Week 2 Lecture 2: Problem Solving

Review and Preview

- Reviewed if
- Looping
- wcount.c program

- Finish up wcount.c
- Problem Solving
- Assignment 1

K&R: Chapter 1
Sections 1.1 to 1.6
Sections 2.1 to 2.6

Inside the Loop

Remember that the variable **`state`** *starts as* **`OUT`**

```
numChars++;

if ( c == '\n' ) {
    numLines++;
}

if ( c == ' ' || c == '\n' || c == '\t' ) {
    state = OUT;
} else if (state == OUT) {
    state = IN;
    numWords++;
}
```

`if` the character is a *Newline* character

`if` the character is a Space, Newline, or Tab

`else if` the character is **NOT** a Space, Newline, **or** Tab **and** you were outside of a word

You were outside of a word but now you have found a letter, i.e. another word, so increment the word count and set **`state`** to be inside of a word.

```
int main ( int argc, char *argv[] ) {
   char c;
   int numLines = 0;  /* number of lines */
   int numWords = 0;  /* number of words */
   int numChars = 0;  /* number of characters */

   int state = OUT;

   while ( (c = getchar()) != EOF ) {
      numChars++;

      if ( c == '\n' ) {
         numLines++;
      }

      if ( c == ' ' || c == '\n' || c == '\t' ) {
         state = OUT;
      } else if (state == OUT) {
         state = IN;
         numWords++;
      }
   }
   printf ( "%d %d %d\n", numLines, numWords, numChars );

   return ( 0 );
}
```

