Programming In C++

Course 2: Lecture 6, String

Prepared By Dr. Ali Mohsin

Nineveh University- Faculty of IT- department of software

Local Variables

Variables defined inside a function are **local** to that function. They are hidden from the statements in other functions, which normally cannot access them.

```
void anotherFunction()
{
     int num = 20; // Local variable

     cout << "In anotherFunction, num is " << num << endl;
}

int main()
{
     int num = 1; // Local variable

     cout << "In main, num is " << num << endl;
     anotherFunction();
     cout << "Back in main, num is " << num << endl;
     return 0;
}
```

Program Output

```
In main, num is 1
In anotherFunction, num is 20
Back in main, num is 1
```

Global Variables

A global variable is any variable defined outside all the functions in a program. The scope of a global variable is the portion of the program from the variable definition to the end.

global variable can be accessed by all functions that are defined after the global variable is defined.

```
int num = 2;
void anotherFunction()
{
    cout << "In anotherFunction, num is " << num << endl;
    num = 50;
    cout << "But, it is now changed to " << num << endl;
}
```

```
int main()
{
    cout << "In main, num is " << num << endl;
    anotherFunction();
    cout << "Back in main, num is " << num << endl;
    return 0;
}
```

Program Output

```
In main, num is 2
In anotherFunction, num is 2
But, it is now changed to 50
Back in main, num is 50
```

Global and Local Variables

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

void myFunc();  // Function prototype

int main()
{

     int var = 100;

     cout << var << endl;
     myFunc();
     cout << var << endl;
     return 0;
  }

  // Definition of function myFunc
  void myFunc()
  {
     int var = 50;

     cout << var << endl;
  }
```

What is the output of the program

String

The C++ library provides several functions for testing characters. To use these functions you must include the cctype header file.

Character Function	Description
`isalpha`	Returns true (a nonzero number) if the argument is a letter of the alphabet. Returns 0 if the argument is not a letter.
`isalnum`	Returns true (a nonzero number) if the argument is a letter of the alphabet or a digit. Otherwise it returns 0.
`isdigit`	Returns true (a nonzero number) if the argument is a digit from 0 through 9. Otherwise it returns 0.
`islower`	Returns true (a nonzero number) if the argument is a lowercase letter. Otherwise, it returns 0.
`isprint`	Returns true (a nonzero number) if the argument is a printable character (including a space). Returns 0 otherwise.
`ispunct`	Returns true (a nonzero number) if the argument is a printable character other than a digit, letter, or space. Returns 0 otherwise.
`isupper`	Returns true (a nonzero number) if the argument is an uppercase letter. Otherwise, it returns 0.
`isspace`	Returns true (a nonzero number) if the argument is a whitespace character. Whitespace characters are any of the following: space `' '` vertical tab `'\v'` newline `'\n'` tab `'\t'` Otherwise, it returns 0.

String

```
// This program demonstrates some character-testing functions.
#include <iostream>
#include <cctype>
using namespace std;

int main()
{
     char input;

     cout << "Enter any character: ";
     cin.get(input);
     cout << "The character you entered is: " << input << endl;

     if (isalpha(input))
         cout << "That's an alphabetic character.\n";
     if (isdigit(input))
         cout << "That's a numeric digit.\n";
     if (islower(input))
         cout << "The letter you entered is lowercase.\n";
     if (isupper(input))
         cout << "The letter you entered is uppercase.\n";
     if (isspace(input))
         cout << "That's a whitespace character.\n";
     return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter any character: A [Enter]
The character you entered is: A
That's an alphabetic character.
The letter you entered is uppercase.
```

Program Output with Different Example Input Shown in Bold

```
Enter any character: 7 [Enter]
The character you entered is: 7
That's a numeric digit.
```

String

```
cin.getline(customer, SIZE);
```

The C++ getline() is **a standard library function that is used to read a string or a line from an input stream**.

Character Case Conversion

The C++ library offers functions for converting a character to upper- or lowercase.

Function	Description
`toupper`	Returns the uppercase equivalent of its argument.
`tolower`	Returns the lowercase equivalent of its argument.

Each of the functions in Table accepts a single character argument. If the argument is a lowercase letter, the toupper function returns its uppercase equivalent. For example, the following statement will display the character A on the screen:

```
cout << toupper('a');    // Display A
cout << toupper('*');    // Displays *
cout << toupper ('&');   // Displays &
cout << toupper('%');    // Displays %
```

Character Case Conversion

toupper and tolower don't actually cause the character argument to change, they simply return the upper- or lowercase equivalent of the argument. For example, in the following program segment, the variable letter is set to the value 'A'. The tolower function returns the character 'a', but letter still contains 'A'.

