WEEK 7
ARRAYS AND STRINGS

Dr. Yuzana Win

ARRAYS AND STRINGS

Objectives

After completing this topic, you will be able to

- Give a brief overview of arrays and strings
- Explain the memory layout of single and multi dimensional arrays
- List the merits and de-merits of arrays

OUTLINES

- Arrays Fundamentals
- Arrays and Strings
  - One-dimensional Array
  - Multi-dimensional Array
  - Memory Organization
  - Strings
  - Advantages & limitations
- Arrays as Class Member Data
- Arrays of Structures
- Arrays of Strings

ARRAYS

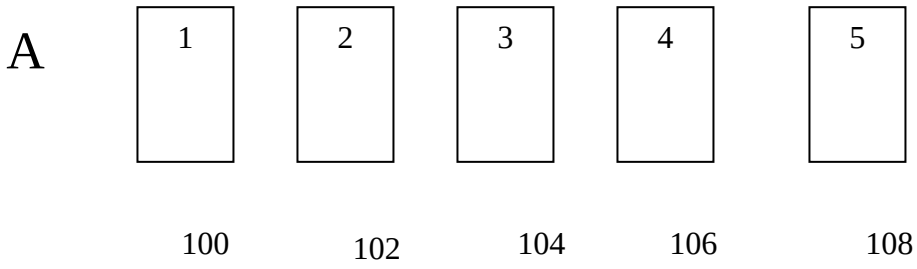
- Introduction
  - Array is one of the simplest data structure in computer programming.
  - Arrays hold a fixed number of equally sized data elements, generally of the same data type.

INTRODUCTION TO ARRAYS

- Arrays are a data type that are used to represent a large number of homogeneous values, that is values that are all of the one data type.
- The data type could be of type char, in which case we have a string.
- The data type could just as easily be of type int, float or even another array.

MEMORY ORGANIZATION FOR AN ARRAY

- The elements in the Array are always stored in consecutive memory locations.
- The Memory organization of an integer array A of 5 elements is shown below

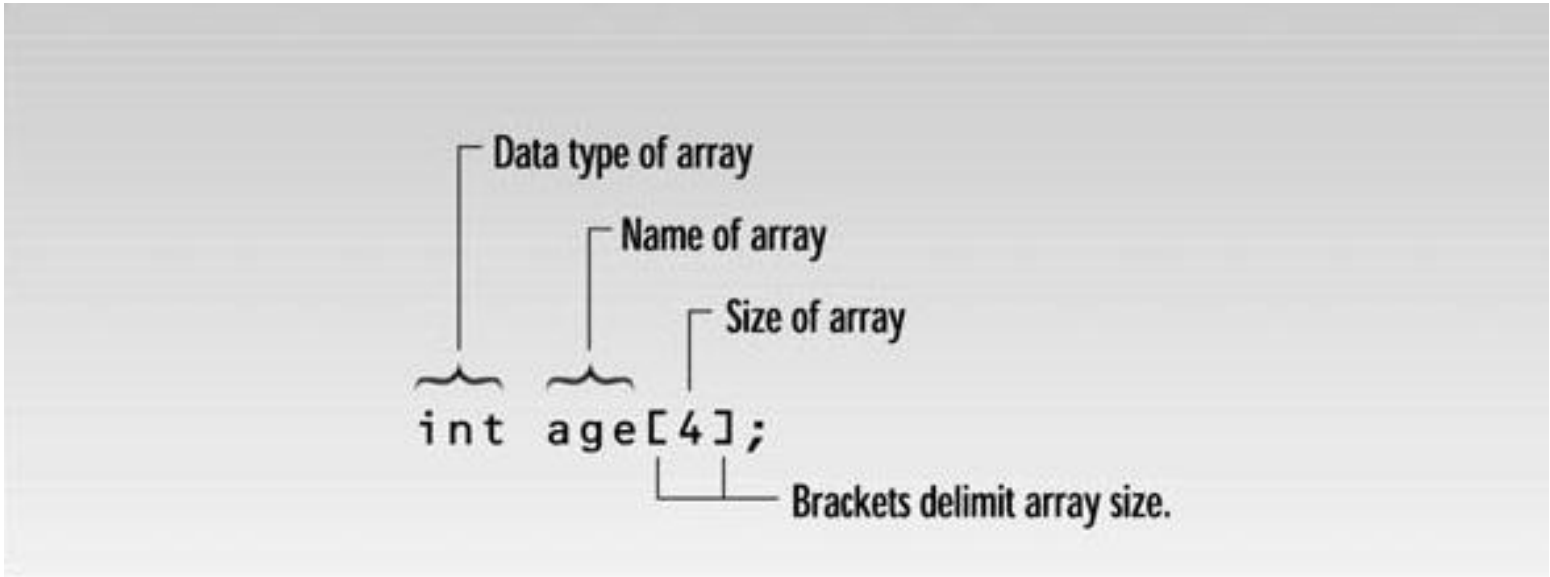


ARRAY DECLARATION AND INITIALIZATION

- Syntax

  dataType arrayName[arraysize];

