Object Oriented Programming

Shaobai Kan

chapter 7 (Continue...)

Outline

Outline

- C-style, pointer-based string processing

Outline

- C-style, pointer-based string processing
- Arrays of pointers

C-style, pointer-based string processing

C++ strings

Recall. C++ offers two types of strings

- string class objects
- C-style, pointer-based strings

C++ strings

Recall. C++ offers two types of strings

- string class objects
- C-style, pointer-based strings

Fact. C++'s string class is preferred for use in new programs, because it eliminates many of the security problems that can be caused by manipulating C strings.

Character *v.s.* String

Character — A character constant is an integer value represented as a character in single quotes.

—— e.g. 'a', '\n', '\0'

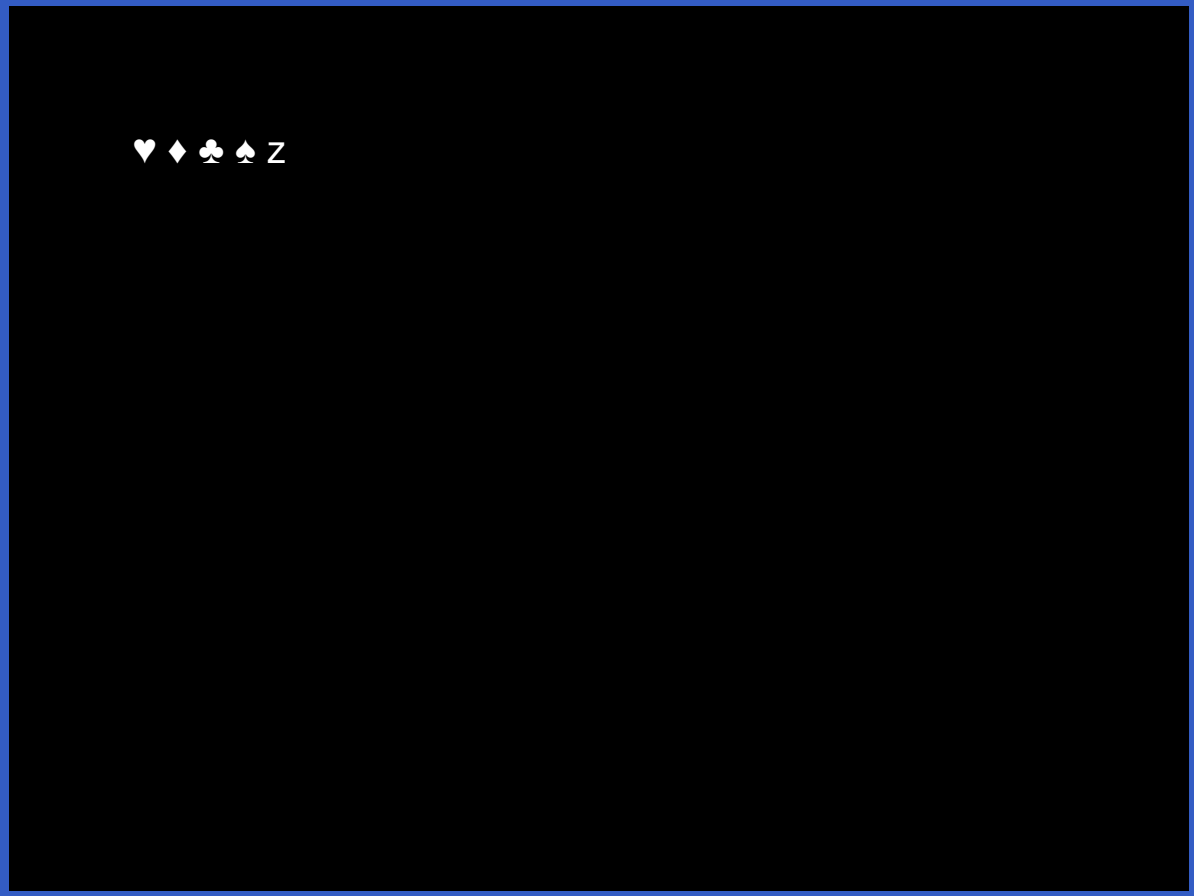
Fact. The value of a character constant is the integer value of the character in the machine's character set.

