Strings

Amrita Kaur

June 28, 2023

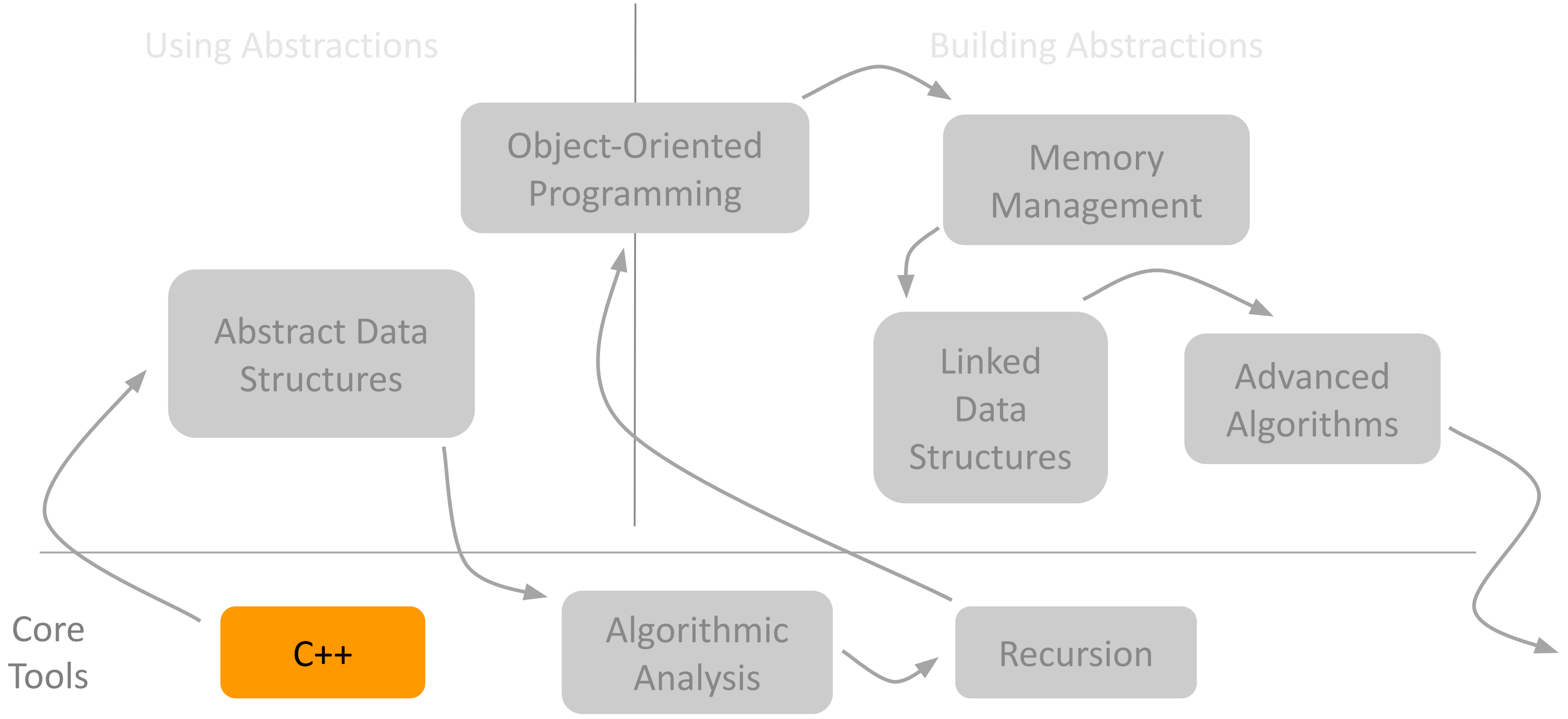
Contributions made from previous CS106B Instructors

Stanford University

Announcements and Reminders

- Sections start today!
  - Should have already gotten section assignment in an email
  - Change section on section signup page
- Assignment 0 due Friday at 11:59pm
- Please always email **both** Elyse and Amrita when reaching out
  - You'll get a faster response that way
- OH: Today 3-5pm in Durand 303

