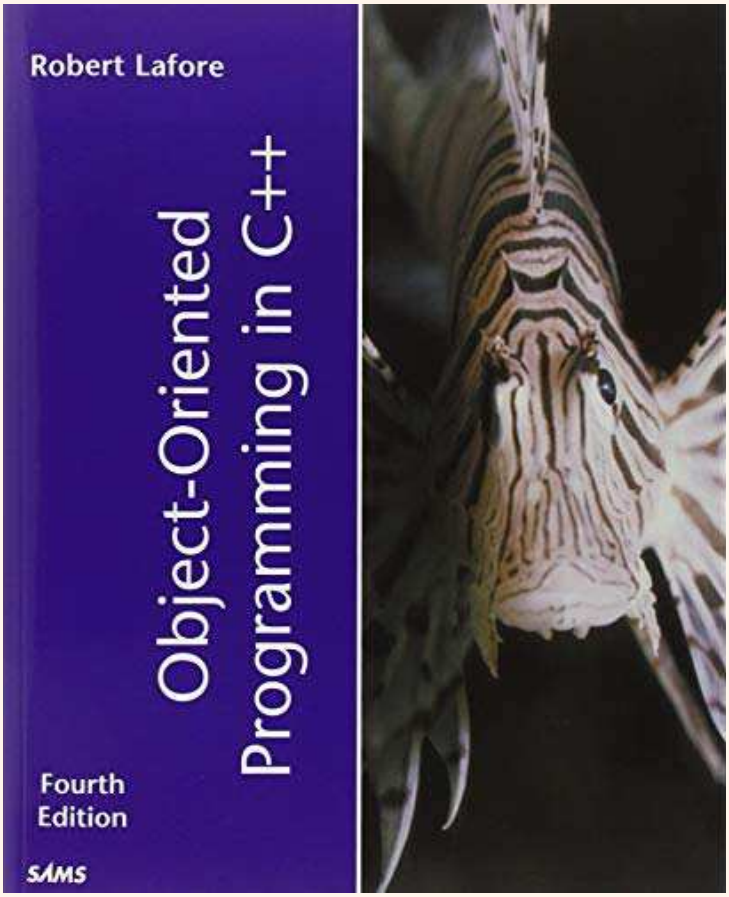


Chapter 7

Arrays and Strings

Animated Version

Topics

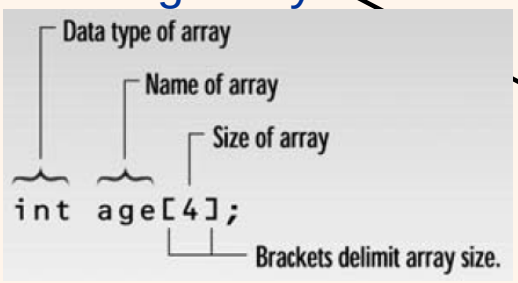
- Array Fundamentals
- Arrays as Class Member Data
- Arrays of Objects
- C-Strings
- C++ Objects as Data Types
- The Standard C++ string Class

Introduction

- In C++ *array* can be used to group together data items of the same type.
  - can be simple types such as int or float , or they can be user-defined types such as structures and objects.
- A structure usually groups items of different types, an array groups items of the same type.
  - the items in a structure are accessed by name, while those in an array are accessed by an index number.

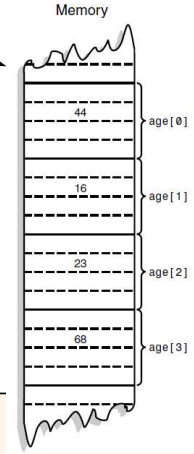
Array Fundamentals

- Defining Arrays



- Array Elements
- Accessing Array Elements

```
// replay.cpp
// gets four ages from user, displays them
#include <iostream>
using namespace std;

int main()
  {
  int age[4];                      //array 'age' of 4 ints

  for(int j=0; j<4; j++)           //get 4 ages
     {
     cout << "Enter an age: ";
     cin >> age[j];                //access array element
     }
  for(j=0; j<4; j++)               //display 4 ages
     cout << "You entered " << age[j] << endl;
  return 0;
  }
```




```
Output:
Enter an age: 44
Enter an age: 16
Enter an age: 23
Enter an age: 68

You entered 44
You entered 16
You entered 23
You entered 68
```

Array Fundamentals (2)

