Lecture 15: Strings

Announcements and reminders

Submissions:

- HW 5 -- due Saturday at 6 PM

Practicum 1 -- Wednesday 5:30 - 7 PM (staggered start, don't be late, nor alarmed)

- Practicum 1 room assignments posted to Piazza:
  - 301, 303 -- ECCR 265
  - 302 -- ECCR 200
  - 304 -- ECCR 1B40
- Practice problems on Moodle -- **DO THEM. They are excellent practice problems for the practicum. That's why we call them "practice problems"**

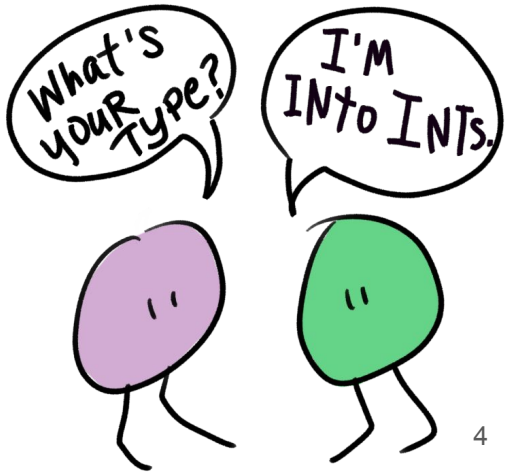
Last time on *Intro Computing…*

- We learned about **`for`** loops and **`do... while`** loops!
  - … how to repeat a set of instructions a **fixed number of times** (_____ loops)
  - … or how to do a while loop that **checks the condition at the end, after running at least once** (_____ loops)

Chapter 2: Fundamental Data Types

Chapter topics:

- Variables
- Arithmetic
- Input and output
- Problem solving: first do it by hand
- **Strings**



Strings

Strings are sequences of characters. For example: “Hello world!”

We need to include the <string> header file, so we can create variables to hold strings:

```
#include <iostream>
#include <string>
using namespace std;
...
string my_name = “Tony”;       // literal string “Tony” is stored in this variable.
cout << “Your name is: ” << my_name << endl;
```

Strings -- initialization

Strings are automatically initialized to the empty string if you don't initialize them:

```
#include <iostream>
#include <string>
using namespace std;
...
string my_name;                    // no initialization this time! Let's see what happens…
cout << "Your name is: " << my_name << endl;
```

Definition: "" is called the **empty** or the **null string**

Strings -- concatenation

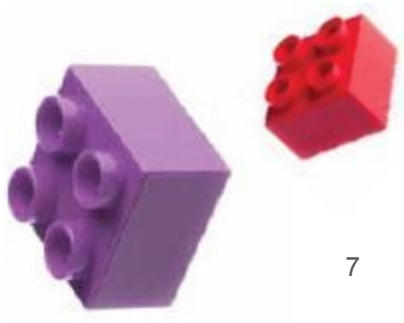
Definition: Combining strings to yield a longer string is called **concatenation**.

You can use the + operator to **concatenate** strings:

```
string fname = "Harry";
string lname = "Potter";

string fullname_wrong = fname + lname;
cout << fullname_wrong << endl;

string fullname_right = fname + " " + lname;
cout << fullname_right << endl;
```



Common error: literal strings

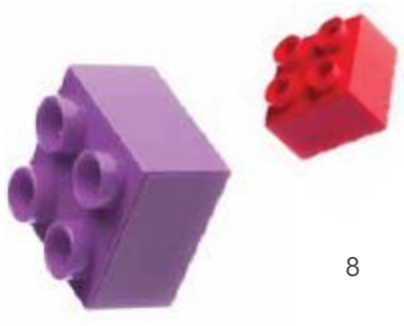
Literal strings (as opposed to variables storing strings) cannot be concatenated

```
string greeting = "Hello, " + " World!";
```

But why in the hello world would you do that in the first place?