CS106B Roadmap



C++ Types

Numbers

- `int`, `long` `// 100`
- `float`, `double` `// 3.14`

Text

- `char`, `string` `// ‘a’, “apple”`

Booleans

- `bool` `// true, false`

String

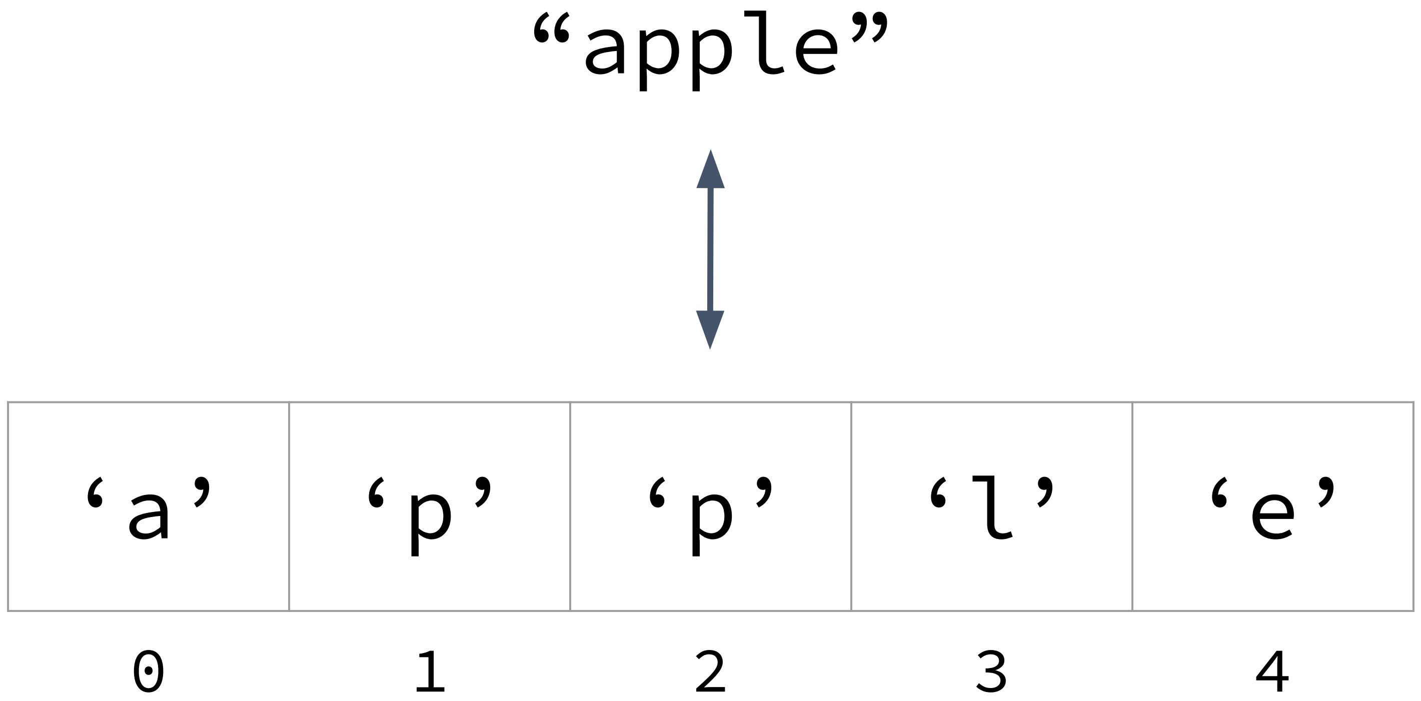
- Data type that represents a sequence of characters
- Marked by double quotes
- Ex: “`apple`”