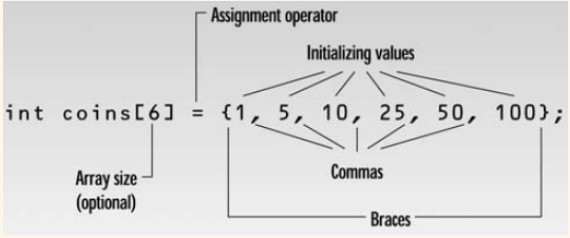
- Averaging Array Elements

```
// sales.cpp
// averages a weeks's widget sales (6 days)
#include <iostream>
using namespace std;

int main()
   {
   const int SIZE = 6;              //size of array
   double sales[SIZE];              //array of 6 variables

   cout << "Enter widget sales for 6 days\n";
   for(int j=0; j<SIZE; j++)        //put figures in array
      cin >> sales[j];

   double total = 0;
   for(j=0; j<SIZE; j++)         //read figures from array
      total += sales[j];            //to find total
   double average = total / SIZE;  // find average
   cout << "Average = " << average << endl;
   return 0;
   }
```

```
Output:
Enter widget sales
for 6 days
352.64
867.70
781.32
867.35
746.21
189.45
Average = 634.11
```


Array Fundamentals (3)

- Initializing Arrays

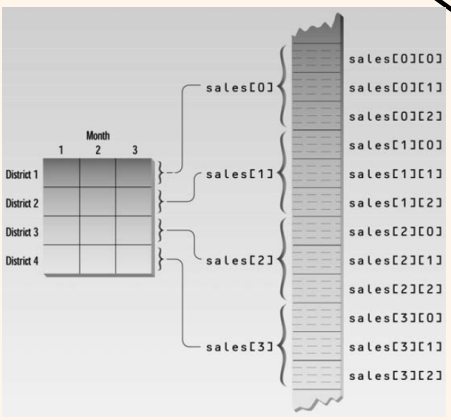



```
// days.cpp
// shows days from start of year to date specified
#include <iostream>
using namespace std;

int main()
   {
   int month, day, total_days;
   int days_per_month[12] = { 31, 28, 31, 30, 31, 30,
                              31, 31, 30, 31, 30, 31 };

   cout << "\nEnter month (1 to 12): ";  //get date
   cin >> month;
   cout << "Enter day (1 to 31): ";
   cin >> day;
   total_days = day;                        //separate days
   for(int j=0; j<month-1; j++)             //add days each
    month
      total_days += days_per_month[j];
   cout << "Total days from start of year is: " <<
    total_days
        << endl;
   return 0;
   }
```

```
Output:
Enter month (1 to 12): 3
Enter day (1 to 31): 11
Total days from start of
year is: 70
```

Array Fundamentals (4)

- Multidimensional Arrays
  - Defining Multidimensional Arrays
  - Accessing Multidimensional Array Elements



  - Formatting Numbers

```
// salemon.cpp
// displays sales chart using 2-d arr
#include <iostream>
#include <iomanip>                    /
using namespace std;
const int DISTRICTS = 4;              /
const int MONTHS = 3;
int main()
   {
   int d, m;
   double sales[DISTRICTS][MONTHS];

   cout << endl;
   for(d=0; d<DISTRICTS; d++)
      for(m=0; m<MONTHS; m++)
         {
         cout << "Enter sales for district " << d+1;
         cout << ", month " << m+1 << ": ";
         cin >> sales[d][m];          //put number in array
         }

   cout << "\n\n";
   cout << "                         Month\n";
   cout << "                1         2         3";
   for(d=0; d<DISTRICTS; d++)
      {
      cout <<"\nDistrict " << d+1;
      for(m=0; m<MONTHS; m++)         //display array values
         cout << setiosflags(ios::fixed)      //not exponential
              << setiosflags(ios::showpoint)  //always use point
              << setprecision(2)              //digits to right
              << setw(10)                     //field width
              << sales[d][m];         //get number from array
      }  //end for(d)
   cout << endl;
   return 0;
   }  //end main
```

```
Output:
Enter sales for district 1, month 1: 3964.23
Enter sales for district 1, month 2: 4135.87
Enter sales for district 1, month 3: 4397.98
Enter sales for district 2, month 1: 867.75
Enter sales for district 2, month 2: 923.59
Enter sales for district 2, month 3: 1037.01
Enter sales for district 3, month 1: 12.77
Enter sales for district 3, month 2: 378.32
Enter sales for district 3, month 3: 798.22
Enter sales for district 4, month 1: 2983.53
Enter sales for district 4, month 2: 3983.73
Enter sales for district 4, month 3: 9494.98

                         Month
                1         2         3
District 1   3964.23   4135.87   4397.98
District 2    867.75    923.59   1037.01
District 3     12.77    378.32    798.22
District 4   2983.53   3983.73   9494.98
```


Array Fundamentals (5)

- Initializing Multidimensional Arrays

```
// saleinit.cpp
// displays sales chart, initializes 2-d array
#include <iostream>
#include <iomanip>                  //for setprecision, etc.
using namespace std;
const int DISTRICTS = 4;            //array dimensions
const int MONTHS = 3;

int main()
   {
   int d, m;
                                    //initialize array elements
   double sales[DISTRICTS][MONTHS]
          = {  {  1432.07,    234.50,    654.01 },
               {   322.00, 13838.32, 17589.88 },
               {  9328.34,    934.00,  4492.30 },
               { 12838.29,   2332.63,    32.93 }  };
   cout << "\n\n";
   cout << "                         Month\n";
   cout << "                1         2         3";
   for(d=0; d<DISTRICTS; d++)
      {
      cout <<"\nDistrict " << d+1;
      for(m=0; m<MONTHS; m++)
         cout << setw(10) << setiosflags(ios::fixed)
              << setiosflags(ios::showpoint) <<
    setprecision(2)
              << sales[d][m];  //access array element
      }
   cout << endl;
   return 0;
   }
```

