Introdução à Programação 2020/2021

António J. R. Neves
Daniel Corujo

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

an@ua.pt / dcorujo@ua.pt

http://elearning.ua.pt/

Topics

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

- String literals
- Strings

- A string literal consists of a sequence of characters enclosed in double quotation marks.
- Example: **`"Hello world!\n"`**
- The representation of individual characters of a string literal follows the same rules described for the values of characters in character constants.
- The double quotation mark **`"`**, the backslash **`\`**, and the new line character (**`\n`**), must be represented by escape sequences.

- C-strings are arrays of type `char` and we can use them also in C++ programs - also known as *null-terminated strings - arrays of characters terminated by a special null character.*

```
char str[32];
cout << "Enter a string: ";
cin >> str;
cout << "You entered: " << str << endl;
```

- The C standard library provides numerous functions to perform basic operations on strings such as comparing, copying, and concatenating them.
- The header file **`string.h`** declares the string manipulation functions, as well as one variable type and one macro:
  - the type is **`size_t`**. This is an unsigned integer type. It is the type of the result returned by several functions and the type of the result given by the **sizeof** operator. In theory, array indices are of this type.
  - the macro is **`NULL`**. This macro is the value of a null pointer constant.
- The names of the string functions begin with **`str`**, as in **`strcpy()`**, for example.

- The C++ strings library includes support for three general types of strings:
  - `std::basic_string` - a template class designed to manipulate strings of any character type
  - `std::basic_string_view` (C++17) - a lightweight non-owning read-only view into a subsequence of a string
  - Null-terminated strings - arrays of characters terminated by a special null character
- Standard C++ includes a class called `string` that is an instantiation of the `std::basic_string` template with a type of `char` and the one we will use in this course
- This class improves on the traditional C- string in many ways:
  - no longer need to worry about creating an array of the right size to hold string variables
  - the string class assumes all the responsibility for memory management
  - the string class allows the use of overloaded operators (ex. concatenate string objects with the + operator

- We can define a string object in several way
  - use a constructor with no arguments, creating an empty string
  - use a one-argument constructor where the argument is a C-string constant

```
string s1("Man");
string s2 = "Beast"; // copy assignment operator
string s3;           // call default construct
```

- The string class uses a number of overloaded operators:

```
s3 = "Neither " + s1 + " nor "; // "Neither Man nor "
s3 += s2; // "Neither Man nor Beast"
```

- Input and output are handled with `cin` and `cout` streams
- The `<<` and `>>` operators are overloaded to handle string objects
- A function `getline()` handles input that contains embedded blanks or multiple lines

```
string full_name, address;
string greeting("Hello, ");
cout << "Enter your full name: ";
getline(cin, full_name);
cout << "Your full name is: " << full_name << endl;
cout << "Enter your address on separate lines\n";
cout << "Terminate with '$'\n";
getline(cin, address, '$');
```

- The `string` class includes a variety of member functions for finding strings and substrings in string objects
  - The `find()` function looks for the string used as its argument in the string for which it was called
  - The `find_first_of()` function looks for any of a group of characters, and returns the position of the first one it finds
  - `find_first_not_of()` finds the first character in its string that is not one of a specified group
  - `rfind()`, `find_last_of()`, `find_last_not_of()`

```
string s1 = "“In Xanadu did Kubla Kahn...";
n = s1.find(“Kubla”);
cout << “Found Kubla at “ << n << endl;
n = s1.find_first_of(“spde”);
n = s1.find_first_not_of(“aeiouAEIOU”);
```

- There are various ways to modify string objects:
  - The `erase()` function removes a substring from a string
  - The `replace()` function replaces part of the string with another string
  - The `insert()` function inserts the string specified by its second argument at the location specified by its first argument
  - The `append()` function appends additional characters to the string
- Replace multiple instances of a substring with another string:

```
size_t x = s1.find(' ');
while( x < s1.size() ){
   s1.replace(x, 1, "/");
   x = s1.find(' ');
}
```

- We can use overloaded operators or the `compare()` function to compare string objects
- These discover whether strings are the same, or whether they precede or follow one another alphabetically
- `int compare (size_t pos, size_t len, const string& str, size_t subpos, size_t sublen = npos)`

Example:

```
string aName = "George";
string userName;
cin >> userName;
if(userName == aName)
    cout << "Greetings, George\n";
else if(userName < aName)
    cout << "You come before\n";
else
    cout << "You come after\n";

int n = userName.compare(0, 2, aName, 0, 2); // 0, <0, >0
```

- You can access individual characters within a string object using the `at()` member function or the overloaded `[]` operator, which makes the string object look like an array
- It's safer to use the `at()` function, which causes the program to stop if you use an out-of-bounds index

```
string word;
cout << "Enter a word: ";
cin >> word;
cout << "One character at a time: ";
for(int j=0; j<word.length(); j++)
    cout << word.at(j);
    //cout << word[j]; // using []
```

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

- Executes a `for` loop over a range
- Used as a more readable equivalent to the traditional `for` loop operating over a range of values, such as all elements in a container
- https://en.cppreference.com/w/cpp/language/range-for

```
string word = "ola";
for(char c : word)
    cout << c;

int a[3] = {10, 20, 30};
for(int i : a)
    cout << i;
```