Char

- Data type that represents a single character (letters, digits, symbols)
- Marked by single quotes
- Ex: ‘a’
- Have numerical representation (ASCII codes)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	**@**	96	60	**`**
1	1	*[START OF HEADING]*	33	21	**!**	65	41	**A**	97	61	**a**
2	2	*[START OF TEXT]*	34	22	**"**	66	42	**B**	98	62	**b**
3	3	*[END OF TEXT]*	35	23	**#**	67	43	**C**	99	63	**c**
4	4	*[END OF TRANSMISSION]*	36	24	**$**	68	44	**D**	100	64	**d**
5	5	*[ENQUIRY]*	37	25	**%**	69	45	**E**	101	65	**e**
6	6	*[ACKNOWLEDGE]*	38	26	**&**	70	46	**F**	102	66	**f**
7	7	*[BELL]*	39	27	**'**	71	47	**G**	103	67	**g**
8	8	*[BACKSPACE]*	40	28	**(**	72	48	**H**	104	68	**h**
9	9	*[HORIZONTAL TAB]*	41	29	**)**	73	49	**I**	105	69	**i**
10	A	*[LINE FEED]*	42	2A	*****	74	4A	**J**	106	6A	**j**
11	B	*[VERTICAL TAB]*	43	2B	**+**	75	4B	**K**	107	6B	**k**
12	C	*[FORM FEED]*	44	2C	**,**	76	4C	**L**	108	6C	**l**
13	D	*[CARRIAGE RETURN]*	45	2D	**-**	77	4D	**M**	109	6D	**m**
14	E	*[SHIFT OUT]*	46	2E	**.**	78	4E	**N**	110	6E	**n**
15	F	*[SHIFT IN]*	47	2F	**/**	79	4F	**O**	111	6F	**o**
16	10	*[DATA LINK ESCAPE]*	48	30	**0**	80	50	**P**	112	70	**p**
17	11	*[DEVICE CONTROL 1]*	49	31	**1**	81	51	**Q**	113	71	**q**
18	12	*[DEVICE CONTROL 2]*	50	32	**2**	82	52	**R**	114	72	**r**
19	13	*[DEVICE CONTROL 3]*	51	33	**3**	83	53	**S**	115	73	**s**
20	14	*[DEVICE CONTROL 4]*	52	34	**4**	84	54	**T**	116	74	**t**
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	**5**	85	55	**U**	117	75	**u**
22	16	*[SYNCHRONOUS IDLE]*	54	36	**6**	86	56	**V**	118	76	**v**
23	17	*[END OF TRANS. BLOCK]*	55	37	**7**	87	57	**W**	119	77	**w**
24	18	*[CANCEL]*	56	38	**8**	88	58	**X**	120	78	**x**
25	19	*[END OF MEDIUM]*	57	39	**9**	89	59	**Y**	121	79	**y**
26	1A	*[SUBSTITUTE]*	58	3A	**:**	90	5A	**Z**	122	7A	**z**
27	1B	*[ESCAPE]*	59	3B	**;**	91	5B	**[**	123	7B	**{**
28	1C	*[FILE SEPARATOR]*	60	3C	**<**	92	5C	\	124	7C	**|**
29	1D	*[GROUP SEPARATOR]*	61	3D	**=**	93	5D	**]**	125	7D	**}**
30	1E	*[RECORD SEPARATOR]*	62	3E	**>**	94	5E	**^**	126	7E	**~**
31	1F	*[UNIT SEPARATOR]*	63	3F	**?**	95	5F	**_**	127	7F	*[DEL]*


“apple”
‘a’
‘p’
‘p’
‘l’
‘e’
0
1
2
3
4

Indexing into a String

```
void printLetter() {
   string word = “apple”;

   // TODO: print out the letter ‘l’ from string
   // 1. How do we index into array?
   // 2. Do you remember how to print in C++?
}
```

Looping Through a String

```
string word = “apple”;
for (int i=0; i < word.length(); i++) {
   cout << word[i] << endl;
}
// OR
for (char letter : word) {
   cout << letter << endl;
}
```

Apple, Reimagined

```
// What is the output of this function?

void changeLetter() {
   string word = “apple”;
   word[1] = ‘q’;
   cout << word << endl;
}
```