Array Fundamentals (6)

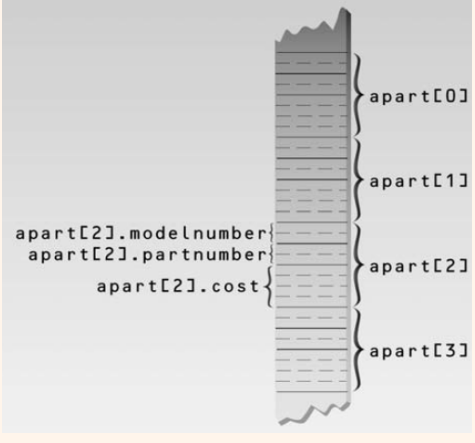
- Passing Arrays to Functions
  - –Function Declaration with Array Arguments
  - –Function Call with Array Arguments
  - –Function Definition with Array Arguments

```
// salefunc.cpp
// passes array as argument
#include <iostream>
#include <iomanip>          //for setprecision, etc.
using namespace std;
const int DISTRICTS = 4;    //array dimensions
const int MONTHS = 3;
void display( double[DISTRICTS][MONTHS] );  //declaration
//-------------------------------------------------------------
int main()
   {                        //initialize two-dimensional array
   double sales[DISTRICTS][MONTHS]
            = {  {  1432.07,   234.50,   654.01 },
                 {   322.00, 13838.32, 17589.88 },
                 {  9328.34,   934.00,  4492.30 },
                 { 12838.29,  2332.63,    32.93 }  };

   display(sales);          //call function; array as argument
   cout << endl;
   return 0;
   }  //end main
//-------------------------------------------------------------
//display()
//function to display 2-d array passed as argument
void display( double funsales[DISTRICTS][MONTHS] )
   {
   int d, m;

   cout << "\n\n";
   cout << "                        Month\n";
   cout << "                1         2         3";

   for(d=0; d<DISTRICTS; d++)
      {
      cout <<"\nDistrict " << d+1;
      for(m=0; m<MONTHS; m++)
         cout << setiosflags(ios::fixed) << setw(10)
              << setiosflags(ios::showpoint) << setprecision(2)
              << funsales[d][m];   //array element
      }  //end for(d)
   }  //end display
```

Array Fundamentals (7)

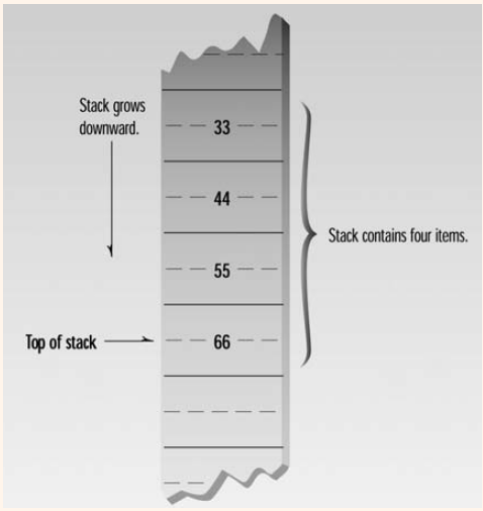
- Arrays of Structures

```
Output:
Enter model number: 44
Enter part number: 4954
Enter cost: 133.45

Enter model number: 44
Enter part number: 8431
Enter cost: 97.59

Enter model number: 77
Enter part number: 9343
Enter cost: 109.99

Enter model number: 77
Enter part number: 4297
Enter cost: 3456.55

Model 44  Part 4954  Cost 133.45
Model 44  Part 8431  Cost 97.59
Model 77  Part 9343  Cost 109.99
Model 77  Part 4297  Cost 3456.55
```




```
// partaray.cpp
// structure variables as array elements
#include <iostream>
using namespace std;
const int SIZE = 4;              //number of parts in array
////////////////////////////////////////////////////////////////
struct part                      //specify a structure
   {
   int modelnumber;              //ID number of widget
   int partnumber;               //ID number of widget part
   float cost;                   //cost of part
   };
////////////////////////////////////////////////////////////////
int main()
   {
   int n;
   part apart[SIZE];             //define array of structures

   for(n=0; n<SIZE; n++)         //get values for all members
      {
      cout << endl;
      cout << "Enter model number: ";
      cin >> apart[n].modelnumber;      //get model number
      cout << "Enter part number: ";
      cin >> apart[n].partnumber;       //get part number
      cout << "Enter cost: ";
      cin >> apart[n].cost;             //get cost
      }
   cout << endl;
   for(n=0; n<SIZE; n++)         //show values for all members
      {
      cout << "Model " << apart[n].modelnumber;
      cout << "  Part "  << apart[n].partnumber;
      cout << "  Cost "  << apart[n].cost << endl;
      }
   return 0;
   }
```

