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. 3388 _____ Трунов Б.Г.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2023

Цель работы.

Изучить сборку программ, заголовочные файлы и работу с функциями в Си. Применить полученные знания на практике, реализовав программный код.

Задание.

Вариант №1

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 20. 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 произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). (multi_between_negative.c) 3 : Найти произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). (multi_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.

Файлы:

data_reading.c — файл содержащий функцию для получения данных из стандартного потока ввода.
data_reading.h — заголовочный файл с прототипом функции.

index_first_negative.c - файл содержащий функцию для получения индекса первого отрицательного числа в массиве.
index_first_negative.h — заголовочный файл с прототипом функции.

index_last_negative.c - файл содержащий функцию для получения индекса последнего отрицательного числа в массиве.
index_last_negative.h — заголовочный файл с прототипом функции.

menu.c — файл содержащий главную функцию main.

multi_between_negative.c - файл содержащий функцию для нахождения произведения элементов массива от первого отрицательного(включительно) до последнего отрицательного (не включительно).
multi_between_negative.h — заголовочный файл с прототипом функции.

multi_before_and_after_negative.c — файл содержащий функцию для произведения элементов массива от начала до первого отрицательного(не включительно) и от последнего отрицательного(включительно) до конца.
multi_before_and_after_negative.h — заголовочный файл с прототипом функции.

print_solve.c — файл содержащий функцию для вывода ответа в стандартный поток вывода.
print_solve.h — заголовочный файл с прототипом функции.

makefile — файл с инструкциями для утилиты make, которая нужна для автоматической сборки проекта.

Подключенные библиотеки:
stdio.h — для реализации ввода/вывода.

Индетификаторы define:
MAX_SIZE_ARR 20 - максимальный размер массива, исходя из условия.

Переменные:
int array[MAX_SIZE_ARR] — массив чисел из стандартного потока ввода.
int operation — номер операции введенный пользователем.
int array_count_items— количество элементов массива array.
int multi_result — произведений элементов массива функций в функциях multi_between_negative, multi_before_and_after_negative.

Функции:
void data_reading(int arr[], int * operation, int * arr_count_items) - принимает массив, указатели на operation и arr_count_items. Читает данные из stdin, заполняет массив, разыменовывая operation и arr_count_items записывает в них значение.
int index_first_negative(int arr[], int arr_count_items) — принимает массив, количество элементов массива. Возвращает индекс первого отрицательного элемента в данном массиве.
int index_last_negative(int arr[], int arr_count_items) — принимает массив, количество элементов массива. Возвращает индекс последнего нуля в данном массиве.
int main() - ничего не принимает. В ней вызываются другие функции, с помощью которых осуществляется основная работа программы.

int multi_between_negative(int arr[], int arr_count_items) — принимает массив, количество элементов массива. Возвращает произведение элементов массива от первого отрицательного до последнего отрицательного индекса.

int multi_before_and_after_negative(int arr[], int arr_count_items) — принимает массив, количество элементов массива. Возвращает произведение элементов массива от элемента с индексом 0 до первого отрицательного(не включительно) и от последнего отрицательного(включительно) до конца массива.

void print_solve(int arr[], int operation, int arr_count_items) — принимает массив, номер операции, количество элементов массива .В зависимости от номера операции 0,1,2,3 выводит результат соответствующей функции.В случае неизвестного номера операции в стандартный поток вывода stdout выводится «Данные некорректны».

Разработанный программный код см. в приложении А

Тестирование.

Результаты тестирования представлены в табл. 1.

№ п/п	Входные данные	Выходные данные	Комментарии
1	0 -5 -3 -5 -8 3 -9 -3	0	Индекс первого отрицательного числа - 0
2	1 -21 10 0 -23 -7 -15 -14 8 -9 10 -13 -14 -27 0 -7 12 -15	16	Индекс последнего отрицательного числа - 16
3	2 29 1 2 -3 4 5 6 -1 2 1 89	-360	Произведение элементов от 3 до 6 индексов - (-360)
4	3 29 1 2 -3 4 5 6 -1 1 2	-116	Произведение элементов (от 0 до 2) и (от 7 до 9) индексов - -116
5	4 -21 120 0 -23 -7 -15 -14 8 -9 10 -123 -14 -127 0 -7 12 -318	Данные некорректны	Функции, которой бы соответствовал бы номер 4 не существует

Таблица 1 – Результаты тестирования.

Выводы.

Были исследованы: сборка программ, работа с заголовочными файлами, использование функций, написанных лично и импортированных из стандартных библиотек.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для обработки команд пользователя использовался оператор множественного выбора switch.

