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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ ЯЗЫКА СИ

Студент гр. 3341 _____ Мальцев К.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2023

Цель работы

Целью работы является освоение работы с управляющими конструкциями на языке Си на примере использующей их программы.

Задание

Вариант 4

Напишите программу, выделив каждую подзадачу в отдельную функцию.

Реализуйте программу, 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 массиве есть хотя бы один четный и нечетный элемент.

В зависимости от значения, функция должна выводить следующее:

0 : индекс первого чётного элемента. (index_first_even)

1 : индекс последнего нечётного элемента. (index_last_odd)

2 : Найти сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний. (sum_between_even_odd)

3 : Найти сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент). (sum_before_even_and_after_odd)

иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения

Если функция описана в другом модуле, то, для ее использования модуль должен быть подключен к текущему модулю. Для подключения модуля используемся директива *#include*. Формат директивы: *#include <ИмяФайла>*

Были использованы следующие библиотеки:

stdlib.h — заголовочный файл стандартной библиотеки C общего назначения. Функции данной библиотеки, использованные в программе: *abs()* (возвращает модуль числа), *atoi()* (конвертирует строку в числовой вид)

stdio.h — заголовочный файл стандартной библиотеки C, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода. Функции данной библиотеки, использованные в программе: *fgets()* (считывает символы из текущей позиции потока), *printf()* (вывод форматированной строки в стандартный поток вывода)

string.h — заголовочный файл стандартной библиотеки C, содержащий функции для работы со строками. Функции данной библиотеки, использованные в программе: *strtok()* (последовательность вызовов функции разбивает строку на лексемы)

Выполнение работы

Создаётся макрос *#define ARR_SIZE 100*

Определяются следующие функции:

1. *void fill_array(char[], int[], int*);*
2. *void allocator(char, int[], int);*
3. *int index_first_even(int[], int);*
4. *int index_last_odd(int[], int);*
5. *int sum_between_even_odd(int[], int);*
6. *int sum_before_even_and_after_odd(int[], int);*

Создаётся переменная *buf_size* (тип *int*), равная 2048 (т.к. в условии явно не сказана длина вводимой последовательности и не даны ограничения на размер чисел в массиве, было взято за основу, что числа в массиве вмещаются в тип *int*, т.е. -2147483648 ≤ *array[i]* ≤ 2147483648. Следовательно, максимально возможная длина последовательности в данном случае 100*(11+1)+1+1 = 1202. Ближайшая степень 2 сверху от полученного числа – 2048)

В переменную *buf* (массив типа char, размера *buf_size*) записывается входная строка с помощью *fgets()*.

В переменную *type* (тип *char*) записывается *buf[0]* (по условию задачи, в зависимости от её значения должна быть решена одна из подзадач). Данное условие нужно для того, чтобы проверить входное значение (т.к. ожидается один символ ‘0’, ‘1’, ‘2’ или ‘3’, то в переменную type записывается buf[0]. Также нужно выполнить проверку, что первая введенная лексема не является строкой длины больше 1 (‘abc’, ‘2qwe’ и т.д.)

Выполняется проверка *buf[1]*: *if (buf[1] != ' ')*. Если условие выполнилось, то переменной *type* присваивается *‘-‘* (после данного преобразования программа гарантированно выведет строку “Данные некорректны”)

Задаётся переменная *array* (массив типа *int*, размера *ARR_SIZE*), где будет храниться, введенный массив чисел.

Задаётся переменная *size* (тип *int*), изначально равная 0. Данная переменная является счётчиком, который будет отображать сколько чисел было введено в массив.

Далее отрабатывает функция *fill_array(buf, array, &size)*, которая разбивает *buf* на пробелы, переводит полученные лексемы в тип int, записывает их в массив *array*, увеличивает счётчик *size*.

Далее отрабатывает функция *allocator(type, array, size)*, которая в зависимости от значения переменной *type* выводит ответ на одну из подзадач или выводит строку “Данные некорректны”.