Arrays as Class Member Data

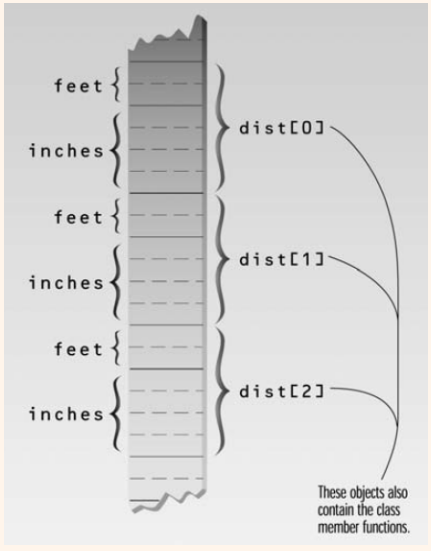



```
// stakaray.cpp
// a stack as a class
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////
class Stack
   {
   private:
      enum { MAX = 10 };            //(non-standard syntax)
      int st[MAX];                  //stack: array of integers
      int top;                      //number of top of stack
   public:
      Stack()                       //constructor
         { top = 0; }
      void push(int var)            //put number o
         { st[++top] = var; }
      int pop()                     //take number
         { return st[top--]; }
   };
////////////////////////////////////////////////////////////
int main()
   {
   Stack s1;

   s1.push(11);
   s1.push(22);
   cout << "1: " << s1.pop() << endl;  //22
   cout << "2: " << s1.pop() << endl;  //11
   s1.push(33);
   s1.push(44);
   s1.push(55);
   s1.push(66);
   cout << "3: " << s1.pop() << endl;  //66
   cout << "4: " << s1.pop() << endl;  //55
   cout << "5: " << s1.pop() << endl;  //44
   cout << "6: " << s1.pop() << endl;  //33
   return 0;
   }
```

Output:

```
1: 22
2: 11
3: 66
4: 55
5: 44
6: 33
```


Arrays of Objects




```
// englaray.cpp
// objects using English measurements
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////
class Distance                  //English Distance class
   {
   private:
      int feet;
      float inches;
   public:
      void getdist()            //get length from user
         {
         cout << "\n   Enter feet: ";  cin >> feet;
         cout << "   Enter inches: ";  cin >> inches; }
      void showdist() const     //display distance
         { cout << feet << "\'-" << inches << '\"'; }
   };
////////////////////////////////////////////////////////////
int main()
   {
   const int MAX = 100;
   Distance dist[MAX];          //array of distances
   int n=0;                     //count the entries
   char ans;                    //user response ('y' or 'n')
   cout << endl;

   do {                         //get distances from user
      cout << "Enter distance number " << n+1;
      dist[n++].getdist();      //store distance in array
      cout << "Enter another (y/n)?: ";
      cin >> ans;
      } while( ans != 'n' );    //quit if user types 'n'

   for(int j=0; j<n; j++)       //display all distances
      {
      cout << "\nDistance number " << j+1 << " is ";
      dist[j].showdist();       }
   cout << endl;
   return 0;
   }
```

Output:

```
Enter distance number 1
   Enter feet: 5
   Enter inches: 4
Enter another (y/n)? y
Enter distance number 2
   Enter feet: 6
   Enter inches: 2.5
Enter another (y/n)? y
Enter distance number 3
   Enter feet: 5
   Enter inches: 10.75
Enter another (y/n)? n

Distance number 1 is 5'-4"
Distance number 2 is 6'-2.5"
Distance number 3 is 5'-10.75"
```

Arrays of Objects (2)

- Arrays of Cards

```
// cardaray.cpp
// cards as objects
#include <iostream>
#include <cstdlib>          //for srand(), rand()
#include <ctime>            //for time for srand()
using namespace std;

enum Suit { clubs, diamonds, hearts, spades };
//from 2 to 10 are integers without names
const int jack = 11;
const int queen = 12;
const int king = 13;
const int ace = 14;
////////////////////////////////////////////////////////
class card
   {
   private:
      int number;         //2 to 10, jack, queen, king, ace
      Suit suit;          //clubs, diamonds, hearts, spades
   public:
      card()                      //constructor
         { }
      void set(int n, Suit s)     //set card
         { suit = s; number = n; }
      void display();             //display card
   };
```

```
//-----------------------------------------------------------
void card::display()              //display the card
   {
   if( number >= 2 && number <= 10 )
      cout << number;
   else
      switch(number)
         {
         case jack:  cout << "J"; break;
         case queen: cout << "Q"; break;
         case king:  cout << "K"; break;
         case ace:   cout << "A"; break;
         }
   switch(suit)
      {
      case clubs:    cout << static_cast<char>(5); break;
      case diamonds: cout << static_cast<char>(4); break;
      case hearts:   cout << static_cast<char>(3); break;
      case spades:   cout << static_cast<char>(6); break;
      }
   }
```

