МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Сборка программ в Си

Студент гр. 1304 _____ Крупин Н. С.

Преподаватель _____ Чайка К. В.

Санкт-Петербург

2021

Цель работы.

Вынести подзадачи 1 лабораторной работы в отдельные файлы и написать make-файл для получившегося проекта.

Задание.

Вариант 6.

«В текущей директории создайте проект с make-файлом. Главная цель должна приводить к сборке проекта. Файл, который реализует главную функцию, должен называться menu.c; исполняемый файл – menu. Определение каждой функции должно быть расположено в отдельном файле, название файлов указано в скобках около описания каждой функции.

Реализуйте функцию-меню, на вход которой подается одно из значений 0, 1, 2, 3 и массив целых чисел размера не больше 100. Числа разделены пробелами. Строка заканчивается символом перевода строки.

В зависимости от значения, функция должна выводить следующее:

0 : индекс первого отрицательного элемента. (index_first_negative.c)

1 : индекс последнего отрицательного элемента. (index_last_negative.c)

2 : Найти сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). (sum_between_negative.c)

3 : Найти сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). (sum_before_and_after_negative.c)

ИНАЧЕ : необходимо вывести строку "Данные некорректны".

Ошибкой в данном задании считается дублирование кода!

Подсказка: функция нахождения модуля числа находится в заголовочном файле stdlib.h стандартной библиотеки языка Си.

При выводе результата, не забудьте символ переноса строки».

Выполнение работы.

Функции для выполнения подзадач размещены в отдельных файлах, также для каждой из них создан заголовочный файл с прототипом этой функции. С помощью директив препроцессора #ifndef-#defiine-#endif в созданные заголовочные файлы добавлена защита от повторного включения.

Создан make-файл для автоматизации сборки программы. В make-файле добавлена возможность сбрасывать результат работы утилиты – цель clean, удаляющая все объектные и исполняющий файлы в директории проекта.

Далее представлено описание работы функций, практически полностью совпадающее с представленным в отчёте к ЛР №1. Разработанный программный код и содержимое make-файла см. в приложении А.

Фукция main начинается с последовательного считывания входных данных: код задачи от 0 до 3 перехватывается переменной type; статически выделяется память на 100 целочисленных значений, указатель на её начало хранится в переменной arr; далее выделенная память заполняется числами из введённой строки, пока не встретится символ её окончания, для этого используется переменная-счётчик count для хранения текущего индекса элемента и накапливания их количества и символьная переменная gap, хранящая в себе символ, следующий за введённым числом, для возможности его проверки. После считывания реализуется вывод результата в зависимости от кода задачи, для каждого случая печатается результат работы одной из функций подзадач, также проводится примитивная обработка ошибок – в случае ввода несуществующего кода задачи или попытки вывода индекса не присутствующего в массиве отрицательного элемента выводится универсальное сообщение об ошибке "Данные некорректны" с переводом на новую строку; для исключения повторного вызова функций значения, требующие проверки, предварительно записываются в целочисленную переменную res_to_print.

Функция index_first_negative принимает в качестве аргументов указатель на начало массива arr и количество его элементов count. Использует переменную-счётчик i для хранения текущего индекса элемента и сохранения после выполнения цикла индекса первого отрицательного элемента, а если такового не окажется, принимает значение count. Функция возвращает индекс первого отрицательного элемента или значение count, если такого не существует.

Функция index_last_negative принимает в качестве аргументов указатель на начало массива arr и количество его элементов count. Использует переменную-счётчик i для хранения текущего индекса элемента и сохранения после выполнения цикла индекса последнего отрицательного элемента, а если такового не окажется, принимает значение -1. Функция возвращает индекс последнего отрицательного элемента или значение -1, если такого не существует.

Функция sum_between_negative принимает в качестве аргументов указатель на начало массива arr и количество его элементов count. Использует переменную-счётчик i для хранения текущего индекса элемента и целочисленную переменную sum для накопления суммы модулей элементов с индексами от index_first_negative (включительно) до index_last_negative (не включительно), данные значения индексов подсчитываются названными функциями. Функция возвращает сумму модулей элементов в указанном промежутке или значение 0, если промежуток пуст (при единственном отрицательном или их полном отсутствии).

Функция sum_before_and_after_negative принимает в качестве аргументов указатель на начало массива arr и количество его элементов count. Использует переменную-счётчик i для хранения текущего индекса элемента и целочисленную переменную sum для накопления суммы модулей элементов с индексами меньше index_first_negative (не включительно) или больше index_last_negative (включительно), данные значения индексов

подсчитываются названными функциями. Функция возвращает сумму модулей элементов в указанных промежутках, а в случае единственного отрицательного элемента или их полном отсутствии – сумму модулей всех элементов массива.

Тестирование.

Результаты тестирования, аналогичные полученным при тестировании 1 лабораторной работы, представлены в таблице 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 1 -2 3 -4 5 -6 7 8	1	Больше 2 отрицательных.
2.	1 1 -2 3 -4 5 -6 7 8	5	
3.	2 1 -2 3 -4 5 -6 7 8	14	
4.	3 1 -2 3 -4 5 -6 7 8	22	
5.	0 -2 3 -4	0	2 отрицательных.
6.	1 -2 3 -4	2	
7.	2 -2 3 -4	5	
8.	3 -2 3 -4	4	
9.	0 1 -2 3	1	1 отрицательный
10.	1 1 -2 3	1	
11.	2 1 -2 3	0	
12.	3 1 -2 3	6	
13.	0 0 1 2	Данные некорректны	Нет отрицательных.
14.	1 0 1 2	Данные некорректны	
15.	2 0 1 2	0	
16.	3 0 1 2	3	
17.	-1 0 -1 2 -3 4	Данные некорректны	Неверный код задачи.
18.	4 0 -1 2 -3 4	Данные некорректны	
19.	0 1 2 … 99 -100	99	Массив на 100, вместо … ост. числа.