Newline: \n

Tab: \t

Update char count

Update line count

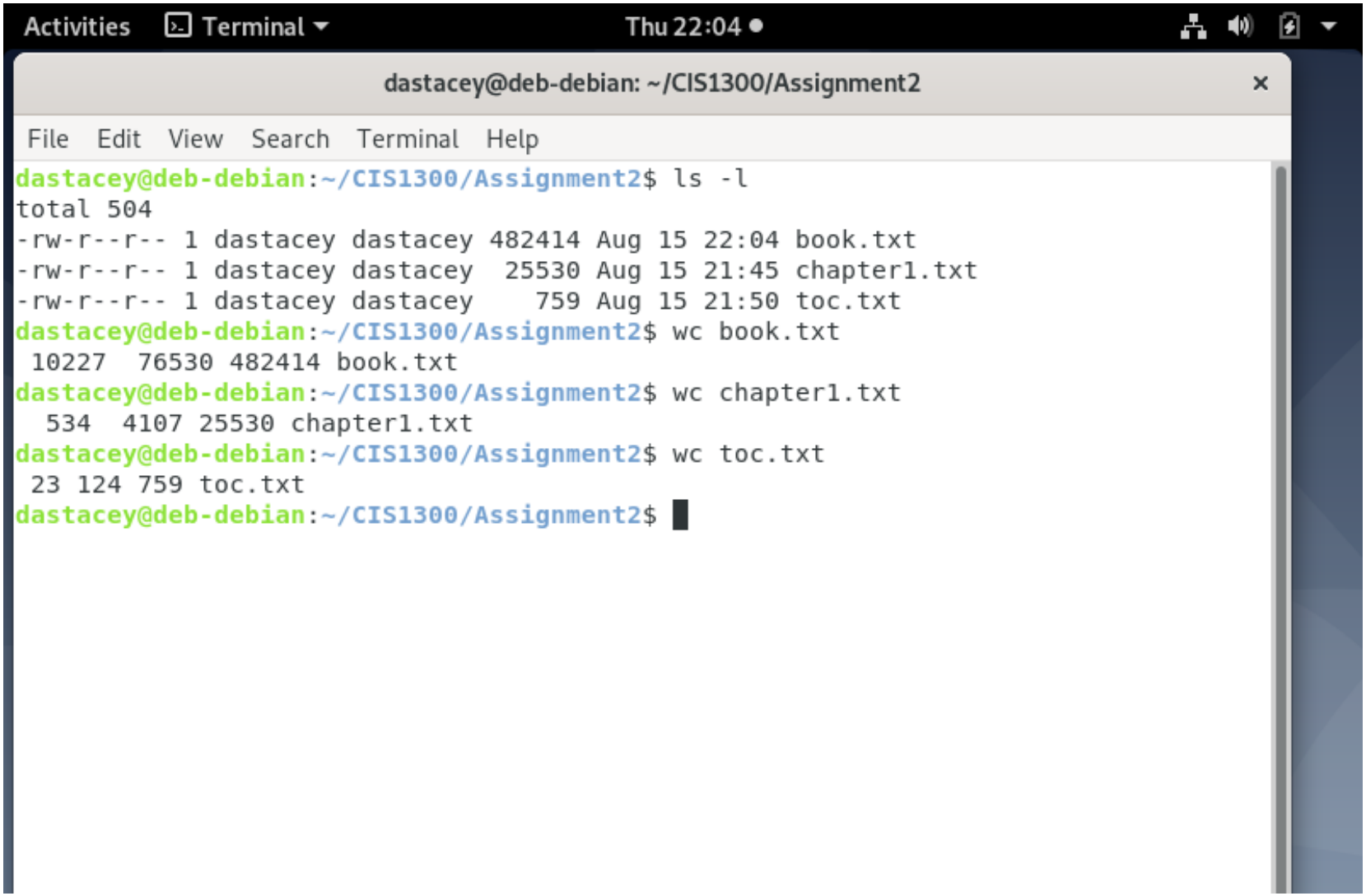
Update word count

	T	**h**	**i**	**s**		**i**	**s**		**o**	**n**	**e**	**.**	**\n**
OUT	IN	IN	IN	IN	OUT	IN	IN	OUT	IN	IN	IN	IN	OUT
Char	1	2	3	4	5	6	7	8	9	10	11	12	13
Word	1	1	1	1	1	2	2	2	3	3	3	3	3
Line	0	0	0	0	0	0	0	0	0	0	0	0	1

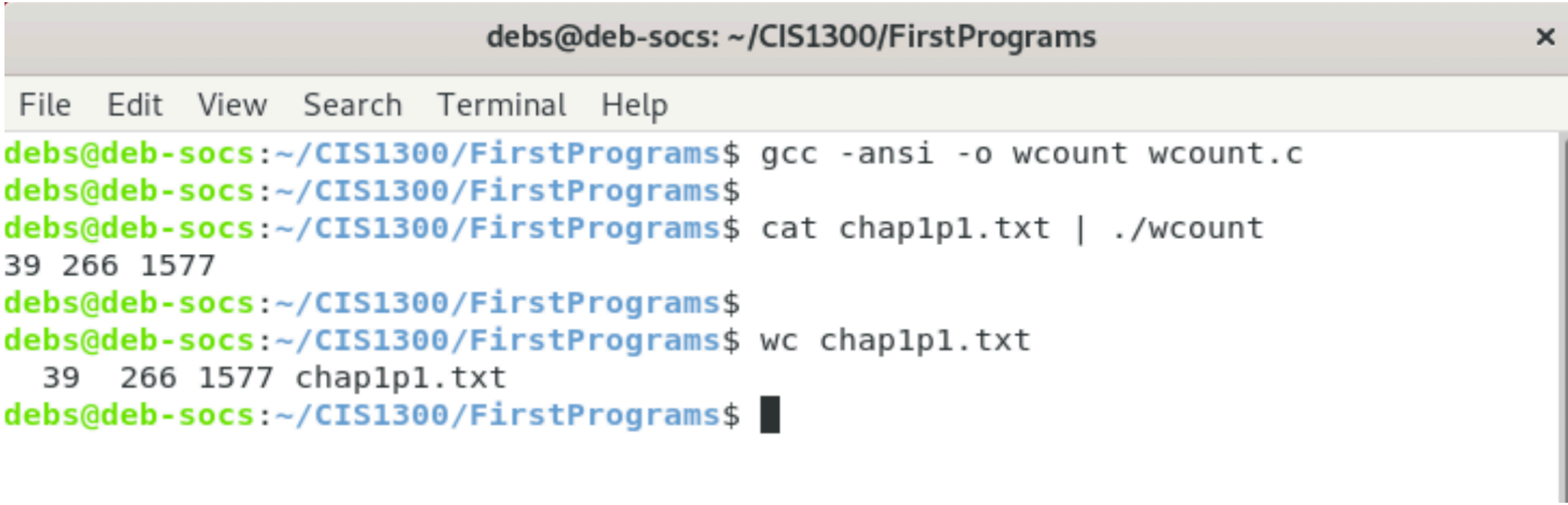
	L	**a**	**s**	**t**		**l**	**i**	**n**	**e**	**.**	**\n**
	IN	IN	IN	IN	OUT	IN	IN	IN	IN	IN	OUT
Char	14	15	16	17	18	19	20	21	22	23	24
Word	4	4	4	4	4	5	5	5	5	5	5
Line	1	1	1	1	1	1	1	1	1	1	2