Graphics Characters

Arrays of Objects (3)

- Arrays of Cards (2)

```
////////////////////////////////////////////////////////
int main()
   {
   card deck[52];
   int j;

   cout << endl;
   for(j=0; j<52; j++)          //make an ordered deck
      {
      int num = (j % 13) + 2;   //cycles through 2 to 14, 4 times
      Suit su = Suit(j / 13);   //cycles through 0 to 3, 13 times
      deck[j].set(num, su);     //set card
      }
   cout << "\nOrdered deck:\n";
   for(j=0; j<52; j++)          //display ordered deck
      {
      deck[j].display();
      cout << "  ";
      if( !( (j+1) % 13) )      //newline every 13 cards
         cout << endl;
      }
   srand( time(NULL) );         //seed random numbers with time
   for(j=0; j<52; j++)          //for each card in the deck,
      {
      int k = rand() % 52;      //pick another card at random
      card temp = deck[j];      //and swap them
      deck[j] = deck[k];
      deck[k] = temp;
      }
   cout << "\nShuffled deck:\n";
   for(j=0; j<52; j++)          //display shuffled deck
      {
      deck[j].display();
      cout << ", ";
      if( !( (j+1) % 13) )      //newline every 13 cards
         cout << endl;
      }
   return 0;
   }  //end main
```

The Card Deck

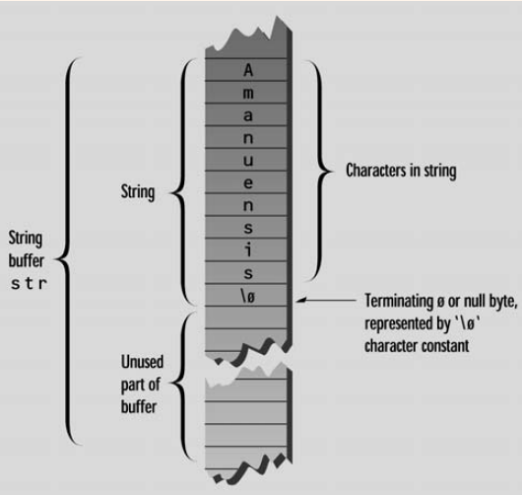
Random Numbers

Output:

```
Ordered deck:
2♢  3♢  4♢  5♢  6♢  7♢  8♢  9♢  10♢  J♢  Q♢  K♢  A♢
2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠  10♠  J♠  Q♠  K♠  A♠
2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥  10♥  J♥  Q♥  K♥  A♥
2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣  10♣  J♣  Q♣  K♣  A♣

Shuffled deck:
3♣  9♠  6♠  K♣  8♥  4♣  7♠  4♢  3♠  3♢  A♥  2♠  9♢
6♢  7♢  9♥  8♣  Q♢  Q♠  10♥  J♢  6♥  4♥  J♠  K♢  5♣
3♥  J♣  5♢  K♠  Q♥  10♠  8♠  2♢  6♣  A♣  4♠  J♥  8♢
10♣  2♥  Q♣  10♢  5♠  A♢  K♥  7♥  5♥  A♠  2♣  9♣  7♣
```

C-Strings

- C-strings and strings that are objects of the string class.
  - C-strings are arrays of type `char`
- C-String Variables




```
// stringin.cpp
// simple string variable
#include <iostream>
using namespace std;

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

   cout << "Enter a string: ";
   cin >> str;                   //put string in str
                                 //display string from
     str
   cout << "You entered: " << str << endl;
   return 0;
   }
```


C-Strings (2)

- Avoiding Buffer Overflow

```
// safetyin.cpp
// avoids buffer overflow with cin.width
#include <iostream>
#include <iomanip>              //for setw
using namespace std;

int main()
   {
   const int MAX = 20;      //max characters in string
   char str[MAX];              //string variable str

   cout << "\nEnter a string: ";
   cin >> setw(MAX) >> str;     //put string in str,
                           // no more than MAX chars
   cout << "You entered: " << str << endl;
   return 0;
   }
```

- String Constants

```
// strinit.cpp
// initialized string
#include <iostream>
using namespace std;

int main()
   {
   char str[] = "Farewell! thou art too dear for
    my possessing.";
   cout << str << endl;
   return 0;
   }
```

  - uses the setw manipulator to specify the maximum number of characters
    - one character fewer than the number specified is inserted, so there is room for the null character.