ПРИЛОЖЕНИЯ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: makefile

```
all: menu.o data_reading.o index_first_negative.o index_last_negative.o print_solve.o multi_between_negative.o multi_before_and_after_negative.o
	gcc menu.o data_reading.o index_first_negative.o index_last_negative.o print_solve.o multi_between_negative.o multi_before_and_after_negative.o -o menu
menu.o: menu.c data_reading.h print_solve.h
	gcc -c menu.c
data_reading.o: data_reading.c data_reading.h
	gcc -c data_reading.c
print_solve.o: print_solve.c print_solve.h index_first_negative.h index_last_negative.h
	gcc -c print_solve.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
multi_between_negative.o: multi_between_negative.c multi_between_negative.h index_first_negative.h index_last_negative.h
	gcc -c multi_between_negative.c
multi_before_and_after_negative.o: multi_before_and_after_negative.c multi_before_and_after_negative.h index_first_negative.h index_last_negative.h
	gcc -c multi_before_and_after_negative.c
clean:
	rm -f *.o
	rm -f *.out
```

Название файла: menu.c

```
#include "data_reading.h"
#include "print_solve.h"
#define MAX_SIZE_ARR 20
int main(){
	int array[MAX_SIZE_ARR];
```

```
	int operation = -1;
	int array_count_items = 0;
	data_reading(array, &operation, &array_count_items);
	print_solve(array, operation, array_count_items);
	return 0;
}
```

Название файла: data_reading.c

```
#include <stdio.h>
#include "data_reading.h"
#define MAX_SIZE_ARR 20
void data_reading(int arr[], int * operation, int * arr_count_items){
	scanf("%d", operation);
	do{
		if (*arr_count_items < MAX_SIZE_ARR){
			scanf("%d", &arr[(*arr_count_items)++]);
		}
	}while(getchar() != '\n');
}
```

Название файла: data_reading.h

```
#ifndef DATA_READING_HEADER
#define DATA_READING_HEADER

void data_reading(int arr[], int * operation, int * arr_count_items);

#endif
```

Название файла: index_first_negative.c

```
#include "index_first_negative.h"

int index_first_negative(int arr[], int arr_count_items){
	for (int x = 0; x < arr_count_items; x++){
		if (arr[x] < 0){
			return x;
			break;
		}
	}
	return -1;
}
```

Название файла: index_first_negative.h

```
#ifndef INDEX_FIRST_NEGATIVE_HEADER
#define INDEX_FIRST_NEGATIVE_HEADER

int index_first_negative(int arr[], int arr_count_items);

#endif
```
Название файла: index_last_negative.c

```
#include "index_last_negative.h"

int index_last_negative(int arr[], int arr_count_items){
      for (int x = arr_count_items - 1; x > 0; x--){
            if (arr[x] < 0){
                  return x;
                  break;
            }
      }
      return -1;
}
```
Название файла: index_last_negative.h

```
#ifndef INDEX_LAST_NEGATIVE_HEADER
#define INDEX_LAST_NEGATIVE_HEADER

int index_last_negative(int arr[], int arr_count_items);

#endif
```
Название multi_between_negative.c

```
#include "multi_between_negative.h"
#include "index_first_negative.h"
#include "index_last_negative.h"

int multi_between_negative(int arr[], int arr_count_items){
      int first_negative = index_first_negative(arr, arr_count_items);
      int last_negative = index_last_negative(arr, arr_count_items);
      int multi_result = 1;
      for (int x = first_negative; x < last_negative; x++){
            multi_result *= arr[x];
      }
```

```
    return multi_result;
}
```
Название файла: multi_between_negative.h

```
#ifndef MULTI_BETWEEN_NEGATIVE_HEADER
#define MULTI_BETWEEN_NEGATIVE_HEADER

int multi_between_negative(int arr[], int arr_count_items);

#endif
```
Название файла: multi_before_and_after_negative.c

```
#include "multi_before_and_after_negative.h"
#include "index_first_negative.h"
#include "index_last_negative.h"

int multi_before_and_after_negative(int arr[], int arr_count_items){
    int first_negative = index_first_negative(arr, arr_count_items);
    int last_negative = index_last_negative(arr, arr_count_items);
    int multi_result = 1;
    for (int x = 0; x < first_negative; x++){
        multi_result *= arr[x];
    }
    for (int x = last_negative; x < arr_count_items; x++){
        multi_result *= arr[x];
    }
    return multi_result;
}
```
Название файла: multi_before_and_after_negative.h

```
#ifndef MULTI_BEFORE_AND_AFTER_NEGATIVE_HEADER
#define MULTI_BEFORE_AND_AFTER_NEGATIVE_HEADER

int multi_before_and_after_negative(int arr[], int arr_count_items);

#endif
```
Название файла: print_solve.c

```
#include <stdio.h>
#include "print_solve.h"
```

```
#include "index_first_negative.h"
#include "index_last_negative.h"
#include "multi_between_negative.h"
#include «multi_before_and_after_negative.h"
#define PRINT_DOUBLE "%d\n"

void print_solve(int arr[], int operation, int arr_count_items){
     switch(operation){
          case 0:
               printf(PRINT_DOUBLE,  index_first_negative(arr,
arr_count_items));
               break;
          case 1:
               printf(PRINT_DOUBLE,  index_last_negative(arr,
arr_count_items));
               break;
          case 2:
               printf(PRINT_DOUBLE,  multi_between_negative(arr,
arr_count_items));
               break;
          case 3:
             printf(PRINT_DOUBLE, multi_before_and_after_negative(arr,
arr_count_items));
               break;
          default:
               printf("Данные некорректны\n");
               break;
     }
}
```

Название файла: print_solve.h

```
#ifndef PRINT_SOLVE_HEADER
#define PRINT_SOLVE_HEADER

void print_solve(int arr[], int operation, int arr_count_items);

#endif
```