```
Script started on Thu 23 Feb 2017 09:17:24 PM CST
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ pw
d
/home/students/b_pepa/CSC122/string_translation
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ca
t strin_\b\033[K\033[Kg_translation.info
Brandon Pepa
CSC122-001
String translation
Lab

(Level 2)

**(level 2)**

Description:
        The function of this program is to translate a string with a number into
the long equivelent. for example "123" becomes the number 123. the function needs
to ignore any leading spaces in the string, and it needs to ignore (and stop
translating) any non digits following the number.

**EXTRA CREDIT**
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ca
t strextra.\b\033[Kh
#ifndef STREXTRA_H_INC
#define STREXTRA_H_INC

long strToLong(const char str[], unsigned long & position);
long strToLong(const char str[]);

#endif
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ca
t strextra.cpp
#include <cctype>
#include <iostream>

long strToLong(const char str[], unsigned long & pos)
{
   pos = 0;
   //true will represent positive, false will represent negative
   bool sign = true;
   //Will represent the value of the string of digits
   long value = 0;

   //Removes all leading spaces
   while(isspace(str[pos]))
   {
     ++pos;
   }


   //Deals with the sign
   if(str[pos] == '-')
   {
      sign = false;
      ++pos;
   }
   else if(str[pos] == '+')
   {
      sign = true;
      ++pos;
   }
   else
   {
      sign = true;
   }



   //Main while loop for long interpretation
   while(isdigit(str[pos]))
   {
      //moves the value over by a 10's place (does nothing when value is 0)
      value *= 10;

      //typecasts the character to the ascii equivelent and then subtracts
      //48. '0' starts at 48 and is sequential so '1' is 49 and so on.
      //subtracting 48 will bring each digit to its proper long equivelent
      value += (static_cast<long>(str[pos])-48);
      ++pos;
   }


   //Makes the long negative if there was a negative sign in front of
   // the number
   if(!sign)
   {
      return -value;
   }
   return value;

}

long strToLong(const char str[])
{
   unsigned long pos = 0;
   //true will represent positive, false will represent negative
   bool sign = true;
   //Will represent the value of the string of digits
   long value = 0;

   //Removes all leading spaces
   while(isspace(str[pos]))
   {
     ++pos;
   }

   //Deals with the sign
   if(str[pos] == '-')
   {
      sign = false;
      ++pos;
   }
   else if(str[pos] == '+')
   {
      sign = true;
      ++pos;
   }
   else
   {
      sign = true;
   }


   //Main while loop for long interpretation
   while(isdigit(str[pos]))
   {
      //moves the value over by a 10's place (does nothing when value is 0)
      value *= 10;

      //typecasts the character to the ascii equivelent and then subtracts
      //48. '0' starts at 48 and is sequential so '1' is 49 and so on.
      //subtracting 48 will bring each digit to its proper long equivelent
      value += (static_cast<long>(str[pos])-48);
   }

   //Makes the long negative if there was a negative sign in front of
   // the number
   if(!sign)
   {
      return -value;
   }
   return value;

}
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ca
t strDriver.cpp
#include <iostream> //for cin and cout
#include "strextra.h"
using namespace std;

int main()
{
   unsigned long position;
   const short MAX_STR = 128;
   char str[MAX_STR];
   char cont;
   long convLong;

   do
   {
      cout << "Enter the string you wish to be converted" << endl;
      cin.getline(str,INT_MAX);
      convLong = strToLong(str, position);
      cout << "The Long is: " << convLong
           << "\nThe position ended is: " << position << endl;

      cout << "\nWould you like to find another number? (Y/N) " << endl;
      cin >> cont;
      cin.ignore(INT_MAX,'\n');

   }while(cont == 'y' || cont == 'Y');

return 0;
}
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ CP
P driver.cpp\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[KstrDriver strextra
strDriver.cpp***
strextra.cpp...


\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ./
strDriver.out
Enter the string you wish to be converted
12
The Long is: 12
The position ended is: 2

Would you like to find another number? (Y/N)
y
Enter the string you wish to be converted
         15.2
The Long is: 15
The position ended is: 11

Would you like to find another number? (Y/N)
y
Enter the string you wish to be converted
-1234
The Long is: -1234
The position ended is: 5

Would you like to find another number? (Y/N)
y
Enter the string you wish to be converted
+1.4
The Long is: 1
The position ended is: 2

Would you like to find another number? (Y/N)
y
Enter the string you wish to be converted
apes
The Long is: 0
The position ended is: 0

Would you like to find another number? (Y/N)
n
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ca
t string_translation.tpq
Thought Provoking Questions

1. What argument(s) does your function need? Will you change the argument(s)?
   What type of value(s) (if any) does it return?