```
char letter = 'A';
cout << tolower(letter) << endl;
cout << letter << endl;
```

These statements will cause the following to be displayed:

```
a
A
```

C-Strings Stored in Arrays

if you want to store a C-string in memory, you have to define a char array that is large enough to hold the string, plus one extra element for the null character. Here is an example:

```
const int SIZE = 21;
char name[SIZE] = "Jasmine";
```

You can implicitly size a char array by initializing it with a string literal, as shown here:

```
char name[] = "Jasmine";
```

C-Strings Stored in Arrays

C-string input can be performed by the cin object. For example, the following code allows the user to enter a string (with no whitespace characters) into the name array:

```
const int SIZE = 20;
char name[SIZE];
cin >> name;
```

cin getline member function to get a line of input (including whitespace characters) and store it in the line array:

```
cin.getline(line, SIZE);
```

The first argument tells getline where to store the string input.
The second argument indicates the maximum length of the string, including the null terminator. In this example, the SIZE constant is equal to 80

Library Functions for Working with C-Strings

The strlen Function

The strlen function to determine the length of the string stored in the name array:

```
char name[] = "Thomas Edison";
int length;
length = strlen(name);
```

The strlen function accepts a pointer to a C-string as its argument. It returns the length of the string, which is the number of characters up to, but not including, the null terminator.

the variable length will have the number 13 stored in it

Library Functions for Working with C-Strings

The strcat Function

The strcat function accepts two pointers to C-strings as its arguments. The function concatenates, or appends one string to another.

```
const int SIZE = 13;
char string1[SIZE] = "Hello ";
char string2[] = "World!";

cout << string1 << endl;
cout << string2 << endl;
strcat(string1, string2);
cout << string1 << endl;
```

These statements will cause the following output:

```
Hello
World!
Hello World!
```

Library Functions for Working with C-Strings

The strcpy Function

The strcpy function can be used to copy one string to another. Here is an example of its use:

```
const int SIZE = 13;
char name[SIZE];
strcpy(name, "Albert Einstein"); //the strcpy function will copy the string “Albert Einstein” to the name array.
```

```
const int SIZE = 10;
char string1[SIZE] = "Hello", string2[SIZE] = "World!";
cout << string1 << endl;
cout << string2 << endl;
strcpy(string1, string2);
cout << string1 << endl;
cout << string2 << endl;
```

Here is the output:

```
Hello
World!
World!
World!
```

Library Functions for Working with C-Strings

The strstr Function

The strstr function searches for a string inside of a string.

The function's first argument is the string to be searched.

The second argument is the string to look for.

If the function finds the second string inside the first, it returns the address of the occurrence of the second string within the first string. Otherwise it returns nullptr (the address 0).

```
char arr[] = "Four score and seven years ago";
char *strPtr = nullptr;
cout << arr << endl;
strPtr = strstr(arr, "seven"); // search for "seven"
cout << strPtr << endl;
```

this segment will display the following:

```
Four score and seven years ago
seven years ago
```

Library Functions for Working with C-Strings

The strcmp Function

Because C-strings are stored in char arrays, you cannot use the relational operators to compare two C-strings.

To compare C-strings, you should use the library function strcmp.

This function takes two C-strings as arguments and returns an integer that indicates how the two strings compare to each other.

```
int strcmp(char *string1, char *string2);
```

The value of the result is set accordingly:

- The result is zero if the two strings are equal on a character-by-character basis
- The result is negative if string1 comes before string2 in alphabetical order
- The result is positive if string1 comes after string2 in alphabetical order

Library Functions for Working with C-Strings

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

int main()
{
    // Two arrays to hold two strings.
    const int NAME_LENGTH = 30;
    char name1[NAME_LENGTH], name2[NAME_LENGTH];

    // Read two strings.
    cout << "Enter a name (last name first): ";
    cin.getline(name1, NAME_LENGTH);
    cout << "Enter another name: ";
    cin.getline(name2, NAME_LENGTH);

    // Print the two strings in alphabetical order.
    cout << "Here are the names sorted alphabetically:\n";
    if (strcmp(name1, name2) < 0)
        cout << name1 << endl << name2 << endl;
    else if (strcmp(name1, name2) > 0)
        cout << name2 << endl << name1 << endl;
    else
        cout << "You entered the same name twice!\n";

    return 0;
}
```

The End