`wc` - character, word and line counts

- print newline, word, and byte counts for each file



Testing against the shell command wc



Problem Solving

What is it?
How do we Use it in Coding?

What is Problem Solving?

- Problem solving is a **4 step process**:
  - **Understand** the problem/current situation.
  - **Identify** the cause(s) of the problem.
  - Develop an effective and efficient action **plan**.
  - **Execute** the plan, **analyze** its effectiveness and **modify** it until the problem is solved.

Watanabe, Ken, Problem Solving 101, Penguin Books, 2009

Understand the problem/current situation

Communication

- Has anyone else run into this problem before?
- Ask for advice when applicable but do not expect others do your work for you!

Reflection

- Have you ever run into this type of problem before?
- Did you write down what the problem was and how you solved it?

Understand the problem/current situation

Research

- Google, library, reference books, books on topic - look up information before formulating a plan.
- Go to forums, post questions, email experts if appropriate - the answer may be out there!

Testing and Experimentation

- Do you know enough about the problem?
- More testing never hurts - refer to appropriate documents like specifications, design docs, etc.

Execute the plan, analyze its effectiveness and modify it until the problem is solved

---

Do one thing at a time!

- Solve problems in order - root cause first and then go down your **list** of priorities.
- Before moving on to the next step in your plan, test to make sure that the issue is resolved.
- **Keep copies!** Use revision control or some other system so that you can rollback any changes or actions that did not have the desired effect.

Assignment #1

How to Approach the Assignment
An Example of Problem Solving

Stage A

- ./daysCalculatorA dd1 mm1 yyyy1 dd2 mm2 yyyy2
  where dd1 mm1 yyyy1 represents the start date and dd2 mm2 yyyy2 represents the end date. The program returns the number of days for
  - From and including the Start Date
  - To, but not including, the End Date

Very Important!!!

- The output format is **just** the number. For example,

  $ ./daysCalculatorA 14 9 2019 9 12 2019

  86

What can we do first? What do we already know?

- Command line arguments: `argc`, `argv`
  - Check if there are enough parameters
  - Assign the values (with the correct type) to the appropriate variables
    - e.g. `dd = atoi ( argv[1] );`

What information do you need?

- The number of days in each month.
- Is the year a leap year? If it is then it changes the number of days in February
- But do we really need the number of days per month?
- We could use **Day-of-Year**
    - Need to understand what Day-of-Year is and how it could be used for calculations