C-Strings (3)

- Reading Embedded Blanks

```
Enter a string: Law is a bottomless pit.
You entered: Law
```

```
// blanksin.cpp
// reads string with embedded blanks
#include <iostream>
using namespace std;

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

   cout << "\nEnter a string: ";
   cin.get(str, MAX);                //put string in str
   cout << "You entered: " << str << endl;
   return 0;
   }
```

```
Enter a string: Law is a bottomless pit.
You entered: Law is a bottomless pit.
```

- Reading Multiple Lines

```
// linesin.cpp
// reads multiple lines, terminates on '$'
    character
#include <iostream>
using namespace std;

const int MAX = 2000;  //max characters in string
char str[MAX];         //string variable str

int main()
   {
   cout << "\nEnter a string:\n";
   cin.get(str, MAX, '$');      //terminate with $
   cout << "You entered:\n" << str << endl;
   return 0;
   }
```

```
Enter a string:
Ask me no more where Jove bestows
When June is past, the fading rose;
For in your beauty's orient deep
These flowers, as in their causes, sleep.
$
You entered:
Ask me no more where Jove bestows
When June is past, the fading rose;
For in your beauty's orient deep
These flowers, as in their causes, sleep.
```


C-Strings (4)

- Copying a String the Hard Way

```
// strcopy1.cpp
// copies a string using a for loop
#include <iostream>
#include <cstring>                    //for strlen()
using namespace std;

int main()
   {                              //initialized string
   char str1[] = "Oh, Captain, my Captain! "
                 "our fearful trip is done";
   const int MAX = 80;          //size of str2 buffer
   char str2[MAX];              //empty string

   for(int j=0; j<strlen(str1); j++)//copy strlen
    characters
      str2[j] = str1[j];        //   from str1 to str2
   str2[j] = '\0';              //insert NULL at end
   cout << str2 << endl;        //display str2
   return 0;
   }
```

- Copying a String the Easy Way

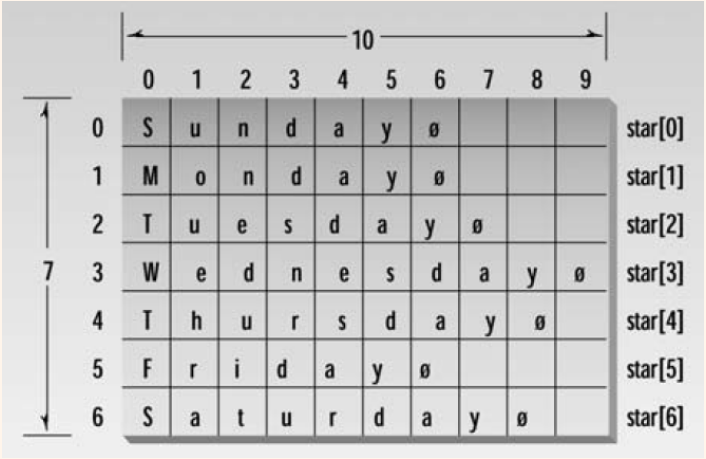
```
// strcopy2.cpp
// copies a string using strcpy() function
#include <iostream>
#include <cstring>                  //for strcpy()
using namespace std;

int main()
   {
   char str1[] = "Tiger, tiger, burning bright\n"
                 "In the forests of the night";
   const int MAX = 80;       //size of str2 buffer
   char str2[MAX];           //empty string

   strcpy(str2, str1);       //copy str1 to str2
   cout << str2 << endl;     //display str2
   return 0;
   }
```

C-Strings (5)

- Arrays of Strings

```
// straray.cpp
// array of strings
#include <iostream>
using namespace std;

int main()
   {
   const int DAYS = 7;  //number of strings in array
   const int MAX = 10;  //maximum size of each string
                        //array of strings
   char star[DAYS][MAX] = { "Sunday", "Monday",
               "Tuesday", "Wednesday", "Thursday",
               "Friday", "Saturday"  };

   for(int j=0; j<DAYS; j++)    //display every string
      cout << star[j] << endl;
   return 0;
   }
```

```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```



C-Strings (6)

- Strings as Class Members

```
// strpart.cpp
// string used in widget part object
#include <iostream>
#include <cstring>        //for strcpy()
using namespace std;
////////////////////////////////////////////////
class part
   {
   private:
      char partname[30];  //name of widget part
      int partnumber;     //ID number of widget part
      double cost;        //cost of part
   public:
      void setpart(char pname[], int pn, double c)
         {
         strcpy(partname, pname);
         partnumber = pn;
         cost = c;
         }
      void showpart()     //display data
         {
         cout << "\nName="   << partname;
         cout << ", number=" << partnumber;
         cout << ", cost=$"  << cost;
         }
   };
////////////////////////////////////////////////
int main()
   {
   part part1, part2;
   part1.setpart("handle bolt", 4473, 217.55);   //set parts
   part2.setpart("start lever", 9924, 419.25);
   cout << "\nFirst part: ";  part1.showpart();  //show parts
   cout << "\nSecond part: "; part2.showpart();
   cout << endl;
   return 0;
   }
```

```
First part:
Name=handle bolt, number=4473, cost=$217.55
Second part:
Name=start lever, number=9924, cost=$419.25
```

C-Strings (7)

- A User-Defined String Type

