) 654-3452",
        "Jone", "(909) 453-3421",
        "Rachel", "(909) 453-8970",
        "Tom", "(909) 987-8787"
    };

    cout << "Enter a name: ";
    cin >> str;

    for(t=0; t < 10; t += 2)
        if(!strcmp(str, numbers[t])) {
            cout << "Phone number is " << numbers[t+1] << endl;
            break;
        }

    if(t == 10) cout << "Not found." << endl;

    return 0;
}
```

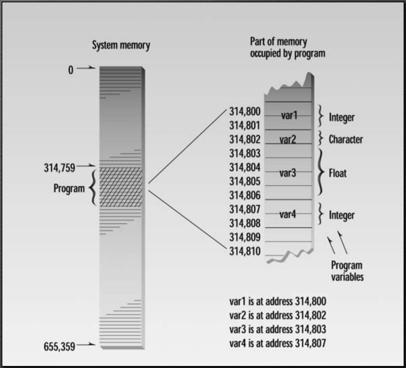
# What are pointers?

**Definition**

A pointer is an object that contains a memory address.

**General Form**

```
type *var;
```



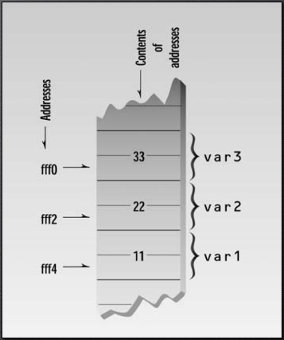
var1 is at address 314,800
var2 is at address 314,802
var3 is at address 314,803
var4 is at address 314,807

# The Address-of Operator &

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

int main()
{
    // Declares and initializes 3 variables
    int var1 = 11;
    int var2 = 22;
    int var3 = 33;

    // Prints the addresses of these variables
    cout << &var1 << endl
         << &var2 << endl
         << &var3 << endl;

    return 0;
}
```

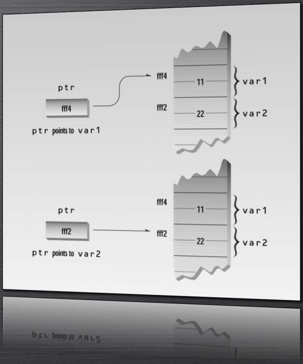
Output

```
0x7fff5fbff798
0x7fff5fbff794
0x7fff5fbff790
```



# Pointer Variables

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

int main()
{
    int var1 = 11; // two integer variables
    int var2 = 22;

    cout << &var1 << endl; // prints addresses
    cout << &var2 << endl;

    cout << endl;

    int *ptr; // pointer to integers
    ptr = &var1; // pointer points to var1
    cout << ptr << endl; // prints pointer value

    ptr = &var2; // pointer points to var2
    cout << ptr << endl; // prints pointer value

    return 0;
}
```



# Pointer Operators

## Two Special Operators

1. The & operator returns the memory address. For example,

   ptr = &var1; // the address of the *var1* variable

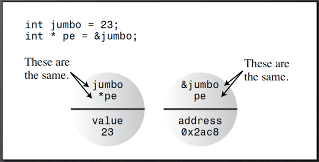
2. The * operator is the complement of &. For example,

   value = *ptr; // assign the value of *var1* into value

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

int main()
{
    int var1, *ptr, value;

    var1 = 999; // assigns 999 to var1
    cout << var1 << endl;

    ptr = &var1; // gets the address of var1
    cout << ptr << endl;

    value = *ptr; // gets value at that address
    cout << value << endl;

    return 0;
}
```

# Initialize a Pointer

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

int main()
{
    int var = 8; // declares a variable
    int *p_var; // declares pointer to an int

    p_var = &var; // assigns address of int to pointer

    // Displays values in two ways
    cout << "Values: var = " << var;
    cout << ", *p_var = " << *p_var << endl;

    // Displays address in two ways
    cout << "Addresses: &var = " << &var;
    cout << ", p_var = " << p_var << endl;

    // Uses pointer to change value
    (*p_var)++;
    cout << "Now var = " << var << endl;

    return 0;
}
```



```
int jumbo = 23;
int * pe = &jumbo;
```

# Assign Values via a Pointer

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

int main()
{
    int num, *ptr;

    ptr = &num; // assigns the address of num
    *ptr = 99; // assigns 99 to num via ptr
    cout << num << endl;

    (*ptr)++; // adds 1 to num via ptr
    cout << num << endl;

    (*ptr)--; // subtracts 1 from num via ptr
    cout << num << endl;

    return 0;
}
```

Output

```
99
100
99
```

# Base Type of a Pointer

Q: How does the compiler transfer the proper number of bytes for any assignment involving a pointer?

A: The base type of the pointer will determine what data type the pointer operates.

```
// Do you think this statement is correct?
int *p;
double f;
......
p = &f;
......
```

```
// How about this statement?
int *p;
double f;
......
p = (int *) &f;
......
```

# Base Type of a Pointer

Q: How does the compiler transfer the proper number of bytes for any assignment involving a pointer?

