Mr. Giansante

C++ Programming

Strings and Chars

August 2018

getline() Function

Before we begin exploring string and char functions, we will examine a method for inputting strings from the user.

```
string userinput;
cin << userinput;
```

With the code above, if the user types:

```
My name is Phil
```

The variable `userinput` will be set to "`My`" (not "`My name is Phil`" as you might expect.

That is because when you use cin for user input, it stops at a space or newline (`\n` or `\f` or `\n\f`).

When we require the user to input something that might include spaces, we need to use `getline()`.

The `getline()` function is defined in the `string` header.

```
include <string>
string userinput;
getline(cin, userinput);
```

Clearing the Input Stream

The `getline()` function will read all the characters up to the newline character `\n` and store them in the specificed string variable.

Unfortunately, the newline character `\n` is left in the Input Stream and this might cause subsequent `cin()` or `getline()` functions to act unpredictably.

You need to flush the newline character out of the buffer. You can do it by using `cin.ignore()`

cin.ignore()

```
include <string>
string userinput;
cout << "Enter your full name: ";
cin.ignore();
getline(cin, userinput);
```

cin.ignore() can be called three different ways:

No arguments

A single character is taken from the input buffer and discarded.

```
cin.ignore(); // discard 1 character
```

One argument

The number of characters specified are taken from the input buffer and discarded.

```
cin.ignore(33); // discard 33 characters
```

Two arguments

Discard the number of characters specified, or discard characters up to and including the specified delimiter (whichever comes first).

```
cin.ignore(26, '\n');
// discard 26 characters or to a newline,
// whichever comes first
```

String Functions

String Library

The `<string>` library offers C++ many functions that can operate on string variables.

The examples on this page assume you have made the following declarations.

```
#include <string>

string myString;
myString = "Programming is fun";
```

Length of a String

The Length property returns the number of characters in the string.

```
n = myString.Length();
```

In this case, the variable n would be set to 18 since there are 18 characters in "Programming is fun".
Note: spaces are counted as characters.

At() Function

Returns a reference to the character at a position in the string.

```
// Returns the first character in string
myString.at(0);

// Returns the second character in string
myString.at(1);

// Returns the third character in string
myString.at(2);
```

To return the last character in a string, we can use the length function. If the lenght of the string is 10, then the last character is in position 9 (ie. length - 1)

```
// Returns the last character in string
myString.at(myString.length() - 1);
```

Substrings

`Substr()` returns a new string that is a substring of the original string.

The substring begins at the specified index and extends up to the specified length.

```
myString.substr(___, ___);
```

Start at index

How many characters to extract.

Note: Recall that the first character in the string is in position 0, not 1.

If you want to take the first 5 characters of a string ...

```
myString.substr(0, 5);
```

If you want to take the third to seventh characters of a string, use ...

```
myString.substr(2, 5);
```

If you want to take the last character of a string ...

```
myString.substr(myString.Length() - 1, 1)
```

If you want to take the last 5 characters of a string ...

```
myString.substr(myString.Length() - 5, 5)
```

Find inside a String

The `find()` function returns the starting postion of a substring inside a string. If the substring is not found, it returns a number that is NOT between `0` and `myString.lenght - 1`

```
i = myString.find("fun");
```

This will set the variable i to 15.

You can also specify the position where find starts looking ...

```
i = myString.find("fun", 5);
```

Char Functions

cctype Library

The <cctype> library offers C++ many functions that can operate on string variables.

The examples on this page assume you have made the following declarations.

In addition to strings, you can have chars. Chars can only contain 1 character.

Tip: When using chars, use single quotes rather than double quotesl

The examples on this page assume you have made the following declaration.

```
#include <cctype>

char myChar;
myChar= 'A';
```

Char Functions

Check if character is lowercase:

```
if (islower(myChar)) ...
```

Check if character is uppercase:

```
if (isupper(myChar)) ...
```

Check if character is alphabetic:

```
if (isalpha(myChar)) ...
```

Check if character is a digit:

```
if (isdigit(myChar)) ...
```

Check is character is alphanumber (ie. letter or digit):

```
if (isalnum(myChar)) ...
```

Check is character is a space:

```
if (isspace(myChar)) ...
```

Check is character is punctuation

```
if (ispunct(myChar)) ...
```

Changing Case

To change a character to uppercase ...

```
cout << toupper(myChar);
```

To change a character to lowercase ...

```
cout << tolower(myChar);
```

Sample Program

The following program will examine a string "character by character" and display only the lowercase characters.

```
#include <iostream>
#include <cctype>
#include <string>

using namespace std;

int main()
{
    string myString = "Programming is FUN!";
    char myChar;

    for(int i = 0; i < myString.length(); i++)
    {
         myChar = myString.at(i);

         if(islower(myChar))
         {
              cout << myChar;
         }
     }

       return 0;
}
```

The resulting output would be ...

```
rogrammingis!
```

Note: All uppercase characters and spaces have been removed.

Examples

Searching for Characters

The following example uses a **For** loop to search a string for the letter "a".

```
string myString = "Programming is Fun";
bool found;

found = false;

for(int i = 0; i < myString.length(); i++)
{
	if(myString.at(i) == 'a') { found = true; }
}

cout << found;
```

We can modify the program to check for uppercase A's as well ...

```
if(myString.at(i) == 'a' || myString.at(i) == 'A') { found = true; }
```

Validating a Password

The following program asks the user to create a password that follows the following rules below and will inform the user if any of the rules have been broken.

- Must be at least 6 characters
- First character must be a digit
- Other characters must be only numbers or letters

```
#include <iostream>
#include <cctype>
#include <string>

using namespace std;

int main()
{
     string password;

     cout << "Please enter a password: ";
     cin >> password;

     if(password.length() < 6) { cout << "Password too short\n"; }

     if(not isalpha(password.at(0))) { cout << "Password must start with a letter\n"; }

     for(int i = 1; i < password.length(); i++)
     {
          if(not isalnum(password.at(i))) { cout << "Password can only contain letters or numbers\n"; }
     }

     return 0;
}
```