→ Instead, just do it directly:

```
string greeting = "Hello, World!";
```



String Input

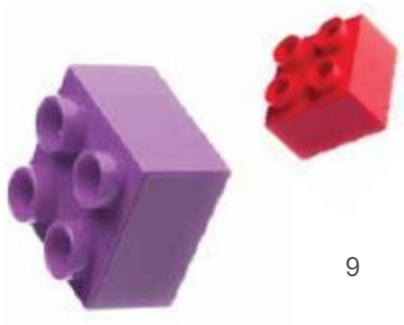
You can read a string from console input (cin):

```
cout << "Please enter your name: ";
string name;
cin >> name;
```

Warning! When a string is read in with the >> operator, only **one word is placed into the string variable**

→ If the user enters: `Harry Potter`

Then only the string "`Harry`" will be placed into the variable `name`



String Input

You can read a string from console input (cin):

```
cout << "Please enter your name: ";
string name;
cin >> name;
```

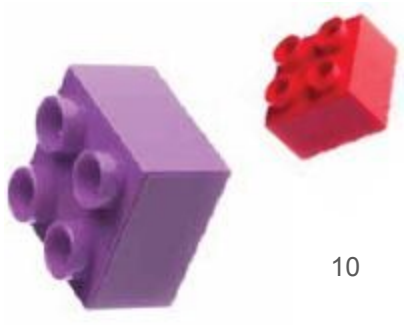
Warning! When a string is read in with the >> operator, only **one word is placed into the string variable**

→ If the user enters: `Harry Potter`

Then only the string "Harry" will be placed into the variable `name`

You can use a second input string to read the second word:

```
cout << "Please enter your name (first, then last): ";
string fname, lname;
cin >> fname >> lname;
```



Strings Functions

Definition: A **member function** is a type of function that is defined specifically for objects of a certain class.

The **length** member function for string variables yields the number of characters in a string.

Unlike sqrt or pow functions, the length function is ***invoked*** using the ***dot notation***:

```
string name = "Harry";
int n = name.length();
```

Strings Functions

Definition: The **substr** member function extracts **substrings** from a string variable (smaller pieces of the original string)

```
s.substr(start, length)
```

… returns a string that is made from the characters in the string `s`,
    … starting at character # `start`, and
    … containing `length` characters total (`start` and `length` are both integers)

```
string greeting = "Hello, World!";
string sub = greeting.substr(0, 2);
```

→ What will `sub` contain?

Strings Functions

Example: What will sub contain?

```
string greeting = "Hello, World!";
string sub = greeting.substr(7, 5);
```

sub = "World"

- "World" is 5 characters long
- 7 is the position of the "W"
- … why isn't it 8??

  → because the first character has index of 0, not 1

  → and note that the comma and space both count as characters

Strings Data Representation and Character Positions

```
H   e   l   l   o   ,       W   o   r   l   d   !

0   1   2   3   4   5   6   7   8   9   10  11  12
```

- In most computer languages, starting position of 0 means “start at the beginning” (including C++)
- The first position in a string is labeled 0, the second 1, and so on
- Spaces, commas, and other symbols are characters too…
- … but the quotation marks used to define the string are **not**

Strings Data Representation and Character Positions

```
H   e   l   l   o   ,       W   o   r   l   d   !
0   1   2   3   4   5   6   7   8   9   10  11  12
```

- In most computer languages, starting position of 0 means “start at the beginning” (including C++)
- The first position in a string is labeled 0, the second 1, and so on
- Spaces, commas, and other symbols are characters too…
- … but the quotation marks used to define the string are **not**
- **Handy Rule:** The index of the last character is always one less than the length of the string

Strings Data Representation and Character Positions

H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

```
string greeting = "Hello, World!";
string sub = greeting.substr(7);
```

sub = "World!"

- **Another Rule:**
  If you do not specify the length of your desired substring, substr() will give you from start to the end.