—— ASCII character set

Example: Character

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

int  main ( )
{
        char a, b, c, d, e;
        a = 3 ;
        b = 4 ;
        c = 5 ;
        d = 6 ;
        e = 122 ;

        cout << a << ' '  << b << ' ' << c
             << ' ' << d << ' ' << e << endl ;

        return 0 ;
   }
```

Example: Character



Character *v.s.* String

String — A string is a series of characters treated as a single unit; string literals or string constants in C++ are written in double quotation marks.

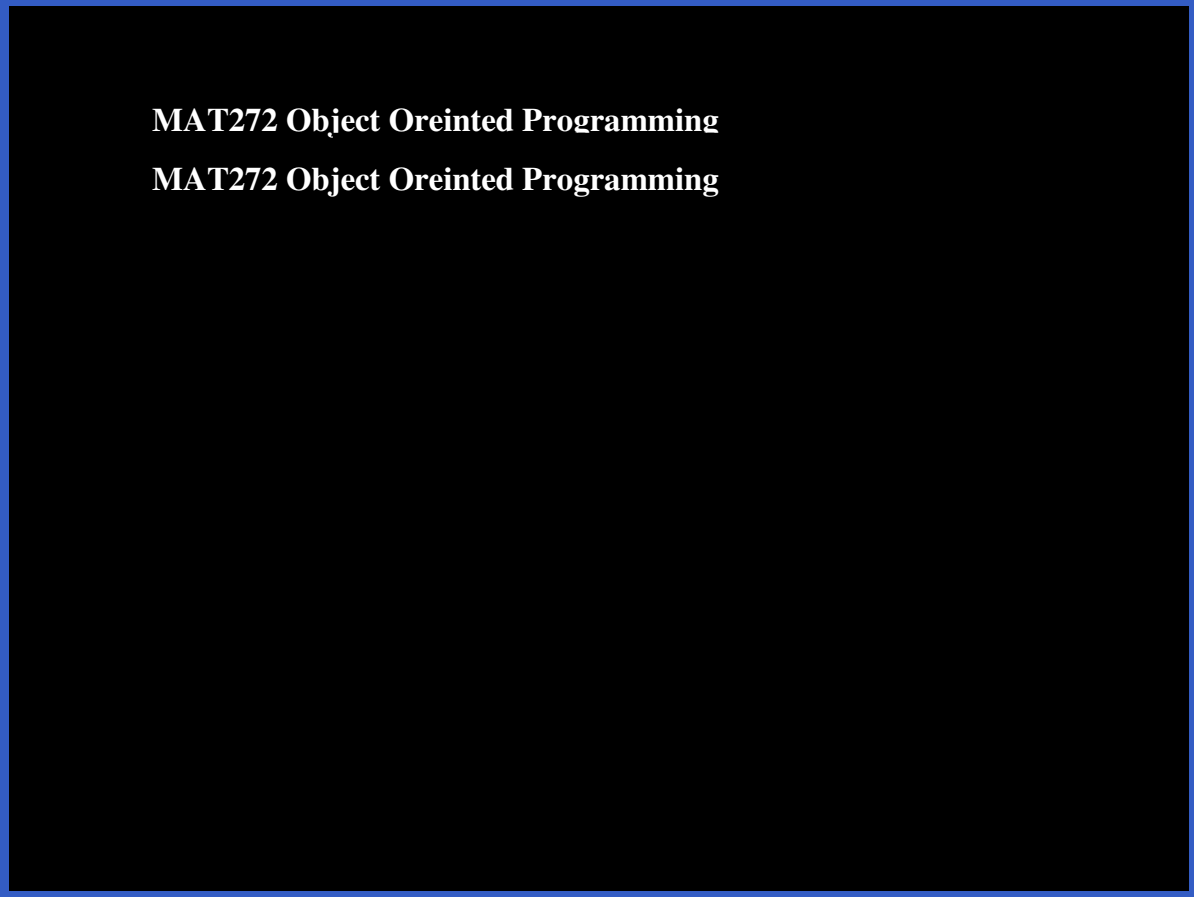
—— e.g. "Shaobai Kan", "(201)-1212"

Fact. A pointer-based string is an array of characters ending with a null character ('\0').

Example1: String

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

int  main ( )
{
    char nameOfCourse[ ] = "MAT272 Object Oriented Programming";
    //const char * nameOfCourse = "MAT272 Object Oriented Programming";

    //Printing the string
    for ( int i = 0; nameOfCourse[i] != '\0' ; i ++ )
          cout << nameOfCourse[i];

    cout << endl;

    //Printing the string
     cout << nameOfCourse;

    cout << endl;

   return 0 ;
 }
```