Calculations - First Simplify the Problem

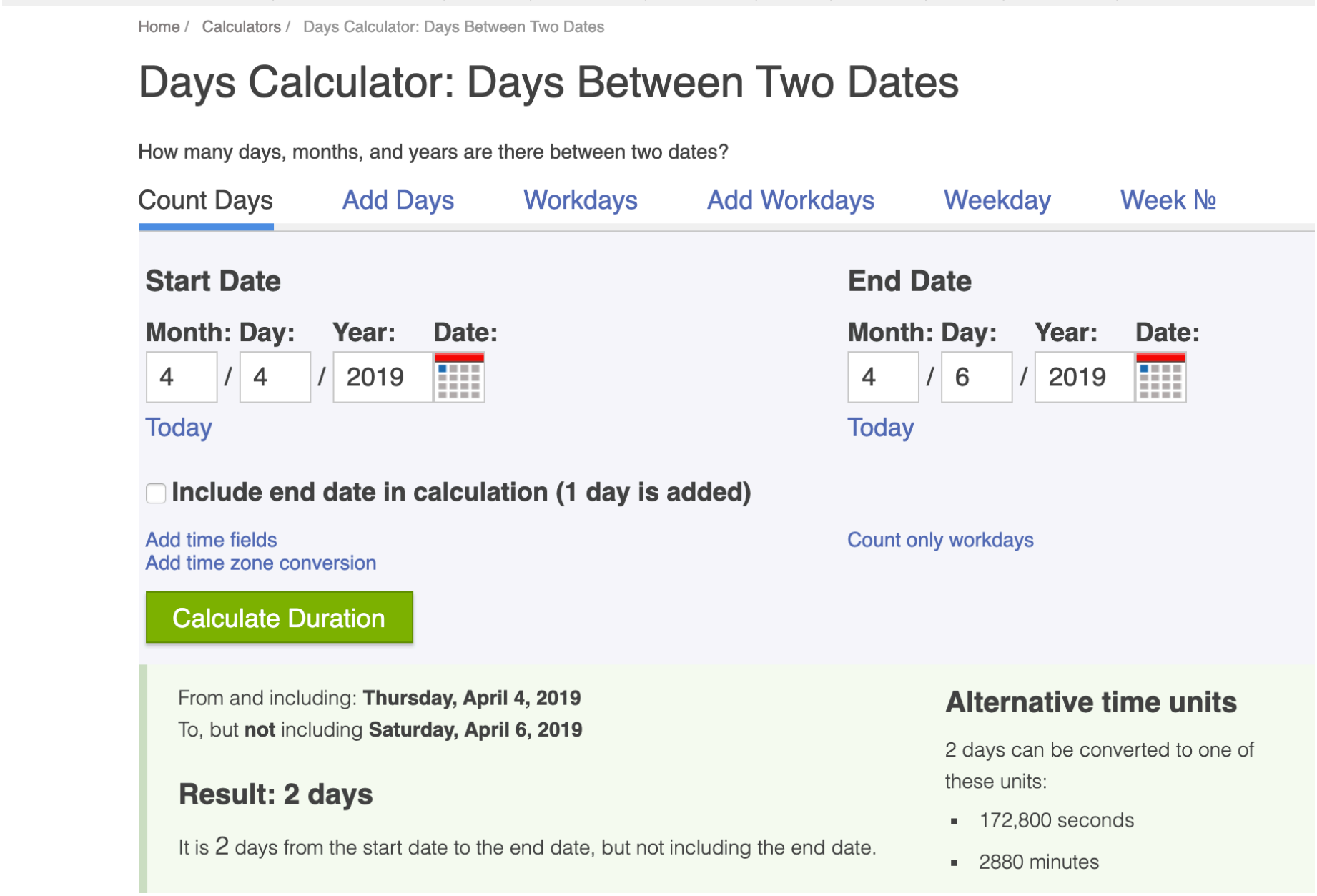
- How to start - what about if the start and end dates are in the same month of the same year?
  - Pick two dates: April 4 - April 6
- How can we use the Day-of-Year information?
- Convert the given table to an array

Jan: 1	May: 121	Sep: 244
Feb: 32	Jun: 152	Oct: 274
Mar: 60	Jul: 182	Nov: 305
Apr: 91	Aug: 213	Dec: 335

Calculations - First Simplify the Problem

- April 1 = 91 4 - 1 = 3 6 - 1 = 5
- April 4 = 94 91 + 3 (Day-of-Year for April 4)
- April 6 = 96 91 + 5 (Day-of-Year for April 6)
- From April 4 (94) to but not including April 6 (96)
- 96 - 94 = 2

Checking your answer



Home / Calculators / Days Calculator: Days Between Two Dates

Days Calculator: Days Between Two Dates

How many days, months, and years are there between two dates?