Выводы.

Был изучен процесс сборки программ на языке С.

Разработан проект с make-файлом из исходного кода ЛР №1, функции выделены в отдельные файлы и к ним написаны заголовочные файлы.

Получена программа, которая выполняет считывание с клавиатуры исходных данных и команды пользователя (кода задачи). Ввод строки с неизвестным количеством чисел организован с помощью цикла for. Для обработки команд пользователя использовался оператор switch, позволяющий также выполнить примитивную проверку корректности введённого кода. Для поиска соответствующих значений в вычисляющих функциях применялись циклы for и условные операторы if. Во избежание возникновения исключительных ситуаций при выводе индекса несуществующего элемента в главной функции добавлена обработка оператором if-else.

Не проведена полная проверка на соответствие введённых данных заявленным требованиям (такая задача не формулировалась, но программа из-за этого становится опасной в применении – например, может обращаться к невыделенной памяти, если будет введено более 100 чисел массива).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```c
#include <stdio.h>
#include "index_first_negative.h"
#include "index_last_negative.h"
#include "sum_between_negative.h"
#include "sum_before_and_after_negative.h"
#define LIMIT 100 //Limit of array.
#define ERROR "Данные некорректны\n" //Universal error message.

int main(){
    //Scan type of task and array of int.
    int type; scanf("%d\n", &type);
    int arr[LIMIT], count; char gap;
    for (count = 0, gap = ' '; gap != '\n'; count++)
        scanf("%d%c", &arr[count], &gap);

    //Print result of current task.
    int res_to_print; //To avoid calling funcs twice.
    switch (type){
        case 0:
            res_to_print = index_first_negative(arr, count);
            if (res_to_print < count)
                printf("%d\n", res_to_print);
            else printf(ERROR);
            break;
        case 1:
            res_to_print = index_last_negative(arr, count);
            if (res_to_print >= 0)
                printf("%d\n", res_to_print);
            else printf(ERROR);
            break;
        case 2:
            printf("%d\n", sum_between_negative(arr, count));
            break;
        case 3:
            printf("%d\n",
                        sum_before_and_after_negative(arr, count));
            break;
        default:
            printf(ERROR);
    }

    return 0;
}
```

Название файла: index_first_negative.h

```
#ifndef ___index_first_negative___
    #define ___index_first_negative___
    int index_first_negative(int*, int);
#endif
```

Название файла: index_first_negative.c

```
#include "index_first_negative.h"

int index_first_negative(int* arr, int count){
    int i;
    for (i = 0; i < count; i++)
        if (arr[i] < 0) break;
    return i;
    //If i = count, there aren't negatives.
}
```

Название файла: index_last_negative.h

```
#ifndef ___index_last_negative___
    #define ___index_last_negative___
    int index_last_negative(int*, int);
#endif
```

Название файла: index_last_negative.c

```
#include "index_last_negative.h"

int index_last_negative(int* arr, int count){
    int i;
    for (i = count-1; i >= 0; i--)
        if (arr[i] < 0) break;
    return i;
    //If i < 0, there aren't negatives.
}
```

Название файла: sum_between_negative.h

```
#ifndef ___sum_between_negative___
    #define ___sum_between_negative___
    int sum_between_negative(int*, int);
#endif
```

Название файла: sum_between_negative.c

```
#include <stdlib.h>
#include "index_first_negative.h"
#include "index_last_negative.h"
#include "sum_between_negative.h"

int sum_between_negative(int* arr, int count){
    int i, sum = 0;
    for (i = index_first_negative(arr, count);
                        i < index_last_negative(arr, count); i++)
        sum += abs(arr[i]);
    return sum;
    //If there aren't negatives or there's only one, sum = 0.
}
```

Название файла: sum_before_and_after_negative.h

```
#ifndef ___sum_before_and_after_negative___
    #define ___sum_before_and_after_negative___
    int sum_before_and_after_negative(int*, int);
#endif
```

Название файла: sum_before_and_after_negative.c

```
#include <stdlib.h>
#include "index_first_negative.h"
#include "index_last_negative.h"
#include "sum_before_and_after_negative.h"

int sum_before_and_after_negative(int* arr, int count){
    int i, sum = 0;
    for (i = 0; i < count; i++)
        if (i < index_first_negative(arr, count) ||
                            i >= index_last_negative(arr, count))
            sum += abs(arr[i]);
    return sum;
    //If there aren't negatives or there's only one,
                return sum of absolute values all array elements.
}
```

Название файла: Makefile

```
all:
	menu

menu:
	menu.o index_first_negative.o index_last_negative.o
	sum_between_negative.o sum_before_and_after_negative.o
	gcc
	menu.o index_first_negative.o index_last_negative.o
	sum_between_negative.o sum_before_and_after_negative.o
	-o menu

menu.o:
	menu.c index_first_negative.h index_last_negative.h
	sum_between_negative.h sum_before_and_after_negative.h
	gcc -c
	menu.c

index_first_negative.o:
	index_first_negative.c index_first_negative.h
	gcc -c
	index_first_negative.c

index_last_negative.o:
	index_last_negative.c index_last_negative.h
	gcc -c
	index_last_negative.c

sum_between_negative.o:
	sum_between_negative.c index_first_negative.h
	index_last_negative.h sum_between_negative.h
	gcc -c
	sum_between_negative.c

sum_before_and_after_negative.o:
	sum_before_and_after_negative.c index_first_negative.h
	index_last_negative.h sum_before_and_after_negative.h
	gcc -c
	sum_before_and_after_negative.c

clean:

	rm -rf
	*.o menu
```