Key Characteristics of Strings

- Strings are mutable in C++

Apple, Reimagined (Take 2)

```
// What is the output of this function?

void change
    string w
    word[1]                                 o string
    cout << word << endl;
}
```

```
error: assigning to
'__gnu_cxx::__alloc_traits<std::allo
cator<char>, char>::value_type' (aka
'char') from incompatible type
'const char [2]'
```

Key Characteristics of Strings

- Mutable in C++
- Concatenated using + or +=

More Apples, Please

```
void addLetter() {
    string word = “apple”;
    string letterStr = “s”;

    word = word + letterStr;
    cout << word << endl;
}
```

```
apples
```

More Apples, Please

```
void addLetter() {
    string word = “apple”;
    string letterStr = “s”;

    word += letterStr;
    cout << word << endl;
}
```

```
apples
```

Even More Apples, Please

```
void addLetter() {
    string word = “apple”;
    string letterStr = “s”;

    word += letterStr;
    cout << word << endl;
}
```

```
void addLetterChr() {
    string word = “apple”;
    char letterChr = ‘s’;

    word += letterChr;
    cout << word << endl;
}
```

```
apples
```

Key Characteristics of Strings

- Mutable in C++
- Concatenated using + or +=
  - Add strings and strings, output is a string
  - Add strings with chars, output is a string
  - Adding chars will NOT give a string output

Key Characteristics of Strings

- Mutable in C++
- Concatenated using + or +=
    - Add strings and strings, output is a string
    - Add strings with chars, output is a string
    - Adding chars will NOT give a string output
- Compared using relational operators (<, >, ==, !=)

Apples and Operators

```
void compareStringsV1() {
    string s1 = “apple”;
    string s2 = “banana”;
    if (s1 < s2) {
        cout << s1 << “ < ” << s2 << endl;
    } else {
        cout << s1 << “ > ” << s2 << endl;
    }
}
```

```
apple < banana
```

Apples and Operators

```
void compareStringsV2() {
    string s1 = “apple”;
    string s2 = “Banana”;
    if (s1 < s2) {
        cout << s1 << “ < ” << s2 << endl;
    } else {
        cout << s1 << “ > ” << s2 << endl;
    }
}
```

```
Banana < apple
```

Apples and Operators

```
void compareStringsV3() {
    string s1 = “apple”;
    string s2 = “apples”;
    if (s1 < s2) {
        cout << s1 << “ < ” << s2 << endl;
    } else {
        cout << s1 << “ > ” << s2 << endl;
    }
}
```

```
apple < apples
```

Libraries

- Allow us to use code that was written elsewhere by someone else
- Standard C++ Libraries

```
#include <libraryname>
```

- Local Libraries

```
#include "libraryname.h"
```

Libraries for Strings and Chars

- `<cctype>` library
  - Built-in C++ char methods
- `<string>` library
  - Built-in C++ string methods
- `“strlib.h”` library
  - Stanford string functions

`<cctype>` Library

- **`#include <cctype>`**
- This library provides functions that check a single **`char`** for a property (e..g, if it is a digit), or return a **`char`** converted in some way (e.g., to uppercase)
  - **`isalnum`**: checks if a character is alphanumeric
  - **`isalpha`**: checks if a character is alphabetic
  - **`islower`**: checks if a character is lowercase
  - **`isupper`**: checks if a character is an uppercase character
  - **`isdigit`**: checks if a character is a digit
  - **`isxdigit`**: checks if a character is a hexadecimal character
  - **`iscntrl`**: checks if a character is a control character
  - **`isgraph`**: checks if a character is a graphical character
  - **`isspace`**: checks if a character is a space character
  - **`isblank`**: checks if a character is a blank character
  - **`isprint`**: checks if a character is a printing character
  - **`ispunct`**: checks if a character is a punctuation character
  - **`tolower`**: converts a character to lowercase
  - **`toupper`**: converts a character to uppercase

