Strings

CSE 232 - Dr. Josh Nahum

Reading:

Section 10.2

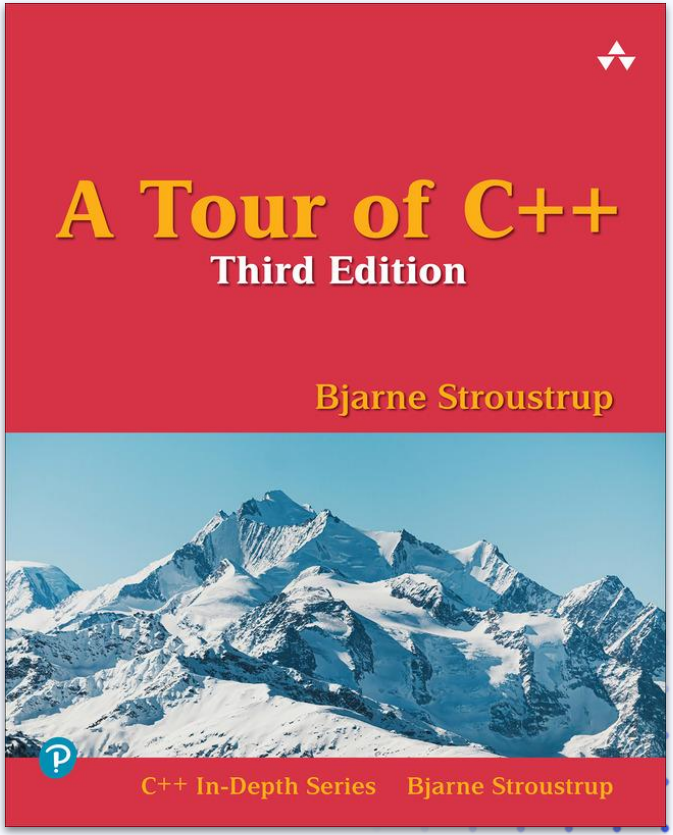

Table of contents

OO

Char

Strings are basically vectors of char

Same methods

.at(), .size(), range-based for loops, all work the same

String Literals

C-style string literals (ex. "abc") can be easily cast to a std::string during initialization:

```
std::string name{"Josh"};
```

Indexing

The element returned by indexing is a `char`.

```
'J' == name.at(0);
```

Useful char functions

```
#include <iostream>;
#include <cctype>;

int main() {
  char c{' '};
  if (std::isalpha(c)) {
    std::cout << c << " is in the alphabet";
  }
}
```

isalnum	checks if a character is alphanumeric (function)
isalpha	checks if a character is alphabetic (function)
islower	checks if a character is lowercase (function)
isupper	checks if a character is an uppercase character (function)
isdigit	checks if a character is a digit (function)
isxdigit	checks if a character is a hexadecimal character (function)
iscntrl	checks if a character is a control character (function)
isgraph	checks if a character is a graphical character (function)
isspace	checks if a character is a space character (function)
isblank (C++11)	checks if a character is a blank character (function)
isprint	checks if a character is a printing character (function)
ispunct	checks if a character is a punctuation character (function)
tolower	converts a character to lowercase (function)
toupper	converts a character to uppercase (function)

01

String Methods

Initialization

Empty

```
string s;
```

Unlike fundamental types, declaring a string initializes it to be empty, not undefined.

Copy

```
string s{"Hi"};
string t{s};
```

You can initialize a string with a C-style string literal, or another instance of a string.

List of Char

```
string pet{'M', 'a', 'l'};
```

You can initialize a string with a list of chars (similar to a vector).

More Member Functions

operator=

```
// Assignment
name = "Josh";
name.at(0) = 'j';
```

push_back

```
name.push_back('!');
```

Repeated Constructor

```
string five(5, 'a');
five == "aaaaa";
```

clear

```
name.clear();
// Erases string
```

operator<<

```
cout << name;
```

Many more

See: https://en.cppreference.com/w/cpp/string/basic_string

String Comparisons

operator==

`str1 == str2;`
Only true if strings are the same length and have the same sequence of characters.

<, >, <=, >=, !=

String comparisons are performed lexicographically (using the ASCII table). For strings composed of just same-case letters, this is alphabetic order.

All True

```
"abc" == "abc";
"ab" != "abc";
"ab" < "abc";
"cat" < "dog";
"DOG" < "cat";
"DOG" <= "cat";
```

02
Find Member Function

std::string::size_type

Many member functions of strings and vectors return unsigned integers instead of int. In the documentation you will see the type called `std::string::size_type` or `size_t`.

Be careful not to accidentally treat this value as an int. You can cast it to an int if needed.

`std::string::npos`

`npos` (no position) is the largest possible unsigned value, and it is used to indicate that the value returned is not a valid index. When cast to a signed int, its value is -1.

std::string::find

```
string my_str = “hello world”;
size_t pos{my_str.find(‘e’)};
// pos gets set to 1
// doesn’t exist?
// return string::npos
```

```
// Check string for space character
std::string::size_type pos{
  my_string.find(' ')};
bool has_space{pos != std::string::npos};
// has_space is true if space in string
```

03

String Input

operator>>

```
string my_str;
cin >> my_str;
// ignores leading whitespace
// reads characters into my_str until
// whitespace or EOF
```

getline function

```
// getline function is useful for reading in lines with whitespace
string my_str;
getline(cin, my_str);
// reads characters into my_str until
// newline character or EOF
getline(cin, my_str, '.');
// reads characters into my_str until
// period character or EOF
```

Attribution

Please ask questions via Piazza

Dr. Joshua Nahum
www.nahum.us
EB 3504

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**