```
// strobj.cpp
// a string as a class
#include <iostream>
#include <cstring>       // for strcpy(), strcat()
using namespace std;
////////////////////////////////////////////////////////////
    ///
class String
   {
   private:
      enum { SZ = 80 };           //max size of Strings
      char str[SZ];               //array
   public:
      String()                    //constructor, no args
         { str[0] = '\0'; }
      String( char s[] )          //constructor, one arg
         { strcpy(str, s); }
      void display()              //display string
         { cout << str; }
      void concat(String s2)      //add arg string to
         {                        //this string
         if( strlen(str)+strlen(s2.str) < SZ )
            strcat(str, s2.str);
         else
            cout << "\nString too long";
         }
   };
```

```
////////////////////////////////////////////////////////////
int main()
   {
   String s1("Merry Christmas!  ");      //uses constructor 2
   String s2 = "Season's Greetings!";    //alternate form of
     2
   String s3;                            //uses constructor 1

   cout << "\ns1="; s1.display();         //display them all
   cout << "\ns2="; s2.display();
   cout << "\ns3="; s3.display();

   s3 = s1;                              //assignment
   cout << "\ns3="; s3.display();         //display s3

   s3.concat(s2);                        //concatenation
   cout << "\ns3="; s3.display();         //display s3
   cout << endl;
   return 0;
   }
```

```
s1=Merry Christmas!
s2=Season's Greetings!
s3=      <------- nothing here yet
s3=Merry Christmas!    <------- set equal to s1
s3=Merry Christmas! Season's Greetings!      <------- s2 concatenated
```


Array Fundamentals (6)

- Passing Arrays to Functions
  - –Function Declaration with Array Arguments
  - –Function Call with Array Arguments
  - –Function Definition with Array Arguments

```
// salefunc.cpp
// passes array as argument
#include <iostream>
#include <iomanip>          //for setprecision, etc.
using namespace std;
const int DISTRICTS = 4;    //array dimensions
const int MONTHS = 3;
void display( double[DISTRICTS][MONTHS] );  //declaration
//-------------------------------------------------------------
int main()
   {                        //initialize two-dimensional array
   double sales[DISTRICTS][MONTHS]
            = {  {  1432.07,   234.50,   654.01 },
                 {   322.00, 13838.32, 17589.88 },
                 {  9328.34,   934.00,  4492.30 },
                 { 12838.29,  2332.63,    32.93 }  };

   display(sales);          //call function; array as argument
   cout << endl;
   return 0;
   }  //end main
//-------------------------------------------------------------
//display()
//function to display 2-d array passed as argument
void display( double funsales[DISTRICTS][MONTHS] )
   {
   int d, m;

   cout << "\n\n";
   cout << "                        Month\n";
   cout << "                1         2         3";

   for(d=0; d<DISTRICTS; d++)
      {
      cout <<"\nDistrict " << d+1;
      for(m=0; m<MONTHS; m++)
         cout << setiosflags(ios::fixed) << setw(10)
              << setiosflags(ios::showpoint) << setprecision(2)
              << funsales[d][m];   //array element
      }  //end for(d)
   }  //end display
```

The Standard C++ string Class

- C++ includes a new class called `string`
  - no longer need to worry about creating an array of the right size to hold string variables
  - allows the use of overloaded operators, so you can concatenate string objects with the + operator:
    - `s3 = s1 + s2`

```
//sstrass.cpp
//defining and assigning string objects
#include <iostream>
#include <string>
using namespace std;

int main()
   {
   string s1("Man");                    //initialize
   string s2 = "Beast";                 //initialize
   string s3;

   s3 = s1;                             //assign
   cout << "s3 = " << s3 << endl;

   s3 = "Neither " + s1 + " nor ";      //concatenate
   s3 += s2;                            //concatenate
   cout << "s3 = " << s3 << endl;

   s1.swap(s2);                         //swap s1 and s2
   cout << s1 << " nor " << s2 << endl;
   return 0;
   }
```

```
Output:
s3 = Man
s3 = Neither Man
nor Beast
Beast nor Man
```


The Standard C++ string Class (2)

- Input/Output with `string` Objects

```
// sstrio.cpp
// string class input/output
#include <iostream>
#include <string>                  //for string class
using namespace std;

int main()
   {                          //objects of string class
   string full_name, nickname, address;
   string greeting("Hello, ");

   cout << "Enter your full name: ";
   getline(cin, full_name);  //reads embedded blanks
   cout << "Your full name is: " << full_name <<
    endl;

   cout << "Enter your nickname: ";
   cin >> nickname;          //input to string object

   greeting += nickname;   //append name to greeting
   cout << greeting << endl;  //output: "Hello, Jim"

   cout << "Enter your address on separate lines\n";
   cout << "Terminate with '$'\n";
   getline(cin, address, '$');//reads multiple lines
   cout << "Your address is: " << address << endl;
   return 0;
   }
```

