CS1002 – Programming Fundamentals

Lecture # 19
Wednesday, October 26, 2022
FALL 2022
FAST – NUCES, Faisalabad Campus

Rizwan Ul Haq

String Datatype

Basic Functions

The string Type

- To use the data type string, the program must include the header file string

  ```
  #include <string>
  ```

- The statement

  ```
  string name = "William Jacob";
  ```

  declares name to be string variable and also initializes name to "William Jacob".

- The position of the first character, 'W', in name is 0, the position of the second character, 'i', is 1, and so on
- The variable name is capable of storing (just about) any size string

The string Type

```
String name = "Ameer Hamza";                    //error
string name = “Ameer Hamza”;                    //OK
```

C++ is a case-sensitive language, means; upper-case and lower-case are two different things

```
string var = "";                //empty string in var

string var = 'h';               //error; value must be in double quotation marks “ ”
string var = "we never practice programing at home";
cout<<var[0]<<endl;
cout<<var[1]<<endl;
cout<<var[2]<<endl;

for(int i=0; var[i] != '\0'; i++)
    cout<<var[i];
```

String basic functions

- Binary operator + (to allow the string concatenation operation), and the array index (subscript) operator [], have been defined for the data type string
- Suppose we have the following declarations

  string str1, str2, str3;

- The statement

  str1 = "Hello There“ ;

  stores the string "Hello There" in str1.

- The statement

  str2 = str1;

  copies the value of str1 into str2.

String basic functions

- If **str1 = "Sunny",** the statement

  str2 = str1 + " Day";

  stores the string "Sunny Day" into str2.

- If **str1 = "Hello“** and **str2 = "There“** then

  str3 = str1 + " " + str2;

  stores "Hello There" into **str3**

- This statement is equivalent to the statement

  str3 = str1 + ' ' + str2;

String basic functions

- The statement

  str1 = str1 + "Mickey“ ;

  updates the value of str1 by appending the string "Mickey" to its old value

- If **str1 = "Hello there"**, the statement

  str1[6] = 'T‘ ;

  replaces the character t with the character T.

The length **Function**

- The **length** function returns the number of characters currently in the string
- The value returned is an unsigned integer
- The syntax to call the length function is:

strVar.length()

where strVar is variable of the type string

- The function length has no arguments

String datatype

- Consider the following statements:

```
string firstName ;
string name ;
string str ;
firstName = “Elizabeth";
name = firstName + " Taylor";
str = "It is sunny outside.";
```

Statement	Output
`cout << firstName.length() << endl;`	`9`
`cout << name.length() << endl;`	`16`
`cout << str.length() << endl;`	`20`

The size Function

- The function size is same as the function length
- Both these functions return the same value
- The syntax to call the function size is:

  strVar.size()

  where **strVar** is variable of the type string.

- As in the case of the function length, the function size has no arguments

The find Function

- The find function searches a string to find the first occurrence of a particular substring and returns an unsigned integer value (of type **string::size_type**)
- If the search is unsuccessful, the function returns the special value of dataType **string::npos**
- The syntax to call the function find is:

  strVar.find(strExp)
- For the search to be successful, the match must be exact
- **string::size_type** is the dataType of value returned by the find function
- **string::npos** is member with value as the highest possible for the size_t data structure.

String datatype

- The following are valid calls to the function find

str1.find(str2)

str1.find("the")

str1.find('a')

str1.find(str2+"xyz")

str1.find(str2+'b')

String datatype

```
string   sentence;
string   str;
string::size_type position;

sentence = "It is cloudy and warm.";
str = "cloudy";
```

Statement	Effect
`cout << sentence.find("is") <<endl;`	Outputs 3
`cout << sentence.find("and") <<endl;`	Outputs 13
`cout << sentence.find('s') <<endl;`	Outputs 4
`cout << sentence.find(str) <<endl;`	Outputs 6
`cout << sentence.find("the") <<endl;`	Outputs the value of string::npos
`position = sentence.find("warm");`	Assigns 17 to position

The substr Function

- The **substr** function returns a particular substring of a string
- The syntax to call the function **substr** is:

  strVar.substr(intExpr1, intExpr2)

  where expr1 and expr2 are expressions evaluating to unsigned integers.

- intExpr1: a position within the string (starting position of the substring).
- intExpr2 specifies the length of the substring to be returned.