Example1: String

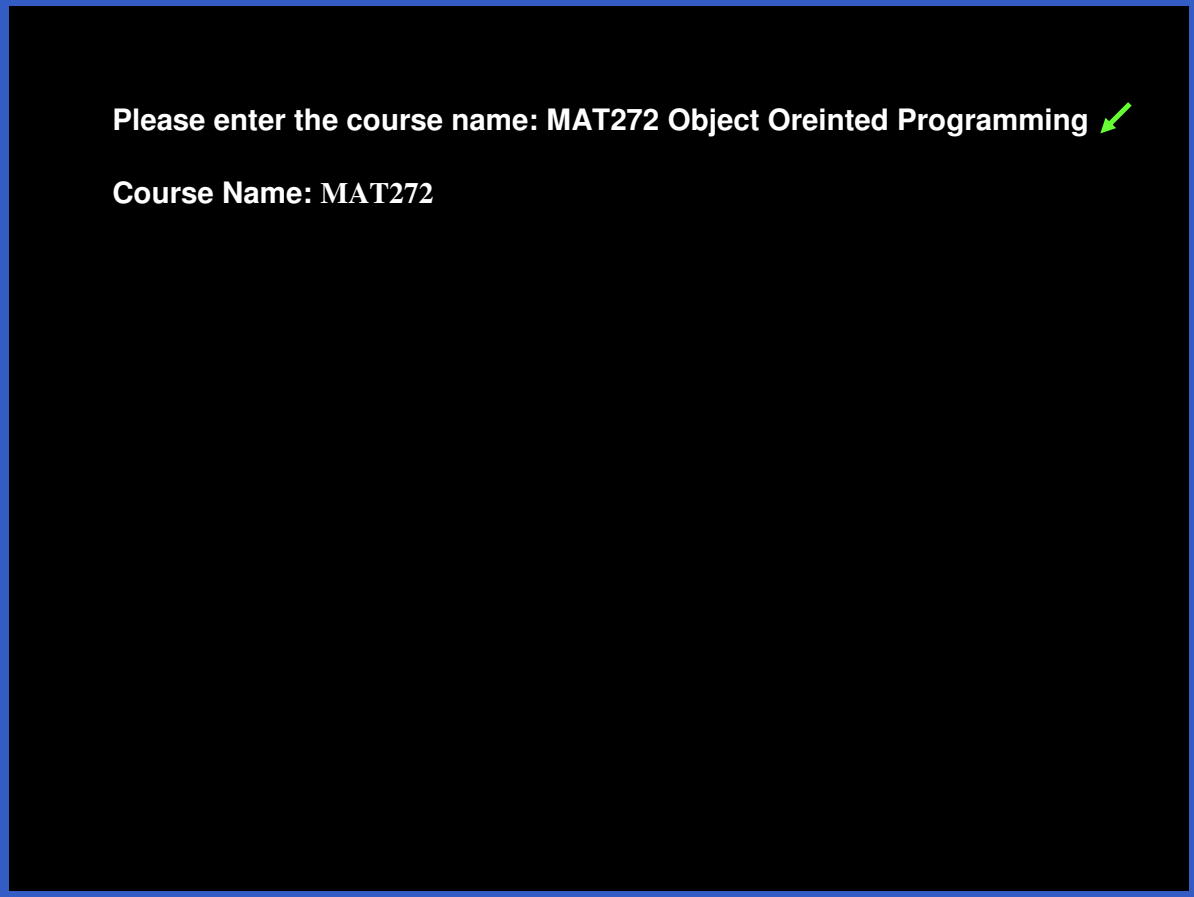


Example2: String

Watch Out!