Count Days | Add Days | Workdays | Add Workdays | Weekday | Week №

Start Date

Month: 4 / **Day:** 4 / **Year:** 2019 **Date:**

Today

End Date

Month: 4 / **Day:** 6 / **Year:** 2019 **Date:**

Today

☐ **Include end date in calculation (1 day is added)**

Add time fields
Add time zone conversion

Count only workdays

Calculate Duration

From and including: **Thursday, April 4, 2019**
To, but **not** including **Saturday, April 6, 2019**

Result: 2 days

It is 2 days from the start date to the end date, but not including the end date.

Alternative time units

2 days can be converted to one of these units:

- 172,800 seconds
- 2880 minutes

Calculations - Expanding our Scope

- Does it matter if the month is different if the year is the same?
  - If the months are in the same year, just calculate their Day-of-Year and use this for calculations.
- What if the year is a leap year?

Jan: 1	May: 122	Sep: 245
Feb: 32	Jun: 153	Oct: 275
Mar: 61	Jul: 183	Nov: 306
Apr: 92	Aug: 214	Dec: 336

Another Benefit of Day-of-Year

- Once you have converted the start and end dates to their Day-of-Year than you can tell if the end date occurs after the start date

```
startDay-of-Year < endDay-of-Year
```

- What if `startDay-of-Year == endDay-of-Year` ?

So What Have We Accomplished?

- Reading in the start and end dates and assigning them to their proper variables.
- Assuming that both dates are in the same year:
  - Determine if the year is a leap-year
  - Using the appropriate Day-of-Year table, convert both start and end dates to Day-of-Year
  - Check to see if the start date occurs before the end date
  - Calculate the days in between start and end dates.

CalculatorA is done.

What else do we need?

- CalculatorB
  - Check to see if there is one more command line argument and if there is then read it in (string variable).
  - Is it “include”?
  - How do you check if one character string is the same as another character string?
  - Let’s ask Google!

Google

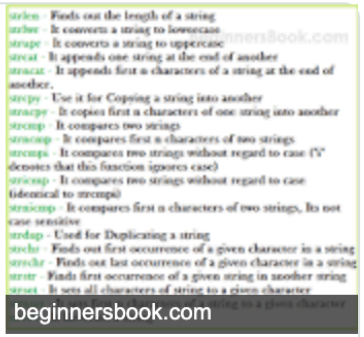
c string functions

All Images Shopping Videos News More Settings Tools

About 935,000,000 results (0.53 seconds)

Search string

The nine most commonly used functions in the string library are:



- strcat - concatenate two **strings**.
- strchr - **string** scanning operation.
- strcmp - compare two **strings**.
- strcpy - copy a **string**.
- strlen - get **string** length.
- strncat - concatenate one **string** with part of another.
- strncmp - compare parts of two **strings**.

More items...

C Programming/String manipulation - Wikibooks, open books ...
https://en.wikibooks.org › wiki › C_Programming › String_manipulation

About Featured Snippets Feedback

Pick →

String Manipulations In C Programming Using Library Functions
https://www.programiz.com › c-programming › string-handling-functions ▾
In this article, you'll learn to manipulate **strings** in **C** using library **functions** such as gets(), puts, strlen() and more. You'll learn to get **string** from the user and ...

C Programming/String manipulation - Wikibooks, open books ...
https://en.wikibooks.org › wiki › C_Programming › String_manipulation ▾

String Manipulations In C Programming Using Library Functions

In this article, you'll learn to manipulate strings in C using library functions such as gets(), puts, strlen() and more. You'll learn to get string from the user and perform operations on the string.

To solve this, C supports a large number of string handling functions in the standard library `"string.h"`.

Few commonly used string handling functions are discussed below:

Function	Work of Function
strlen()	Calculates the length of string
strcpy()	Copies a string to another string
strcat()	Concatenates(joins) two strings
strcmp()	Compares two string
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase

This one looks promising!

