Strings in C++

Dr. Anwar Shah, PhD, MBA(HR)
Assistant Professor in CS
FAST National University of Computer and Emerging Sciences CFD

Section Overview

Characters and Strings

- Character functions
- C-style Strings
- Working with C-style Strings
- C++ Strings
- Working with C++ Strings

Character Functions

Character Functions

```
#include <cctype>
```

```
#include <cctype>

function_name(char)
```

- Functions for testing characters
- Functions for converting character case

Character Functions

Testing characters

`isalpha(c)`	True if c is a letter
`isalnum(c)`	True if c is a letter or digit
`isdigit(c)`	True if c is a digit
`islower(c)`	True if c is lowercase letter
`isprint(c)`	True if c is a printable character
`ispunct(c)`	True if c is a punctuation character
`isupper(c)`	True if c is an uppercase letter
`isspace(c)`	True if c is whitespace

Character Functions

Converting characters

`tolower(c)`	returns lowercase of c
`toupper(c)`	returns uppercase of c

C Style Strings

C-style Strings

- Sequence of characters
  - contiguous in memory
  - implemented as an array of characters
  - terminated by a null character (null)
    - null – character with a value of zero
  - Referred to as zero or null terminated strings

- String literal
  - sequence of characters in double quotes, e.g. “Frank”
  - constant
  - terminated with null character

- C++ Literals

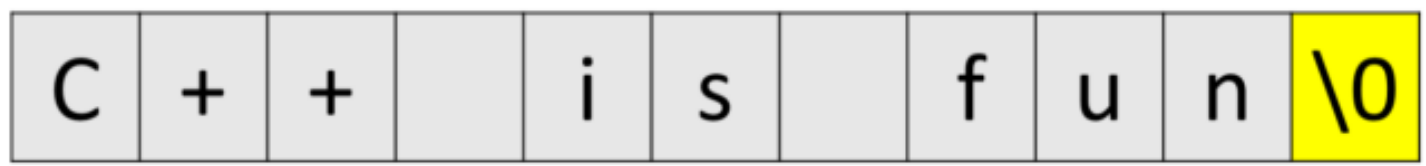
  Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

- To read more about Literals

https://www.programiz.com/cpp-programming/variables-literals#:~:text=C%2B%2B%20Literals,and%20'c'%20are%20literals.

C-style Strings

"C++ is fun"



C-style Strings

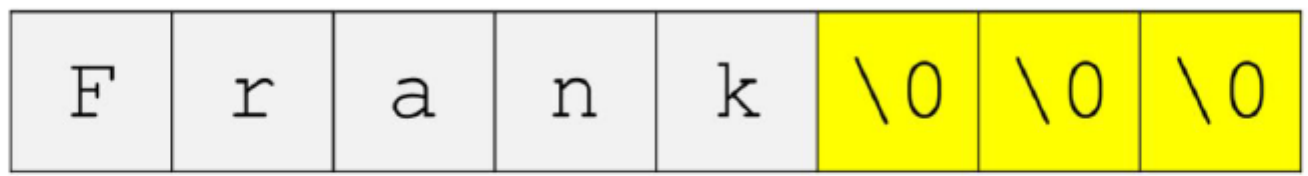
declaring variables

```
char my_name[] {"Frank"};
```

C-style Strings

declaring variables

```
char my_name[8] {"Frank"};
```

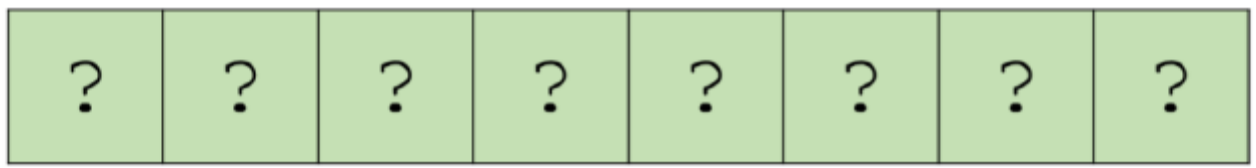



```
my_name[5] = 'y'; // OK
```

C-style Strings

declaring variables

```
char my_name[8];
```




```
my_name = "Frank";          // Error

strcpy(my_name, "Frank"); // OK
```

Working with C Style Strings

#include <cstring>

Functions that work with C-style Strings

- Copying
- Concatenation
- Comparison
- Searching
- and others

#include <cstring>

A few examples

```
char str[80];

strcpy(str, "Hello "); // copy

strcat(str, "there");  // concatenate

cout << strlen(str);   // 11

strcmp(str, "Another"); // > 0
```

#include <cstdlib>

General purpose functions

- Includes functions to convert C-style Strings to
  - integer
  - float
  - long
  - etc.

