2.15 Strings

Strings and string literals

A ***string*** is a sequence of characters. A ***string literal*** surrounds a character sequence with double quotes, as in "Hello", "52 Main St.", or "42", vs. an integer literal like 42 or character literal like 'a'. Various characters may be in a string, such as letters, numbers, spaces, or symbols like $ or %, as in "$100 for Julia!!". Earlier sections showed string literals being output, as in: `cout << "Hello";`.

PARTICIPATION ACTIVITY 2.15.1: A string is stored as a sequence of characters in memory.

Type a string to see how a string is stored as a sequence of characters in memory. In this case, the string happens to be in memory locations 501 to 506. Try typing Hello! or 627.

Type a string (up to 6 characters): Hello!

Memory

501	H
502	e
503	l
504	l
505	o
506	!

Feedback?

PARTICIPATION ACTIVITY 2.15.2: String literals.

Indicate which items are string literals.

1) "Hey"

- (•) Yes
- ( ) No

Correct

Consists of characters: H, e, y

2) "Hey there."

- (●) Yes
- ( ) No

Correct

A string literal may contain spaces, periods, and other non-letter characters.

3) 674

- ( ) Yes
- (●) No

Correct

A string literal must be surrounded by double quotes.

4) "674"

- (●) Yes
- ( ) No

Correct

While looking like a number, the double quotes make this a string literal consisting of characters: 6, 7, 4.

5) 'ok'

- ( ) Yes
- (●) No

Correct

A string literal requires double quotes as in "ok", not single quotes as in 'ok'.

6) "a"

- (●) Yes
- ( ) No

Correct

A string literal may have just one character, but is still considered a string (and not a character).

Feedback?

String variables and assignments

Some variables should hold a string. A string data type isn't built into C++ like char, int, or double, but is available in the standard library and can be used after adding: `#include <string>`. A programmer can then declare a string variable as: `string firstName;`.

A programmer can assign a string just as for other types. Ex: str1 = "Hello", or str1 = str2. The string type automatically reallocates memory for str1 if the right-side string is larger or smaller, and then copies the characters into str1.

A programmer can initialize a string variable during declaration: `string firstMonth = "January";`. Otherwise, a string variable is automatically initialized to an empty string "".

Figure 2.15.1: Declaring and assigning a string.

```
A boy ate an apple.
```

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

int main() {
   string sentenceSubject;
   string sentenceVerb;
   string sentenceObject = "an apple";

   sentenceSubject = "boy";
   sentenceVerb = "ate";

   cout << "A ";
   cout << sentenceSubject << " ";
   cout << sentenceVerb << " ";
   cout << sentenceObject << "." << endl;

   return 0;
}
```

Feedback?

PARTICIPATION ACTIVITY 2.15.3: Declaring and assigning a string variable.

1) Declare a string variable userName.

Check Show answer

2) Write a statement that assigns userName with "Sarah".

Check Show answer

3) Suppose string str1 is initially "Hello" and str2 is "Hi".
After `str1 = str2;`, what is str1?
Omit the quotes.

Check Show answer

4) Suppose str1 is initially "Hello" and

str2 is "Hi".
After `str1 = str2;` and then `str2 = "Bye";`, what is str1?
Omit the quotes.

Check Show answer

5) Write one statement that declares a string named smallestPlanet initialized with "Mercury".

Check Show answer

Feedback?

Getting a string without whitespaces from input

A ***whitespace character*** is a character used to represent horizontal and vertical spaces in text, and includes spaces, tabs, and newline characters. Ex: "Oh my goodness!" has two whitespace characters, one between h and m, the other between y and g.

Below shows the basic approach to get a string from input into variable userString. The approach automatically skips initial whitespace, then gets characters until the next whitespace is seen (leaving that whitespace in the input).

```
cin >> userString;
```

PARTICIPATION ACTIVITY 2.15.4: Getting a string without whitespace from input.

For the given input, indicate what string will be put into userString by:

```
cin >> userString;
```

1) abc

Check Show answer

2) Hi there.

Check Show answer

3) Hello! I'm tired.

Check Show answer

4) Very fun.

Check Show answer

Feedback?

PARTICIPATION ACTIVITY 2.15.5: Getting a string without whitespace from input (continued).

For the given input, indicate what string will be put into secondString by:

```
cin >> firstString;
cin >> secondString;
```

1) Oh my!

Check Show answer

2) Frank
Sinatra

Check Show answer

3) Oh...

...no!

Check Show answer

4) We all know

Check Show answer

Feedback?

Example: Word game

The following example illustrates getting strings from input and putting strings to output.

Figure 2.15.2: Strings example: Word game.