Strings handling functions are defined under `"string.h"` header file.

```
#include <string.h>
```

Needed header file

C strcmp()

The strcmp() function compares two strings and returns 0 if both strings are identical.

C strcmp() Prototype

```
int strcmp (const char* str1, const char* str2);
```

The strcmp() function takes two strings and return an integer.

The strcmp() compares two strings character by character. If the first character of two strings are equal, next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.

It is defined in string.h header file.

Return Value from strcmp()

Return Value	Remarks
0	if both strings are identical (equal)
negative	if the ASCII value of first unmatched character is less than second.
positive integer	if the ASCII value of first unmatched character is greater than second.

Just what we need to know

Example: C strcmp() function

```c
#include <stdio.h>
#include <string.h>

int main()
{
     char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
     int result;

     // comparing strings str1 and str2
     result = strcmp(str1, str2);
     printf("strcmp(str1, str2) = %d\n", result);

     // comparing strings str1 and str3
     result = strcmp(str1, str3);
     printf("strcmp(str1, str3) = %d\n", result);

     return 0;
}
```

Output

```
strcmp(str1, str2) = 32
strcmp(str1, str3) = 0
```

The first unmatched character between string `str1` and `str2` is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67. Hence, when strings `str1` and `str2` are compared, the return value is 32.

When strings `str1` and `str3` are compared, the result is 0 because both strings are identical.

Is the command line argument “include” present?

```
if ( strcmp(“include”, argv[7]) == 0 ) {

    /* we have detected the word include */

}
```

- Now you need to calculate the length of time when you include the end date.

CalculatorB is done.
You have 75%!

Solving Stage C

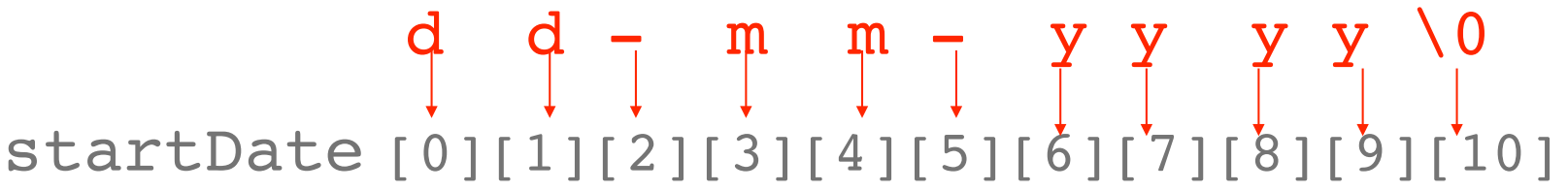
`./daysCalculatorC dd1-mm1-yyyy1 dd2-mm2-yyyy2`

- Notice that there are 2 arguments after the program name on the command line and not 6.
- Each date is now a string of the form

  `dd-mm-yyyy`

- If you put the start date into a char array such as

```
char startDate[11];
```



Strings always end with the character '\0'

Solving Stage C

- There are many ways to solve this:
  - Take the **dd** characters and put them in their own character array and then convert to a string.
  - Remember a string always ends in '\0';

```
char dayStr[3];
dayStr[0] = argv[1][0];
dayStr[1] = argv[1][1];
dayStr[2] = '\0';
```

CalculatorC is done.
You have 85%!

Solving Stage D

- Difficulty level has increased with this stage.
- Recognizing that there is no date string but there is a word ‘today’ is not too hard but…[Hint: same as ‘include’]
- How do you find out what the date is?
- There are two ways to do this.
- There will be a hint on the weekend.

Solving Stage E

- If you cannot solve Stage D, do not worry - move on to Stage E (without Stage D, finishing with Stage E is worth 90%).
- How do you calculate the number of days over multiple years?
  - How many “whole” years are there? They are either 365 or 366 days.
  - How many days in the partial year preceding the whole year(s) and how may days are in the partial year following the whole year(s)?