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

int main( )
{
        char nameOfCourse [35];

        //Title: MAT272 Object Oriented Programming
        cout << "Please enter the course name: ";
        cin >> nameOfCourse;

        cout << endl;

        //Printing the course name
        cout << "Course Name: " << nameOfCourse;

        cout << endl;

        return 0 ;
 }
                // Question: What will happen?
```

Example2: String



getline function

Fact. Users sometimes want to input an entire line of text into a character array. In these cases, cin object provides the member function getline.

getline function

Fact. Users sometimes want to input an entire line of text into a character array. In these cases, cin object provides the member function getline.

***getline* function** takes three arguments:

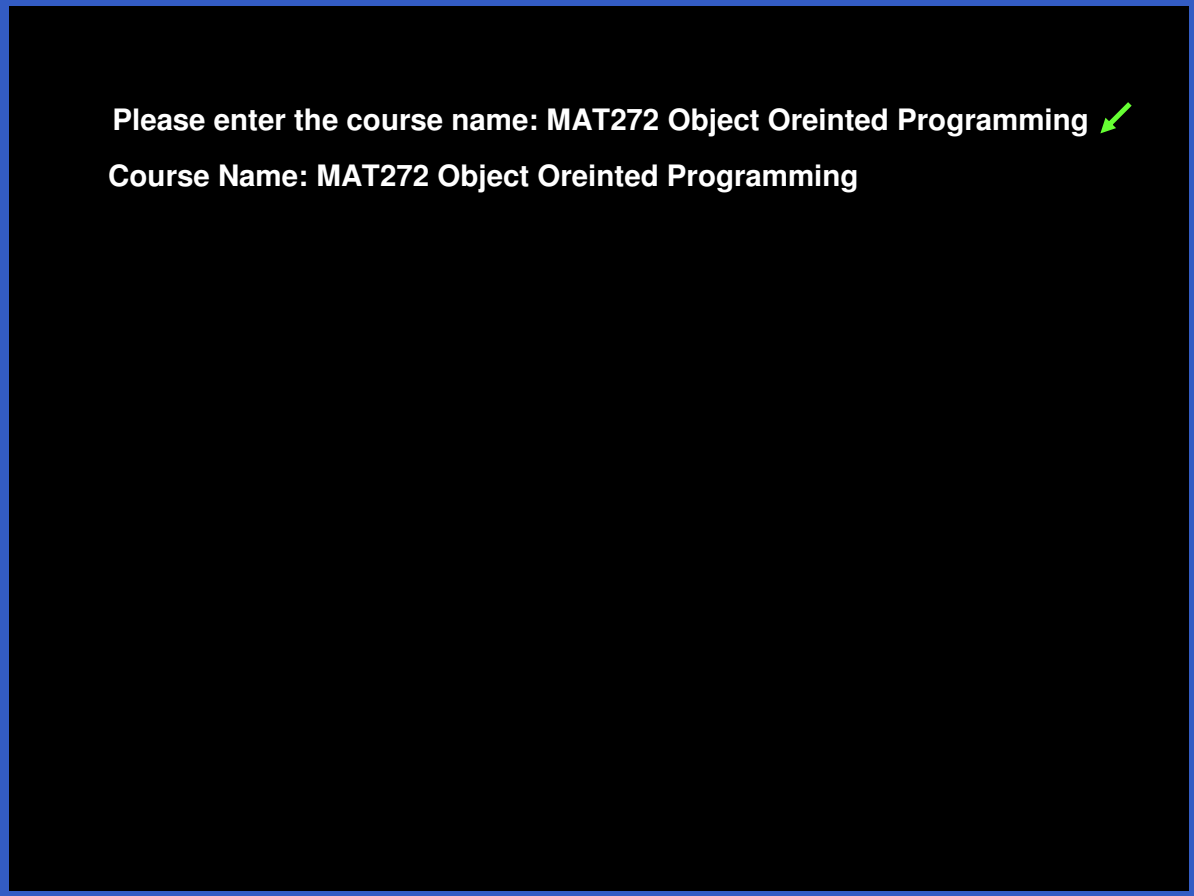
- a character array
- a length
- delimiter character, e.g. '\n'

Example3: String

Watch Out!

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

int main( )
{
        char nameOfCourse [35];

        //Title: MAT272 Object Oriented Programming
        cout << "Please enter the course name: ";
        cin.getline (nameOfCourse, 35, '\n');


        cout << endl;

        //Printing the course name
        cout << "Course Name: " << nameOfCourse;

        cout << endl;

        return 0 ;
 }
                // Question: What will happen?
```