Example: int age[4]; // array of 4 integer elements



ARRAY INITIALIZATION

- Array Initialization

```
int a[5] = {1,2,3,4,5};
int a[] = {1,2,3,4,5};
float d[3] = {1.23,3.5,4.4};
char name[] = “Jhon Smith”;
char name[5] = {‘J’, ‘h’, ‘o’, ‘n’, ‘/0’};
```

ONE-DIMENSIONAL ARRAY

- Insert value to array element

```
int a[3];
  a[0]=1;    a[1]=2;    a[2]=3;
//ask and insert elements
for ( int i=0;i<3;i++)
{
   cout <<  "\nEnter element :" << i << "is";
   cin >> a[i];
}
// print array elements
for ( int i=0;i<3;i++)
{
   cout << "\n Element at" << i << "is :" << a[i];
}
```

```
Enter element 0 is 1
Enter element 1 is 2
Enter element 2 is 3
Element at0 is :1
Element at1 is :2
Element at2 is :3
```

ARRAY EXAMPLES

```
// replay.cpp
// gets four ages from user, displays them
#include <iostream>
using namespace std;
int main()
{
     int age[4];
     for(int j=0; j<4; j++)
    {
          cout << “Enter an age:”;
          cin >> age[j];
    }
    for(j=0; j<4; j++)
          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
```

AVERAGING ARRAY ELEMENTS

```
// sales.cpp
// averages a weeks's widget sales (6 days)
#include <iostream>
using namespace std;
 int main()
{
    const int SIZE = 6;
    double sales[SIZE];
    cout << "Enter widget sales for 6 days\n";
    for(int j=0; j<SIZE; j++)
            cin >> sales[j];
    double total = 0;
    for(int j=0; j<SIZE; j++)
          total += sales[j];
    double average = total / SIZE;
    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
```

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):";
    cin >> month;
    cout << "Enter day (1 to 31):";
    cin >> day;
    total_days = day;
    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
```

MULTI-DIMENSIONAL ARRAYS

- The elements of an array can themselves be arrays.

1 2 3
4 5 6
7 8 9

- Multi-dimensional array is **an array of array(s)**.
- For example, two-dimensional array is an array of arrays.

```
int  matrix[3][3];

matrix[0][0]=1; matrix[0][1]=2; matrix[0][2]=3;
matrix[1][0]=4; matrix[1][1]=5; matrix[1][2]=6;
matrix[2][0]=7; matrix[2][1]=8; matrix[2][2]=9;
```

(or)

```
matrix[3][3]={{1,2,3},{4,5,6},{7,8,9}};   // initialize value
```

ELEMENTS IN A TWO- DIMENSIONAL ARRAY

- The following example declares and creates an integer array with 3 rows and 5 columns

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

int main()
{
    //high scores for 3 players and their top scores
    int high_scores[3][5] = {{500,550,625,700,850},
                             {385,425,450,475,495},
                             {325,330,375,360,380}};
```