```
#include <iostream>
#include <string>     // Supports use of "string" data type
using namespace std;

/* A game inspired by "Mad Libs" where user enters nouns,
 * verbs, etc., and then a story using those words is output.
 */

int main() {
   string wordRelative;
   string wordFood;
   string wordAdjective;
   string wordTimePeriod;

   // Get user's words
   cout << "Type input without spaces." << endl;

   cout << "Enter a kind of relative: " << endl;
   cin  >> wordRelative;

   cout << "Enter a kind of food: " << endl;
   cin  >> wordFood;

   cout << "Enter an adjective: "   << endl;
   cin  >> wordAdjective;

   cout << "Enter a time period: "  << endl;
   cin  >> wordTimePeriod;

   // Tell the story
   cout << endl;
   cout << "My " << wordRelative << " says eating " << wordFood <<
endl;
   cout << "will make me more " << wordAdjective << "," << endl;
   cout << "so now I eat it every " << wordTimePeriod << "." <<
endl;

   return 0;
}
```

```
Type input without spaces.
Enter a kind of relative:
mother
Enter a kind of food:
apples
Enter an adjective:
loud
Enter a time period:
week

My mother says eating
apples
will make me more loud,
so now I eat it every
week.
```

Feedback?

Getting a string with whitespace from input

Sometimes a programmer wishes to get whitespace characters into a string, such as getting a user's input of the name "Franklin D. Roosevelt" into a string variable presidentName.

For such cases, the language supports getting an entire line into a string. The function ***getline***(cin, stringVar) gets all remaining text on the current input line, up to the next newline character (which is removed from input but not put in stringVar).

PARTICIPATION ACTIVITY 2.15.6: Getting a string with whitespace from input.

What does the following statement get into the indicated variable, for the given input?

```
getline(cin, firstString);
getline(cin, secondString);
```

1) Hello there!
Welcome.

firstString gets _____ .

- Hello
- Hello there!
- Hello there!
Welcome.

Correct

The first statement gets the entire first line. The string does not contain the newline, which instead is just disposed.

2) I
don't
know.

firstString gets _____ .

- I
- I don't
- I
don't

Correct

The first line consists of just one letter, so firstString will just have that one letter.

3) Hey buddy.
What's up?

secondString gets _____ .

- Hey buddy.

Correct

The first statement gets the first line, "Hey buddy." The second statement gets "What's up?" into secondString.

- (selected) What's up?
- (empty)

4)

abc
def

secondString gets _____ . (Note that the first line above is blank).

- (selected) abc
- def
- (blank)

Correct

The first statement gets the first line, which is blank, yielding an empty string. Thus the second statement gets "abc" into secondString. This behavior is quite different than the approach for getting a string without whitespace, which ignores leading whitespace and thus would have ignored the initial blank line.

5) Walk away

firstString gets _____ . (Note the leading spaces before Walk).

- Walk away
- (selected) Walk away

Correct

The statement gets the entire line, including leading spaces. The statements do not ignore leading spaces.

Feedback?

Example: Getting multi-word names

Figure 2.15.3: Reading an input string containing spaces using getline.