The substr Function

string sentence;
string str;

sentence = "It is cloudy and warm.";

Statement	Effect
`cout << sentence.substr(0,5)<<endl;`	`Outputs: It is`
`cout << sentence.substr(6,6)<<endl;`	`Outputs: cloudy`
`cout << sentence.substr(6,16)<<endl;`	`Outputs: cloudy and warm.`
`cout << sentence.substr(3,6)<<endl;`	`Outputs: is clo`
`str = sentence.substr(0,8);`	`str = "It is cl“`
`str = sentence.substr(2,10);`	`str = " is cloudy"`

The Function swap

- The function **swap** is used to swap—that is, interchange—the contents of two string variables
- The syntax to use the function swap is

  strVar1.swap(strVar2);

  where **strVar1** and **strVar2** are string variables.

- Suppose you have the following statements:

```
string str1 = "Warm";
string str2 = "Cold";
```

- After the following statement executes, the value of **str1** is **"Cold"** and the value of **str2** is **"Warm"**.

  str1.swap(str2);

Arrays of Strings

- Strings in C++ can be manipulated using either the data type string or character arrays (C-strings)
- On some compilers, the data type **string** may not be available in Standard C++ (i.e., non-ANSI/ISO Standard C++)

Arrays of Strings and the string Type

- To declare an array of 100 components of type string:

```
string list[100];
```

- Basic operations, such as assignment, comparison, and input/output, can be performed on values of the **string** type
- The data in **list** can be processed just like any one-dimensional array

Arrays of Strings and C-Strings (Character Arrays)

- Consider declaration

  char list[100][16];

- Now list[j] for each j, 0 <= j <= 99, is a string of at-most 15 characters in length

```
strcpy(list[1], "Snow White");
```

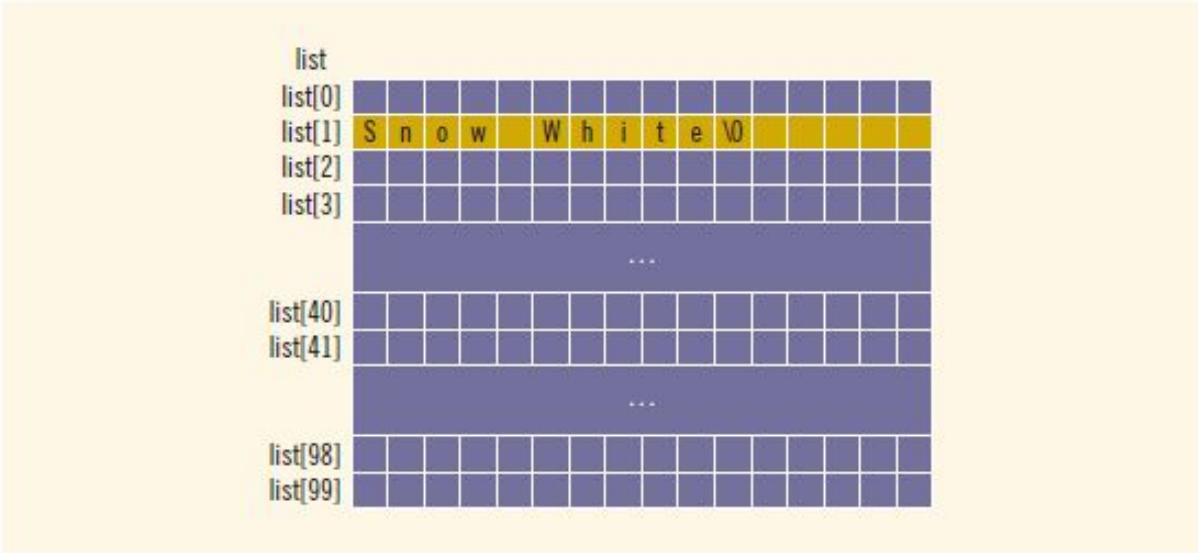

FIGURE 9-17 Array list, showing list[1]

Contd..

Suppose that you want to read and store data in a list and there is one entry per line.
The following code accomplishes this:

```
char list[100][16];
for (int i = 0; i < 100; i++)
{
    cin.get(list[i], 16);
    cin.ignore();
}
```

The following for loop outputs the string in each row:

```
for (int i = 0; i < 100; i++)
    cout << list[i] << endl;
```

You can also use other string functions (such as **strcmp** and **strlen**) and for loops to manipulate list

Questions