Week 7 – SOFT7019 lab session

This week we will utilise an online C IDE called online gdb, please access it at <https://www.onlinegdb.com/>

In the top right corner, you will have the option to select the programming language, please select C.



If you have a problem with this IDE, I would recommend installing CodeBlocks on your local machine. Last week we saw long lag times and errors ub the onlinegdb.com website during some of the lab sessions. I would recommend this guide to follow if installing CodeBlocks:

<https://www.youtube.com/watch?v=GWJqsmitR2I>

# Exercise 1

# **Break a sentence into words**

* write a function that takes a line of text and breaks it into words
* the function should take three parameters:
  + a character array that stores the line of text
  + an array of words to store the words
  + an integer pointer to store the number of words
* in the sentence words are separated by spaces
* as you go through the line of text,
  + if the character is not space then add it to the end of the current word
  + if it is a space then start a new word
  + if it is a newline, return the number of words
* end each word with a 0 character so they can be printed as strings.

Hints: What is an array of words?

* a word is a character array, which is equivalent to a pointer
* an array of words is an array of character arrays, or an array of pointers

There are several ways to approach this:

1. Declare a large matrix of characters with sufficient columns to accommodate long words, and sufficient rows to accommodate long sentences.

char words[50][20]; // can hold up to 50 20 letter words

void to\_words(char \*sentence, char words[][20], int \*num\_words){

// ...

while (input != '\n'){

if (input != ' ') {

words[crt\_word][word\_len++] = input;

}

else {

words[crt\_word][word\_len] = 0;

crt\_word ++;

word\_len = 0;

}

}

words[crt\_word][word\_len] = 0;

// why do we do this?

// ...

}

1. Declare an array of pointers to characters. Inside the function use malloc to allocate a fixed amount of space (e.g. 20 chars) for each word.

char \*words[50]; // can hold up to 50 words

void to\_words(char \*sentence, char \*words[], int \*num\_words){

// ...

while ...

words[crt\_word] = malloc(20); // allocate space for 20 chars

}

1. Declare an array of pointers to characters. Inside the function store the word into a temporary array. When the word is complete, allocate the exact space for the word and copy from the temporary array into the newly allocated word.

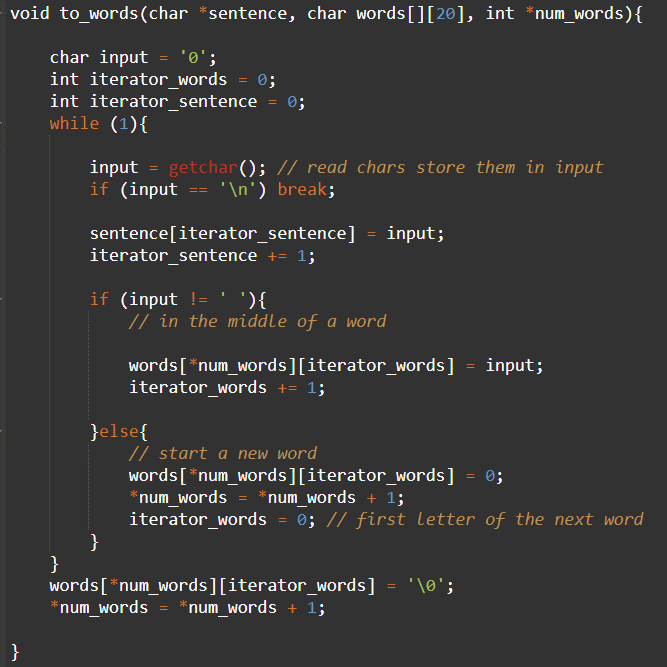


Figure Example code for exercise 1 version 1.

# Exercise 2

**Counting vowels in text**

* read a line of text
* write a function that counts the number of occurrences of each vowel
* the function should take as parameters

–the line of text; should be constant, to prevent modification within the function

–an array of 5 elements, representing the vowels; this should be used as the output of the function

* the function should perform error checking on the pointer parameters and return 1 in case of error, 0 otherwise
* at the end print the text read and for each vowel, the number of occurrences.

Example:

> gcc ex2.c -o ex2

> ./ex2

Input text:

this is a test for the example

You entered:

this is a test for the example

Vowel count:

a -> 2

e -> 3

i -> 2

o -> 1

u -> 0

Hints:

* use a switch statement to classify each character from the line read

switch (input) {

case'a': vowel\_count[0] ++; break;

case'e': vowel\_count[1] ++; break;

....

}

* when printing the vowel count, and going through the vowel\_count array, you need a reverse switch:

switch (i) {

case 0: printf("a: -> %d\n", vowel\_count[i]); break;

....

}