Week 6 – 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

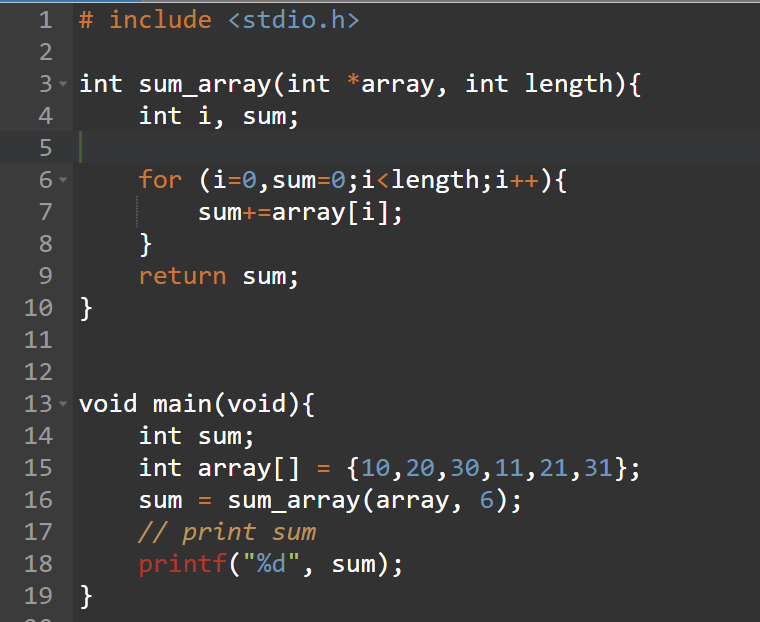
# **Summing up an array**

# write a function to sum up the elements of an array

# prototype: int sum\_array(int \*array, int length)

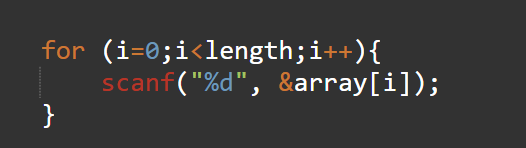
# Write three different versions, each version will define the array elements differently

Version 1: hardcode the array values in the main function



Version 2: Read array values with scanf() in a for loop

Don’t hardcode the array values, they should be user-defined.



Version 3: Declare the array dynamic, read numbers with scanf in a for loop

* ask the user to specify the number of integers in the list
* allocate exactly that with malloc: int \*array; array = (int\*)malloc(sizeof(int)\*num\_ints);
* read with scanf in a loop.

# Exercise 2

**Reading text**

* write a function to read in a line of text from the keyboard into a character array
* function should return the length of the text
* function prototype: int read\_text(char \*array); use getchar to read the text char by char, stopping at '\n'

In main

* declare the array (make it large enough)
* call the function to read the text
* print the array as a string.

Hint: after reading the array remember to put a ‘\0’ at array[length] so that it behaves as a string and can be printed with printf.

# Exercise 3

# **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.