ELEMENTS IN A TWO- DIMENSIONAL ARRAY

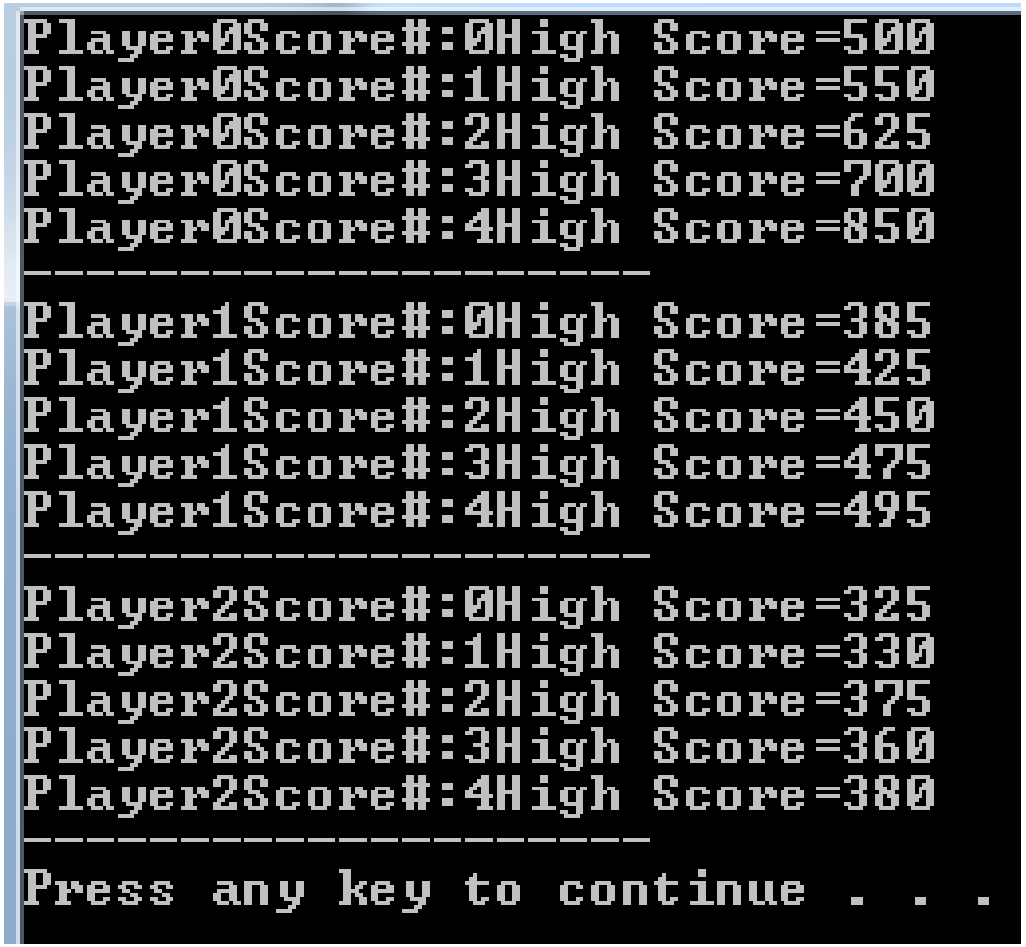
```
//start loopping over rows
for (int i = 0; i < 3; i++)
   {   //now the rows
          for (int j = 0; j < 5; j++)
                    { cout << "Player" << i << "Score#:" << j;
                      cout << "High Score=" << high_scores[i][j] <<
                      endl;
                    }
            cout << "--------------------" << endl;
    }

system("pause");
return 0;
}
```

- The elements are accessed as high_scores[i][j]. This is a consistent notation since high_scores[i][j] is element j of the array high_scores[i].

ELEMENTS IN A TWO- DIMENSIONAL ARRAY

Two-dimensional Array Output



MULTIDIMENSIONAL ARRAYS

```
// displays sales chart using 2-d array
#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];
}
```

```
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)
    << setiosflags(ios::showpoint)
    << setprecision(2)
    << setw(10)
    << sales[d][m];
}
     cout << endl;
     return 0;
}
```

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
```

ARRAYS OF STRUCTURES

```
// partaray.cpp
// structure variables as array elements
#include <iostream>
using namespace std;
const int SIZE = 4;

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

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
```

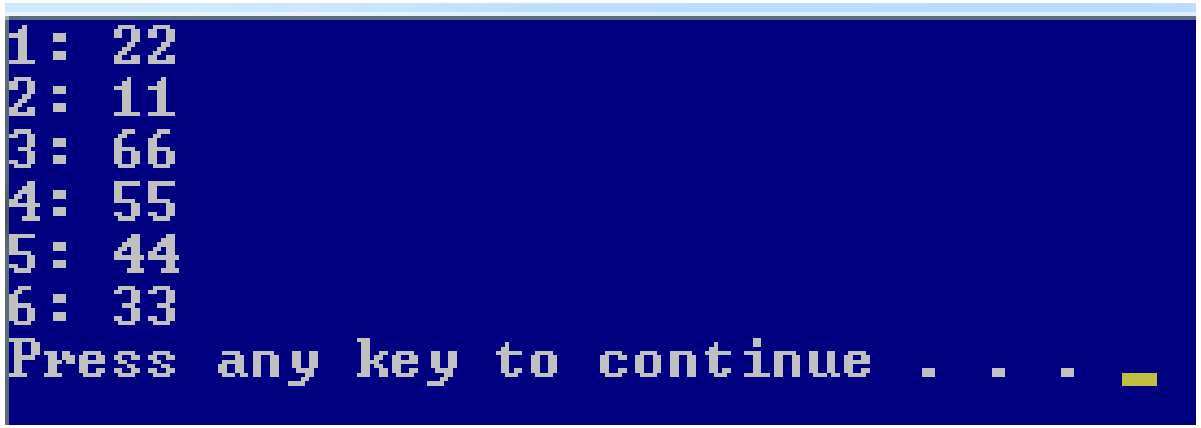
ARRAYS AS CLASS MEMBER DATA

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

```
int main()
{
   Stack s1;
   s1.push(11);
   s1.push(22);
   cout << "1: " << s1.pop() << endl;
   cout << "2: " << s1.pop() << endl;

