Strings

Due this week

- **Homework 3**
  - Submit pdf file on Canvas. PDF
  - Check the due date! **No late submissions!!**
- Start going through the textbook readings and watch the videos
  - Take **Quiz 3**.
  - Check the due date! **No late submissions!!**

Today

- Revisit Functions
  - Prototyping
  - Scope of Variables
- Strings

Strings

Strings

- Strings are sequences of characters:

```
"Hello world"
```

- Include the string header, so you can create variables to hold strings:

```
#include <iostream>
#include <string>
using namespace std;
...
string name = "Harry";
       // literal string "Harry" stored
```

String Initializations

- String variables are automatically initialized to the empty string if you don’t initialize them:

```
string response;
      // literal string "" stored
    // it is not garbage
```

- `""` is called the empty or null string.

Concatenation of Strings

- Use the **+** operator to *concatenate* strings;
  that is, put them together to yield a longer string.

```
string fname = "Harry";
string lname = ”Potter";
string name = fname + lname; //need a space!
cout << name << endl;
name = fname + " " + lname; //got a space
cout << name << endl;
```

The output will be:

```
       HarryPotter
       Harry Potter
```

Common Error – Concatenation of literal strings

```
string greeting = "Hello, " + " World!";
                 // will not compile
```

Literal strings cannot be concatenated. And it’s pointless anyway, just do:

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

String Input

- You can read a string from the console:

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

- When a string is read with the `>>` operator, only one word is placed into the `string` variable.
- For example, suppose the user types

```
Harry Potter
```

as the response to the prompt.

- Only the string "Harry" is placed into the variable name.

String Input

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

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


//fname gets Harry, lname gets Potter
```

String Input

`getline()` function allows us to accepts a full string input

```
cout << "Please enter your name: ";
string name;
getline(cin, name);


//name gets Harry Potter
```

String Functions

- The `length` *member function* yields the number of characters in a string.

- Unlike the `sqrt` or `pow` function, the `length` function is *invoked* with the *dot notation*:

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

String Data Representation & 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, the starting position 0 means "start at the beginning."
- The first position in a string is labeled 0, the second 1, and so on. And don't forget to count the space character after the comma—but the quotation marks are **not** stored.
- The position number of the last character is always one less than the length of the **string.**

`substr` Function

- Once you have a string, you can extract substrings by using the **`substr`** member function.
- `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. (`start` and `length` are integers)
  - NOTE: the first character has an index of 0, not 1.

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

Another Example of the `substr` Function

```
string greeting = "Hello, World!";
string w = greeting.substr(7, 5);
   // w contains "World" (not the !)
```

- `"World"` is 5 characters long but…
- Why is 7 the position of the `“W”` in `"World"`?
- Why is the `"W"` not @ 8?
- *Because the first character has an index of 0, not 1.*

String Character Positions and

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 w = greeting.substr(7);
   // w contains "World!"
```

- If you do not specify how many characters should go into the substring, the call to the **substr**() function will return a substring that starts at the specified index, and goes until the end of the string

String Operations Examples

Statement	Result	Comment

string str = "C";
str = str + "++";
	str is set to "C++"	When applied to strings,+ denotes concatenation.

string str = "C" + "++";	Error	Error: You cannot concatenate two string literals.

cout << "Enter name: ";
cin >> name;
(User input: Harry Morgan)
	name contains "Harry"	The >> operator places the next word into the string variable.

cout << "Enter name: ";
cin >> name >> last_name;
(User input: Harry Morgan)
	name contains "Harry", last_name contains "Morgan"	Use multiple >> operators to read more than one word.

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

string str = "Sally";
string str2 = str.substr(1, 3);
	str2 is set to "all"	Extracts the substring of length 3 starting at position 1. (The 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 until the end are included.

string a = str.substr(0, 1);	a is set to the initial 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. We need not specify the length.

Example: Ubbi Dubbi (Link)

Example: Ubbi Dubbi

```
string penny_says = "Absolutely I Do";
string ubbi_dubbi_word = “ubAbsubolubutubely ubI Dubo”
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	b	s	o	l	u	t	e	l	y		I		D	o

- How many substr functions did you use? How many string concatenations did you use?

Find and replace

- str.find(substring_to_find)
  - Finds a substring if present in a string
  - int position = str.find(“Waldo”); // position has the first occurrence of “Waldo” in str
- str.replace(position, length, string_to_replace)
  - Replaces the characters in str from position with string_to_replace
  - str.replace(6, 10, “Pikachu”);

Representing Characters: Unicode. 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, specifying $2^7 = 128$ codes:
    - 26 uppercase letters A through Z
    - 26 lowercase letters a through z
    - 10 digits
    - 32 typographical symbols such as +, -, ', \...
    - 34 control characters such as space, newline
- Unicode, which has replaced ASCII in most cases, is 21 bits superset of ASCII; the first 128 codes match. The extra bits allow many more characters ($2^{21} > 2 \times 10^6$), required for worldwide languages