A: The base type of the pointer will determine what data type the pointer operates.

```
// Do you think this statement is correct?
int *p;
double f;
......
p = &f;
......
```

```
// How about this statement?
int *p;
double f;
......
p = (int *) &f;
......
```

# Arrays of Pointers

**Definition**

Like other data type, pointers can be arrayed.

type *ptrStr[10];

Q: How do we assign the address of an integer variable to the first element of the pointer array?

A: Similar to a normal array, we will write:

```
int a;
int *ptrStr[10];
ptrStr[0] = &a;
```

Since *ptrStr*[0] is an integer pointer, to find the value of *a*, we should write:

```
*ptrStr[0]
```

# Reversed String

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

int main()
{
   char *start, *end; // pointers used to access the string
   unsigned long length;
   char swap_char, str[80];

   cout << "Enter your string: ";
   cin.getline(str, 80); // obtains the string from the user

   cout << "\nThe original string is: " << str << endl;
   length = strlen(str);
   start = &str[0]; // points to the first character
   end = &str[length-1]; // points to the character before the null terminator

   while (start < end) {
      swap_char = *start;
      *start = *end;
      *end = swap_char;

      start++; // moves forward
      end--;
   }

   cout << "The reversed string is: " << str << endl;

   return 0;
}
```

# File Input and Output

## *fstream* Library

This library provides an interface to read and write data from files.

## File Input and Output

1. Reads data from a file

   ```
   ifstream inputName("inputFile.txt");
   ```

2. Writes data to a file

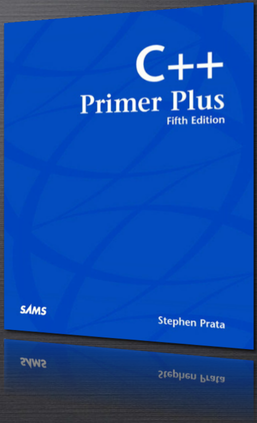
   ```
   ofstream outputName("outputFile.txt");
   ```

E.g: Let's input a matrix A and then output a matrix B

$$A = \begin{pmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{pmatrix} \longrightarrow B = 2A$$

# Summary

1. Arrays
   - One-dimensional Arrays
   - Two-dimensional Arrays
   - Multi-dimensional Arrays
2. Strings
   - Some *string* Functions
   - Arrays of Strings
3. Pointers
   - Pointer Operators
   - Pointer Expression
   - Pointers and Arrays
4. File Input and Output



**Reading**

Stephen Prata.

*C++ Primer Plus, 5th Edition,*

Chapter 4

SAMS Publishing, 2004.

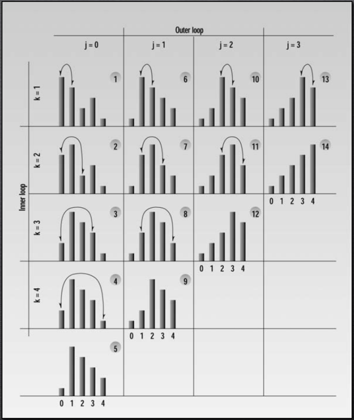
# Bubble sort

Write a C++ to sort an array by using the simplest and easiest to understand algorithm, bubble sort. Basically, the bubble sort uses repeated comparison, in which small values move toward one end, and large ones move toward the other end.

*Hint:* You should use two *for* loops:

1. The inner *for* loop checks and look for out-of-order elements, when an out-of-order element pair is found, the two elements should be exchanged.
2. The outer *for* loop makes the program to repeat until all elements are sorted.

# Bubble sort


Outer loop
j = 0
j = 1
j = 2
j = 3
Inner loop
k = 1
k = 2
k = 3
k = 4
1
2
3
4
5
6
7
8
9
10
11
12
13
14
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4

# Bond pricing

Write a C++ to price a 10-year bond with a 10% coupon rate and a par or maturity value of $1000, and a required yield of 12%.

Assume that coupon payments are made semi-annually to bond holders.

1. Determine the number of coupon payments
2. Determine the value of each coupon payment
3. Determine the semi-annual yield

The formula of the current bond price with discrete compounding is:

$$P_0 = \sum_{t=1}^{T} \frac{C}{(1+r)^t} + \frac{M}{(1+r)^T}$$

# Cholesky Decomposition Algorithm

**Statement**

If $A$ is a positive definite, symmetric matrix of real entries, then we can decompose $A$.

By decomposing, we can write $A$ as

$$A = LL^T \tag{1}$$

*where:*

$L$ : a lower triangular matrix with strictly positive diagonal entries

$L^T$ : the conjugate transpose of L

# General Algorithm for $A_{nxn}$

```
for (int j=0; j<n; j++) {
    1. for (int k=0; k<j; k++):
            $\sum_{k=0}^{j-1} L_{j,k}^2$
    2. Computes $L_{j,j}$
            $L_{j,j} = \sqrt{A_{j,j} - \sum_{k=0}^{j-1} L_{j,k}^2}$
    3. for (int i=j+1; i<n; i++) {
        3.1. for (int k=0; k<j; k++):
                $\sum_{k=0}^{j-1} L_{i,k} L_{j,k}$
        3.2. Computes $L_{i,j}$
                $L_{i,j} = \frac{1}{L_{j,j}} \left( A_{i,j} - \sum_{k=0}^{j-1} L_{i,k} L_{j,k} \right)$
    }
}
```