`<string>` Library

- **`#include <string>`**
  - **`s.append(str)`**: add text to the end of a string
  - **`s.compare(str)`**: return **`-1`**, **`0`**, or **`1`** depending on relative ordering
  - **`s.erase(index, length)`** : delete text from a string starting at given index
  - **`s.find(str)`**
    `s.rfind(str)`: first or last index where the start of **`str`** appears in this string (returns **`string::npos`** if not found)
  - **`s.insert(index, str)`**: add text into a string at a given index
  - **`s.length() or s.size()`**: number of characters in this string
  - **`s.replace(index, len, str)`**: replaces **`len`** chars at given index with new text
  - **`s.substr(start, length) or s.substr(start)`**: the next length characters beginning at **`start`** (inclusive); if length omitted, grabs till end of string

“strlib.h” Library

- **`#include “strlib.h”`**
  - **`endsWith(str, suffix)`**
    `startsWith(str, prefix)`: returns **`true`** if the given string begins or ends with the given prefix/suffix text
  - **`integerToString(int)`**
    `realToString(double)`
    `stringToInteger(str)`
    `stringToReal(str)`: returns a conversion between numbers and strings
  - **`equalsIgnoreCase(s1, s2)`**: **`true`** if **`s1`** and **`s2`** have same **`chars`**, ignoring casing
  - **`toLowerCase(str)`**
    `toUpperCase(str)`: returns an upper/lowercase version of a string
  - **`trim(str)`**: returns string with surrounding whitespace removed

Review

```
void addLetter() {
    string word = “apple” + “sauce”;
    cout << word << endl;
}
```

```
??
```

Not a Review! This is new…

```
void addLetter() {
    string word = “apple” + “sauce”;
    cout << word << endl;
}
```

```
void addLetter() {
    string word = “apple”;
    string letterStr = “s”;

    word = word + letterStr;
    cout << word << endl;
}
```

```
error: invalid operands to binary
expression ('const char [6]' and
'const char [2]')
```

C++ vs. C strings

- C strings, also known as string literals
  - Hard coded string values
  - Ex: “hi there”
  - Have no methods
  - Deal with memory management on your own - dangerous!
- C++ strings, which are string objects
  - Ex: string s1 = “hi there”;
  - Lots of helpful methods!
- When possible, declare C++ strings for better usability

Conversion

- You can convert between string types:
  - `string s = “text”;` converts a C string literal into a C++ string
  - `string("text");` converts C string literal into C++ string
  - `string.c_str()` returns a C string out of a C++ string

C string examples

```
string word = “apple” + “sauce”;
```

- Concatenating C strings with +
- Not possible (does not compile)

```
string word1 = “apple”
string word = word1 + “sauce”;
```

- Concatenating C++ and C string with +
- Works perfectly! (autoconversion of C string)

C string examples

```
string hiQuestion = “hi” + ‘?’;
```

- Concatenating C string and char with +
- Not possible (produces garbage - particularly nefarious)

```
string hiQuestion = string(“hi”) + ‘?’;
```

- Concatenating C++ string and char with +
- Works perfectly! (autoconversion of char)

What is the output?

```
void mystery(string a, string &b) {
    a.erase(0, 1);
    b += a[0];
    b.insert(3, "FOO");
}

int main() {
    string a = "Stanford";
    string b = "Tree";
    mystery(a, b);
    cout << a << " " << b << endl;
    return 0;
}
```

Recap

- Strings - double quoted, sequences of chars
- Chars - single quoted, single-character ASCII numerical values
- Key characteristics of strings
  - Mutable in C++
  - Concatenate C++ strings with +
    - Adding chars can get weird
  - Compare with logical operators
- Standard and Stanford-specific libraries that provide helpful string functions
- C++ has both C strings and C++ strings
  - under the hood, both are sequences of characters
  - C++ strings handle details for you automatically, C-strings do not.
  - C++ strings are much more functional and easier to use
  - Many times (but not always), C-strings auto-convert to C++ strings when necessary