  //get values for all members
    cout << endl;
  for(n=0; n<SIZE; n++)
  {
      cout << “Model” <<
      apart[n].modelnumber;
      cout << “ Part”<<
       apart[n].partnumber;
      cout << “ Cost”<<
       apart[n].cost << endl;
  }
  return 0;
  }
```

Output for simple stack class



ADVANTAGES & LIMITATIONS

Advantages

- Arrays are,
  - Simple and easy to understand
  - Contiguous allocation
  - Fast retrieval because of its indexed nature
  - No need for the user to be worried about the allocation and de-allocation of arrays

Limitations

- If you need m elements out of n locations defined,
  - n-m locations are unnecessarily wasted if n>m (waste memory space)

  or
  - an error occurs if m>n named out of bounds error.
- (fixed size(static) memory allocation)

STRINGS

- Strings are sequences of characters.
- Strings are just character arrays with a few restrictions. One of these restrictions is that the special character '\0' (NULL) is used to indicate the end of a string.
- Declaration and Initialization

```
char name[10]={'J', 'o', 'h', 'n', '/0'};
char name[10]="John";
char name[3];
        name[0] = 'A';
        name[1] = 'R';
        name[2] = '\0';
// Enter from user
cout << "Enter name ";
cin >> "name";
```

C-STRINGS

- cstring header file
- There are several standard routines that work on string variables.
- Some of the string manipulation functions are,

strcpy(string1, string2); - copy string2 into string1

strcat(string1, string2); - concatenate string2 onto the end of string1

int **strcmp(string1,string2);** - **0** if string1 is equals to string2,
< 0 if string1 is less than string2
>0 if string1 is greater than string2

int **strlen(string)** - get the length of a string.

strchr(string1,char) - search specified character in string1

strrev(string1) - reverse character in string1

C-STRINGS 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;
   system("pause");
   return 0;
}
```

```
Enter a string: John
You entered: John
Press any key to continue . . .
```

READING EMBEDDED BLANKS

To read text containing blanks we use another function, **cin.get()**

```
// 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;
system("pause");
return 0;
}
```

```
Enter a string: Law is a bottomless pit.
You entered: Law is a bottomless pit.
Press any key to continue . . .
```

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;
system("pause");
return 0;
}
```

```
Enter a string:
Ask me no more where Jove bestows
When June is past, the fading rose;
$
You entered:
Ask me no more where Jove bestows
When June is past, the fading rose;

Press any key to continue . . .
```

COPYING A STRING

```
// 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";
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
cout << "two string concat is:" << strcat(str1,str2) << endl;
cout << "compare result is: " << strcmp(str1,str2) << endl;
cout << "str length:" << strlen(str2) << endl;
system("pause");
return 0; }
```

```
Tiger, tiger, burning bright
two string concat is:Tiger, tiger, burning brightTiger, tiger, burning bright
compare result is: 1
str length:28
Press any key to continue . . .
```

ARRAY OF STRING

```
// 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;
system("pause");
return 0;
}
```

```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Press any key to continue . . .
```

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

```
Enter your full name: Yuzana Win
Your full name is: Yuzana Win
Enter your nickname: Yu
Hello,Yu
Enter your address on separate lines
Terminate with '$'
Yangon, Myanmar
$
Your address is:
Yangon, Myanmar

Press any key to continue . . .
```

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 consonant at " << n << endl;
system("pause");
return 0; }
```

```
Found Kubla at14
First of spde at 7
First consonant at 1
Press any key to continue . . .
```

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;
system("pause");
return 0;
}
```

```
s1: Don't/send/for/Lord/Graystone!!!
Press any key to continue . . .
```

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;
system("pause");
return 0;
}
```

```
Enter a word: symbiosis
One character at a time: symbiosis
Array contains: symbiosis
Press any key to continue . . .
```

SUMMARY (ARRAYS AND STRINGS)

- The simplest type of data structure is a linear array
- An array can be defined as a collection of homogenous elements, in the form of index/value pairs, stored in consecutive memory locations. An array always has a predefined size and the elements of an array are referenced by means of an index.
- An array can be of more than one dimension. There are no restrictions to the number of dimensions that we can have.
- Arrays of Characters are called as Strings

EXERCISES

- Write a C++ program that calculates and prints average value of up to 20 test scores.
- Write a program that count the no. of odd and even from the int array.
- Write a program to find the number of incidence (i.e., equal and adjacent no.) from int array.

  Eg. 2,3,3,4,5,1,1,7,8 => no. of incidence = 2

  2,4,7,9,1,4,3=>no. of incidence = 0
- Write a C++ program that creates two arrays one for part number and one for price for each part. Moreover that program finds the corresponding price for given part number.