```
Enter first name:
Betty Sue
Enter last name:
McKay

Welcome Betty Sue McKay!
May I call you Betty Sue?
```

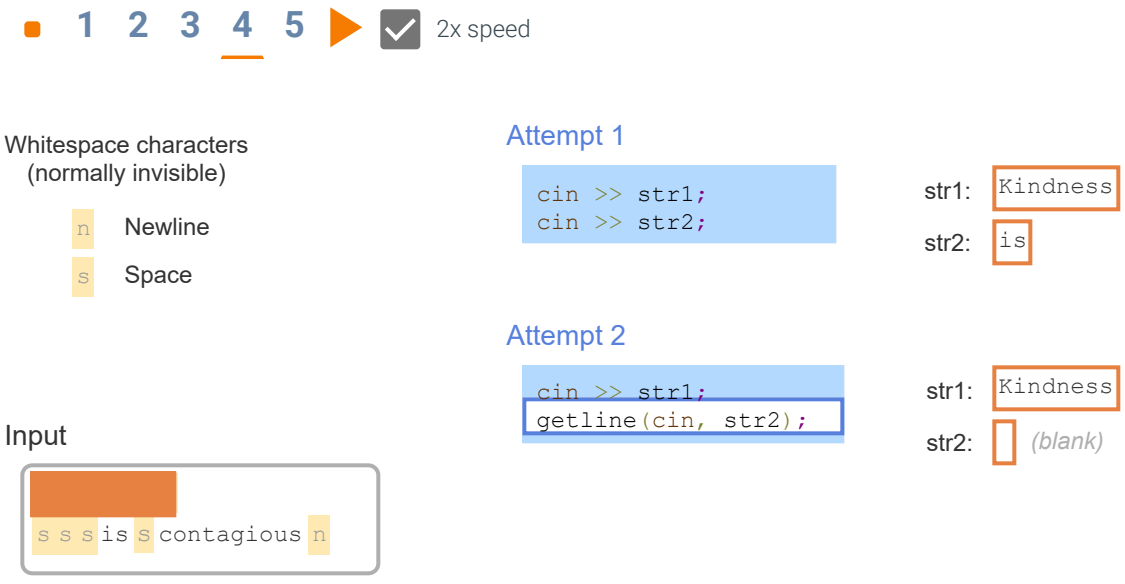
```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string firstName;
   string lastName;

   cout << "Enter first name:" << endl;
   getline(cin, firstName); // Gets entire line up to ENTER

   cout << "Enter last name:" << endl;
   getline(cin, lastName); // Gets entire line up to ENTER

   cout << endl;
   cout << "Welcome " << firstName << " " << lastName << "!" <<
endl;
   cout << "May I call you " << firstName << "?" << endl;

   return 0;
}
```

Feedback?

Mixing cin and getline

Mixing `cin >>` and `getline()` can be tricky, because `cin >>` leaves the newline in the input, while `getline()` does not skip leading whitespace.

PARTICIPATION ACTIVITY 2.15.7: Combining cin and getline() can be tricky.



Combining cin >> with getline() is a little tricky. The cin >> str1 leaves the newline in the input. Then, getline(cin, str2) gets the rest of the line, which is blank.

Feedback?

PARTICIPATION ACTIVITY 2.15.8: Getting strings without and with whitespace.

Given the following input:

```
Every one
is great.
That's right.
```

1) What does the following get into str2?

```
cin >> str1;
cin >> str2;
```

- Every
- one
- Every one

Correct

The first statement gets "Every" stopping at the space after y. The second statement skips that leading space, gets "one", and stops at the newline.

2) What does the following get into str2?

```
cin >> str1;
getline(cin, str2);
```

- Every one
- one
- one (has a leading space)

Correct

The first statement left a space in the input. getline() starts with what's left in the input, so starts with that space, continuing to the newline.

3) What does the following get into str2?

```
cin >> str0;
cin >> str1;
getline(cin, str2);
```

- one
- (blank)
- is great.

4) What does the following get into str2?

Correct

```
cin >> str0;
cin >> str1;
getline(cin, tmpStr);
getline(cin, str2);
```

The first getline() gets past the first newline. So the second getline() starts from the beginning of the second line, thus getting "is great.".

- (blank)
- is great.
- That's right.

5) What does the following get into str3?

```
cin >> str0;
cin >> str1;
getline(cin, tmpStr);
getline(cin, str2);
getline(cin, str3);
```

- That's right.
- (blank)

Correct

The second getline() gets the second line, "is great." into str2, and consumes that line's newline (it does NOT leave the newline in the input). Thus, the third getline() gets the third line, which is "That's right."

6) What does the following put into str2?

```
cin >> str0;
cin >> str1;
cin >> tmpStr;
getline(cin, str2);
```

- is
- is great.
- great. (has a leading space)

Correct

The second cin puts "one" into str1, and left the newline in the input. The third cin skips that leading whitespace, and puts "is" into tmpStr, leaving the trailing space in the input. Thus, the getline() will get " great". The third cin is not a means of skipping a newline; another getline() is needed.

Feedback?

CHALLENGE ACTIVITY 2.15.1: Reading and outputting strings.

Write a program that reads a person's first and last names, separated by a space. Then the program outputs last name, comma, first name. End with newline.

Example output if the input is: Maya Jones

```
Jones, Maya
```

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

int main() {
   string firstName;
   string lastName;

   /* Your solution goes here  */
   cin >> firstName;
   cin >>  lastName;
   cout << lastName << ", " << firstName<< endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with: Maya Jones

Your output `Jones, Maya`

✓ Testing with: Stan Li

Your output `Li, Stan`

Feedback?