Example3: String



Arrays of pointers

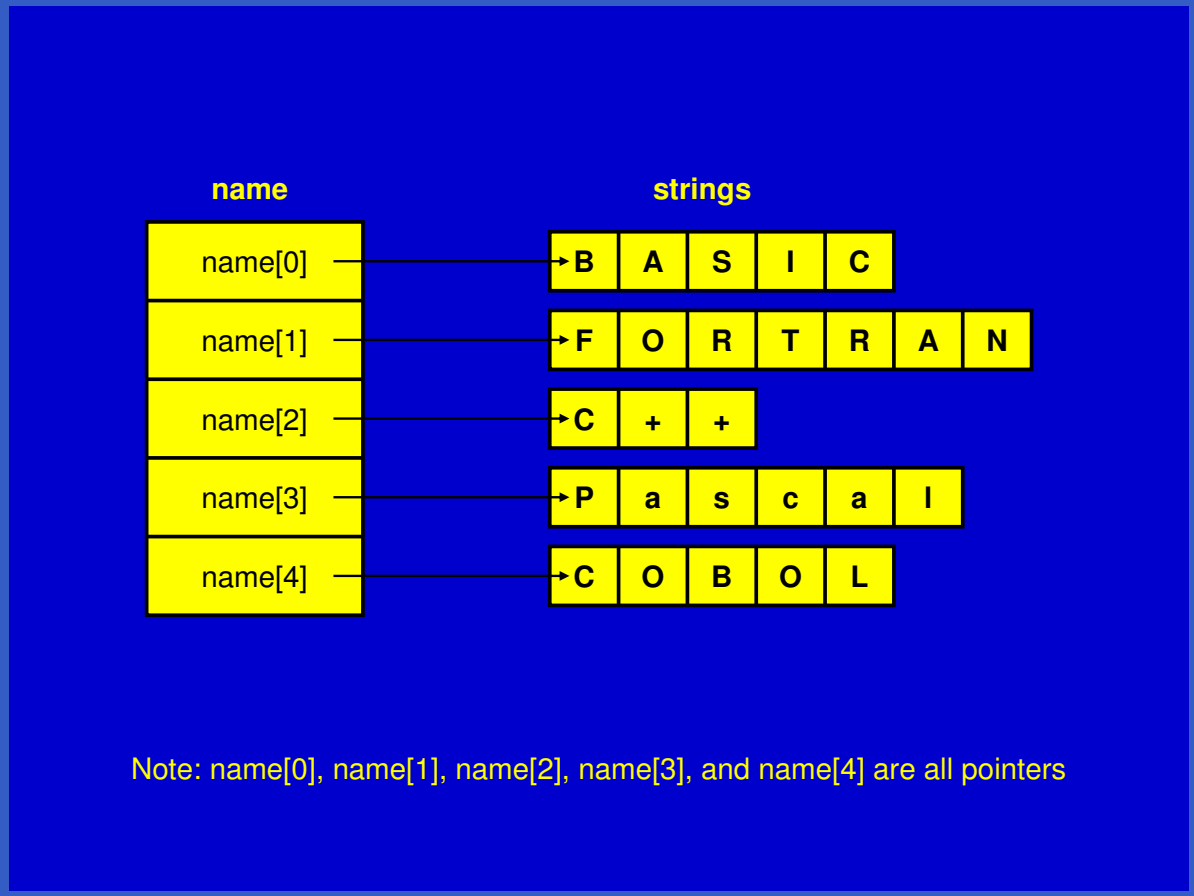
Arrays of pointers

Fact. Array can contain pointers; a common use of such a data structure is to form an array of pointer-based strings (i.e. string array).

e.g.

```
const char * name[ ] = { "BASIC", "FORTRAN", "C++",
                         "PASCAL", "COBOL" };
```

Visual representation: array of pointers



Note: name[0], name[1], name[2], name[3], and name[4] are all pointers

Example1: Array of pointers

Array of pointers

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

void print ( const char * name [ ], int n );

int main( )
{
     char * name [ ] = { “BASIC”, “FORTRAN”, “C++”, “Pascal”, “COBOL” };
     int n = 5;
     print ( name, n);
     return 0 ;
 }


 void print ( const char * name [ ], int n )
{
     for ( int i = 0 ; i < n ; i ++ )
          cout << name [ i ] << endl;
 }
```

Example2: Array of pointers

Homework:

- Read Sec. 7.10 - 7.12
- practice the program in Fig 7.20 (in the textbook section 7.12)
- Exercise 7.11, 7.13, 7.14