C++ Strings

C++ Strings

- std::string is a Class in the Standard Template Library
  - #include <string>
  - std namespace
  - contiguous in memory
  - dynamic size
  - work with input and output streams
  - lots of useful member functions
  - our familiar operators can be used (+, = , < , <=, >, >=, +=, ==, !=, []...)
  - can be easily converted to C-style Strings if needed
  - safer

C++ Strings

Declaring and initializing

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

string s1;                  // Empty
string s2 {"Frank"};        // Frank
string s3 {s2};             // Frank
string s4 {"Frank", 3};     // Fra
string s5 {s3, 0, 2};       // Fr
string s6 (3, 'X');         // XXX
```

Working with C++ Strings

C++ Strings

Assignment =

```
string s1;
s1 = "C++ Rocks!";

string s2 {"Hello"};
s2 = s1;
```

C++ Strings

Concatenation

```
string part1 {"C++"};
string part2 {"is a powerful"};

string sentence;

sentence = part1 + " " + part2 + " language";
        // C++ is a powerful language

sentence = "C++" + " is powerful"; // Illegal
```

C++ Strings

Accessing characters [] and at() method

```
string s1;
string s2 {"Frank"};

cout << s2[0] << endl;    // F
cout << s2.at(0) << endl; // F

s2[0] = 'f';      // frank
s2.at(0) = 'p';   // prank
```

C++ Strings

Accessing characters [] and at() method

```
string s1 {"Frank"};

for (char c: s1)
     cout << c << endl;
F
r
a
n
k
```

C++ Strings

Accessing characters [] and at() method

```
string s1 {"Frank"};

for (int c: s1)
     cout << c << endl;

70   // F
114  // r
97   // a
110  // n
107  // k
0    // null character
```

C++ Strings

Comparing

`==` `!=` `>` `>=` `<` `<=`

The objects are compared character by character lexically.

Can compare:

- two `std::string` objects
- `std::string` object and C-syle string literal
- `std::string` object and C-style string variable

C++ Strings

Comparing

```
string s1 {"Apple"};
string s2 {"Banana"};
string s3 {"Kiwi"};
string s4 {"apple"};
string s5 {s1};              // Apple

s1 == s5              // True
s1 == s2              // False
s1 != s2              // True
s1 < s2               // True
s2 > s1               // True
s4 < s5               // False
s1 == "Apple";        // True
```

C++ Strings

Substrings – `substr()`

Extracts a substring from a std::string

```
object.substr(start_index, length)
```

```
string s1 {"This is a test"};

cout << s1.substr(0,4); // This
cout << s1.substr(5,2); // is
cout << s1.substr(10,4); // test
```

C++ Strings

Searching – `find()`

Returns the index of a substring in a std::string

```
object.find(search_string)
```

```
string s1 {"This is a test"};

cout << s1.find("This");  // 0
cout << s1.find("is");  // 2
cout << s1.find("test");  // 10
cout << s1.find('e');   // 11
cout << s1.find("is", 4);   // 5
cout << s1.find("XX");  // string::npos
```

C++ Strings

Removing characters – `erase()` and `clear()`

Removes a substring of characters from a std::string

```
object.erase(start_index, length)
```

```
string s1 {"This is a test"};

cout << s1.erase(0,5);  // is a test
cout << s1.erase(5,4);  // is a
s1.clear();              // empties string s1
```

C++ Strings

Other useful methods

```
string s1 {"Frank"};

cout << s1.length() << endl; // 5

s1 += " James";
cout << s1 << endl;     // Frank James
```

Many more...

C++ Strings

Input >> and `getline()`

Reading std::string from cin

```
string s1;
cin >> s1;           // Hello there
              // Only accepts up to the first space
cout << s1 << endl;   // Hello

getline(cin, s1);   // Read entire line until \n
cout << s1 << end;    // Hello there

getline(cin, s1, 'x');  // this isx
cout << s1 << endl;   // this is
```

C String Vs C++ String

C-style strings are implemented as arrays of characters terminated by a null character ('\0'). They are commonly used in C programming and require a fixed size to be allocated for the string in advance. C-style strings do not have any built-in functionality for manipulating or storing strings.

C++-style strings, on the other hand, are implemented as an object of the std::string class, which provides a rich set of member functions for string manipulation. C++ strings are dynamically allocated and can be resized as needed during runtime, making them more flexible than C-style strings. C++ strings are also null-terminated, but the null character is automatically added and managed by the std::string class.

C String Vs C++ String

Some key differences between C-style and C++-style strings are:

1. C-style strings are implemented as character arrays, while C++-style strings are implemented as objects of the std::string class.
2. C-style strings are null-terminated, while C++ strings are null-terminated automatically by the std::string class.
3. C-style strings require a fixed size to be allocated in advance, while C++ strings can be dynamically resized during runtime.
4. C++ strings provide a rich set of member functions for string manipulation, while C-style strings do not have any built-in functionality for string manipulation.
5. C++ strings can be easily passed as arguments to functions, while C-style strings require the length of the string to be passed as a separate argument.

C String Vs C++ String

in C++, you can define a string as follows:
std::string str = "hello";

C-style string containing the word "hello" would be declared as follows:
char str[] = "hello";
char str[6] = "hello";

Remember below is a character array
char str[6] = {'h', 'e', 'l', 'l', 'o', '\0'};

Important Websites to visit

https://www.programiz.com/cpp-programming

https://www.javatpoint.com/cpp-tutorial

https://www.w3schools.com/cpp/