A: The function needs to be passed a string. The argument doesn't need to be
   changed so it can be passed as a constant call by refernce. The function
   should return a long. A long position will need to be passed as a call by
   refernce in order to return a position to the function of where it stopped
   translating.

2. How can you easily skip past the beginning whitespace? Is there a library
   function that might help detect the characters within the string are
   whitespace characters?

A: The character manipulation library (cctype) has the function "isspace"
   you pass a character to the function and it will return a bool. I can skip
   all the characters in the beginning that return true to this function.

3. Is the translation of "-123" actually significantly different than that
   for "+123" or "123"? What one little thing must change?

A: The translation shouldn't change except for at the end where you subtract
   the quantity from zero

4. When translating "1" how can you access just that character '1'? What
   other character is in the string?

A: At the end of all c-strings is the null character '/0' that sybolizes the
   end of the string. you need to make sure to only look at characters
   instead of the whole string at a time.

5. When translating "1" how can you convert the character representation of
   the digit 1 to a numeric version?

A: the character '1' needs to be typecasted to a long. this will give the
```

```
   ascii character value of ’1’ which would be 49. all the digits are in
   order in ascii values starting at ’0’ = 48 to ’9’ = 57 so 48 would just
   need to be subtracted to get the correct value

6. When translating "12", how can you ensure that you end up with 12 instead
   of 3? (i.e. you don’t just add the digits together, right? Maybe you
   should look back at the notes on how cin’s >> operator does it...)

A: cin’s >> operator works for numbers by looking at the first digit, assign
   it to the variable, multiply the number by 10, then add the next digit.
   I can do it the same way in my program as well.

7. How can you stop and ignmore the extra information in "+1.4"? (i.e the
   ".4" is irrelevant to the translation of the integer value "+1"-->1.) what
   tells you you are out of useful information? is this different from the
   string "1 box"? how about "12false"? Is there a library function that
   might help detect what kind of character you are [about to be] dealing
   with? (again, not a string function.)

A: the while loop looking for digits should look use the function "isdigit"
   from the character manipulation library to detect if it’s a digit. duh

8. How can you tell the caller the position at which you stopped translating?
   (Hint: Is there a reason that return wasn’t highlighted as a keyword above
   but rather put in quotes like that?)

A: An increment variable will need to be used and this can be the same as the
   call by refernce position varible in order to return to the caller the
   position where the function stopped translating.

9. When the string contains no integer, how can you have the user’s value be/
   remain undefined? (That is, in the same state that it entered your
   function..?)

A: The function can return 0 when it is undefined and didn’t return a value

10.How can the user detect that their value hasn’t been modified? (i.e. when
   their string didn’t contain a valid integer, how can they tell?)(Hint: if
   if you couldn’t translate the anything, what position did you tell them
   you stopped?)

A: the position it stopped should be 0 if it didn’t translate anything... but
   this kind of makes it a little complicated if there were leading spaces
   before a non-integer value... should I account for this?

11.When translating a string to an integer, we can start from either the
   front or the end of the string. Discuss(briefly) the merits of each and
   conclude which is best.

A: translating the string from the beginning would be ideal due to the
   already used formula cin uses to translate along with the fact that is the
   order the user is already inputing data. translating from the back can
   also run into a weird problem if you have characters seperating numbers
   such as "12 apples 5" where from the back it would translate 5 and the
   front would translate 12.

12.How do you protect your library from being circularly included?

A: #ifndef and #define will be used in the header file followed by the
   functions then the #endif.

13.What changes are needed in your main application (the test application
   here) to get it to work with the library? What about the compiling process   ?

A: the main application needs to include the header file, while when
   compiling, you need to compile the library implementation file with the
   main application

14.How many filess does your library consist of? What are they? Which ones
   do you #include?

A: the header file is the one that is #include while the implementation file
   (.cpp) will need to be compiled with the main function.
\033]0;b_pepa@mars:~/CSC122/string_translation\007[b_pepa@mars string_translation]$ ex
it
exit

Script done on Thu 23 Feb 2017 09:19:27 PM CST
```