Some string operation examples

statement	result	comment
string str = "C"; str = str + "++";		
string str = "C" + "++";		
cout << "Enter name: "; cin >> name; (UserInput: Harry Morgan)		
cout << "Enter name: "; cin >> name >> last_name; (UserInput: Harry Morgan)		
string greeting = "H & S"; int n = greeting.length()		

Some string operation examples

statement	result	comment
`string str = "C";` `str = str + "++";`	`str` set to "C++"	For strings, + denotes concatenation
`string str = "C" + "++";`	**Error**	Can't concatenate two string literals

`cout << "Enter name: ";`
`cin >> name;`
`(UserInput: Harry Morgan)`
	`name` contains "Harry"	>> operator only places next **word**; the space truncates the input

`cout << "Enter name: ";`
`cin >> name >> last_name;`
`(UserInput: Harry Morgan)`
	`name` contains "Harry", `last_name` contains "Morgan"	Multiple >> operators can be used to read multiple words

`string greeting = "H & S";` `int n = greeting.length()`	n is set to 5	Each letter, character and space counts as one character

Some string operation examples

statement	result	comment
`string str = "Sally";` `string str2 = str.substr(1,3);`	`str2` is set to "all"	Extracts substring of length 3 starting at position 1 (initial position is 0)
`string str = "Sally";` `string str2 = str.substr(1);`	`str2` is set to "ally"	If you omit the length, **all characters** from the position to end are included
`string a = str.substr(0, 1);`	`a` is set to the first letter in `str`	Extracts the substring of length 1 starting at position 0
`string b =` `  str.substr(str.length() - 1);`	`b` is set to the last letter in `str`	The last letter has position `str.length()-1`, so no need for length

Strings Functions

Example: Add some appropriate comments before this function. What does it do? What headers are needed?

```
int main()
{
   cout <<  "Enter your first name: ";
   string first;
   cin >> first;
   cout << "Enter your significant other's first name: ";
   string second;
   cin >> second;
   string initials = first.substr(0,1) + "&" + second.substr(0,1);
   cout << initials << endl;
   return 0;
}
```

Strings Functions

Example: Add some appropriate comments before this function. What does it do? What headers are needed?

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

int main()
{
   cout <<  "Enter your first name: ";
   string first;
   cin >> first;
   cout << "Enter your significant other's first name: ";
   string second;
   cin >> second;
   string initials = first.substr(0,1) + "&" + second.substr(0,1);
   cout << initials << endl;
   return 0;
}
```

Maybe something like:

```
/* Function to print out the couples' initials
 * Asks user to enter from keyboard their
 * and their significant other's first names
 * Output to screen the couples' first initials,
 * combined with a &
 */
```

Representing Characters: Unicode and ASCII

Printable characters in a `string` are stored as bits in a computer, just like `int` and `double` variables

The bit patterns are standardized:

- **ASCII** (American Standard Code for Information Interchange) is 7 bits long
  - → can represent 2^7 = 128 different things
  - 26 uppercase letters A-Z, + 26 lowercase letters a-z
  - 10 digits
  - 32 typographical symbols like +, -, ‘, \, …
  - 34 control characters like spaces, newline, …
  - 32 others for controlling printers and other devices

Representing Characters: Unicode and ASCII

Printable characters in a `string` are stored as bits in a computer, just like `int` and `double` variables

The bit patterns are standardized:

- **Unicode** has mostly replaced ASCII, and is 21 bits long
  - ***Superset*** of ASCII → first 128 codes match
  - Extra bits allow more characters -- $2^{21} \approx 2$ million
    - Required for worldwide languages
  - About 136,000 characters have been assigned so far
  - UTF-8 is the 8-bit subset of Unicode, and UTF-16 is the 16-bit version, often used for websites and compilers

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	[ENG 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]

What just happened?

- We learned about the `string` variable type!
  - … how to input strings from keyboard (`cin`)
  - … how to take subsets of strings (`str.substr(start, length)`)
  - … how to find the length of a string (`str.length()`)