```
Enter your full name: F. Scott Fitzgerald
Your full name is: F. Scott Fitzgerald
Enter your nickname: Scotty
Hello, Scotty
Enter your address on separate lines:
Terminate with '$'
1922 Zelda Lane
East Egg, New York$
Your address is:
1922 Zelda Lane
East Egg, New York
```

The Standard C++ string Class (3)

- Finding `string` Objects

```
//sstrfind.cpp
//finding substrings in string objects
#include <iostream>
#include <string>
using namespace std;

int main()
   {
   string s1 =
      "In Xanadu did Kubla Kahn a stately pleasure
     dome decree";
   int n;

   n = s1.find("Kubla");
   cout << "Found Kubla at " << n << endl;

   n = s1.find_first_of("spde");
   cout << "First of spde at " << n << endl;

   n = s1.find_first_not_of("aeiouAEIOU");
   cout << "First consonent at " << n << endl;
   return 0;
   }
```

Output:
```
Found Kubla at 14
First of spde at 7
First consonent at 1
```

- Modifying `string` Objects

```
//sstrchng.cpp
//changing parts of string objects
#include <iostream>
#include <string>
using namespace std;

int main()
   {
   string s1("Quick! Send for Count Graystone.");
   string s2("Lord");
   string s3("Don't ");

   s1.erase(0, 7);               //remove "Quick! "
   s1.replace(9, 5, s2); //replace "Count" with "Lord"
   s1.replace(0, 1, "s");     //replace 'S' with 's'
   s1.insert(0, s3);    //insert "Don't " at beginning
   s1.erase(s1.size()-1, 1);     //remove '.'
   s1.append(3, '!');            //append "!!!"

   int x = s1.find(' ');         //find a space
   while( x < s1.size() )   //loop while spaces remain
      {
      s1.replace(x, 1, "/");     //replace with slash
      x = s1.find(' ');          //find next space
      }
   cout << "s1: " << s1 << endl;
   return 0;
   }
```

Output:
```
s1: Don't/send/for/Lord/Graystone!!!
```


The Standard C++ string Class (4)

- Comparing `string` Objects

```
//sstrcom.cpp
//comparing string objects
#include <iostream>
#include <string>
using namespace std;

int main()
   {
   string aName = "George";
   string userName;

   cout << "Enter your first name: ";
   cin >> userName;
   if(userName==aName)                 //operator ==
      cout << "Greetings, George\n";
   else if(userName < aName)           //operator <
      cout << "You come before George\n";
   else
      cout << "You come after George\n";
                           //compare() function
   int n = userName.compare(0, 2, aName, 0, 2);
   cout << "The first two letters of your name ";
   if(n==0)
      cout << "match ";
   else if(n < 0)
      cout << "come before ";
   else
      cout << "come after ";
   cout << aName.substr(0, 2) << endl;
   return 0;
   }
```

```
Enter your first name: Alfred
You come before George
The first two letters of your name come before Ge
```

- Accessing Characters in `string` Objects

```
//sstrchar.cpp
//accessing characters in string objects
#include <iostream>
#include <string>
using namespace std;

int main()
   {
   char charray[80];
   string word;

   cout << "Enter a word: ";
   cin >> word;
   int wlen = word.length();//length of string object

   cout << "One character at a time: ";
   for(int j=0; j<wlen; j++)
      cout << word.at(j);//exception if out-of-bounds
//    cout << word[j]; //no warning if out-of-bounds

   word.copy(charray, wlen, 0); //copy string object
     to array
   charray[wlen] = 0;          //terminate with '\0'
   cout << "\nArray contains: " << charray << endl;
   return 0;
   }
```

```
Enter a word: symbiosis
One character at a time: symbiosis
Array contains: symbiosis
```

Summary (1)

- Arrays contain a number of data items of the same type. This type can be a simple data type, a structure, or a class.
  - The items in an array are called elements. Elements are accessed by number; this number is called an index.
  - Elements can be initialized to specific values when the array is defined.
  - Arrays can have multiple dimensions. A two-dimensional array is an array of arrays.
  - The address of an array can be used as an argument to a function; the array itself is not copied.
  - Arrays can be used as member data in classes.
  - Care must be taken to prevent data from being placed in memory outside an array.
- C-strings are arrays of type char .
  - The last character in a C-string must be the null character, '\0' .
  - C-string constants take a special form so that they can be written conveniently: the text is surrounded by double quotes. A variety of library functions are used to manipulate C-strings.

Summary (2)

- An array of C-strings is an array of arrays of type char.
  - The creator of a C-string variable must ensure that the array is large enough to hold any text placed in it.
  - C-strings are used as arguments to C-style library functions and will be found in older programs. They are not normally recommended for general use in new programs.
- The preferred approach to strings is to use objects of the string class.
  - These strings can be manipulated with numerous overloaded operators and member functions.
  - The user need not worry about memory management with string objects.