//COS10007 – Developing Technical Software//  
// StudentID – 103234103 Son Nguyen//  
//Lab 9 – Week 9//  
  
1/ Looking into programs you developed in previous weeks (Week 1 to Week 8), which programs should be written in multiple modules? Name three and explain why they need to be rewritten into multiple source files.  
Week 3 Question 1 program – the program can be rewritten in multiple source files, it needs to be written into multiple source files to make the code look neater, the other files can contain function declarations to make the original c file a lot cleaner. It improves readability and less clustered.  
  
Week 5 Question 1 program – the program can be re-written in multiple source files – although not required to , if the program were to become more complex , it might be beneficial to split some of the code – in which case the struct into other files for the sake of organisation and modularity. One could contain the implementation of the linked list data structure and other can hold the function to manipulate and used the linked lis  
  
Week 7 Question 1 program – the program can become more complex if more array is introduced therefore if splitting into multiple source files make it a lot more organisable and increase efficiency. Since it’s a bubble sort algorithm, splitting them into multiple source files also make sure that we are able to check if the algorithm is correct or not , as well as other functionality.  
  
2/Try to rewrite your program in Week 4 Question 2 into multiple files, e.g. more than one .c file and .h header.  
  
stack.h   
#ifndef STACK\_H

#define STACK\_H

typedef struct stack {

char letter;

struct stack \*next;

} Stack;

typedef Stack \*stackPtr;

stackPtr push(stackPtr topPtr, char c);

stackPtr pop(stackPtr topPtr, char \*c);

#endif

stack.c   
  
#include "stack.h"

#include <stdlib.h>

#include <stdio.h>

stackPtr push(stackPtr topPtr, char c) {

stackPtr newPtr = malloc(sizeof(Stack));

if (newPtr == NULL) {

fprintf(stderr, "Memory allocation error\n");

exit(EXIT\_FAILURE);

}

newPtr->letter = c;

newPtr->next = topPtr;

return newPtr;

}

stackPtr pop(stackPtr topPtr, char \*c) {

if (topPtr == NULL) {

fprintf(stderr, "Stack underflow error\n");

exit(EXIT\_FAILURE);

}

\*c = topPtr->letter;

stackPtr tempPtr = topPtr;

topPtr = topPtr->next;

free(tempPtr);

return topPtr;

}

reverse\_string.c   
#include "stack.h"

#include <string.h>

void reverseString(char \*str) {

stackPtr topPtr = NULL;

int len = strlen(str);

for (int i = 0; i < len; i++) {

topPtr = push(topPtr, str[i]);

}

for (int i = 0; i < len; i++) {

topPtr = pop(topPtr, &str[i]);

}

}

main.c   
#include <stdio.h>

#include <string.h>

#include "stack.h"

void reverseString(char \*str);

int main() {

char str1[20];

printf("Enter your text: ");

fgets(str1, 20, stdin);

str1[strcspn(str1, "\n")] = '\0'; // remove trailing newline character

reverseString(str1);

printf("The string in reverse: %s\n", str1);

return 0;

}

3/ Write a complete program that prompt the user for the radius of a sphere and calculates and prints its volume. Use an inline function sphereVolume that returns the result of the volume  
  
#include <stdio.h>

// inline function to calculate the volume of a sphere

inline double sphereVolume(double radius) {

return (4.0/3.0) \* 3.14159 \* radius \* radius \* radius;

}

int main() {

double radius;

printf("Enter the radius of the sphere: ");

scanf("%lf", &radius);

double volume = sphereVolume(radius);

printf("The volume of the sphere is %.2lf cubic units.\n", volume);

return 0;

}

Output   
Text

Description automatically generated  
  
4/ Write a complete C++ program with the two alternate functions specified below. These functions will double the variable count defined in main.   
a. Function doubleByValue that passes a copy of count by value, double the copy and returns the new value.   
b. Function doubleByReference that passes count by reference via a reference parameter, double the original value of count through its alias, i.e. the reference parameter.  
  
#include <iostream>

using namespace std;

int doubleByValue(int);

void doubleByReference(int&);

int main() {

int count = 5; //hardcoded this for output

cout << "Original value of count: " << count << endl;

// double the value of count using doubleByValue (a)

int newCount = doubleByValue(count);

cout << "Value of count after doubleByValue: " << newCount << endl;

// double the value of count using doubleByReference (b)

doubleByReference(count);

cout << "Value of count after doubleByReference: " << count << endl;

return 0;

}

// function that doubles a value passed by value (a)

int doubleByValue(int count) {

return count \* 2;

}

// function that doubles a value passed by reference (b)

void doubleByReference(int& count) {

count \*= 2;

}

Output

Text

Description automatically generated