Функции:

1. *void fill_array(char buf[], int array[], int *size)*

Получает на вход массив *buf* (тип *char*), массив array (тип *int*), указатель на переменную size (тип *int*). Создаётся переменная **substr* (тип *char*), куда будут записываться считанные лексемы. Далее с помощью функции *strtok()* разбивает массив *buf* на лексемы по пробелам, с помощью *atoi()* конвертирует лексему в число и записывает её в массив *array* на позицию *size*. После чего увеличивает счётчик *size* на 1. Причём первая считанная лексема пропускается, т.к. она является значением переменной *type*, описанной выше.

2. *void allocator(char type, int array[], int size)*

Получает на вход переменную *type* (тип *char*), массив *array* (тип *int*), переменную *size* (тип *int*). С помощью оператора множественного выбора *switch()* сравнивает *type* со списком константных выражений {0, 1, 2, 3}

Если *type* = 0, то выводится результат функции *index_first_even(array, size)*

Если *type* = 1, то выводится результат функции *index_last_odd(array, size)*

Если *type* = 2, то выводится результат функции *sum_between_even_odd(array, size)*

Если *type* = 3, то выводится результат функции *sum_before_even_and_after_odd(array, size)*

Если *type* не равен ни одному из перечисленных выше значений, то выводится строка "Данные некорректны"

3. int index_first_even(int array[], int size)

Получает на вход массив *array* (тип *int*), переменную *size* (тип *int*). Возвращает индекс первого чётного элемента. Индекс находится путём прохода по массиву циклом: *for (int i = 0; i < size; i++){...}*. Внутри тела цикла выполняется проверка на чётность очередного *array[i]*: *if(abs(array[i]) % 2 == 0)*. В данной задаче в массиве могут оказаться отрицательные числа, поэтому для корректной проверки на чётность берётся остаток от деления модуля *array[i]* на 2. Если условие выполнилось, то выполняется *return i* и выход из функции. (В задаче гарантировано, что в массиве есть хотя бы один чётный и нечётный элемент, поэтому данное условие гарантировано будет выполнено)

4. int index_last_odd(int array[], int size)

Получает на вход массив *array* (тип *int*), переменную *size* (тип *int*). Возвращает индекс последнего нечётного элемента. Работает аналогично функции *int index_first_even(int array[], int size)*. Отличие заключается в том, что цикл запускается с конца *for (int i = size-1; i >= 0; i--){...}*, и в теле цикла заменено условие проверки на чётность на проверку на нечётность: *if (abs(array[i]) % 2 != 0)*.

5. int sum_between_even_odd(int array[], int size)

Получает на вход массив *array* (тип *int*), переменную *size* (тип *int*). Возвращает сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний. Создаются две переменные типа *int*: *ife = index_first_even(array, size)* и *ilo = index_last_odd(array, size)*. В них хранится первое вхождение чётного элемента и последнее вхождение нечётного элемента, соответственно. Создаётся переменная *sm* (тип *int*), равная 0. В ней будет храниться искомая сумма. Далее запускается цикл: *for (int i = ife; i < ilo; i++){...}*, в теле которого изменяется переменная *sm*:

sm += abs(array[i]). После цикла функция возвращает значение переменной *sm*.

6. *int sum_before_even_and_after_odd(int array[], int size)*

Получает на вход массив *array* (тип *int*), переменную *size* (тип *int*). Возвращает сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент). Работает аналогично функции *int sum_between_even_odd(int array[], int size)*. Отличие заключается в том, что вместо одного цикла *for (int i = ife; i < ilo; i++){...}* используется 2 цикла *for (int i = 0; i < ife; i++){...}* и *for (int i = ilo; i < size; i++){...}*. В телах обоих циклов изменяется переменная *sm*: *sm += abs(array[i])*. После циклов функция возвращает значение переменной *sm*.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 5 -31 -12 4 18	2	—
2.	1 9 17 -23 4 11 6	4	—
3.	2 37 42 15 8 -45 9	110	—
4.	3 -5 9 4 133 200 15 76	105	—
5.	7 9 8 15 3	Данные некорректны	—

Выводы

Была освоена работа с управляющими конструкциями на языке Си, на практическом примере программы, где эти конструкции использовались.

Разработана программа, решающая следующую задачу (вариант 4). Исходные данные представляют одно из значений {0, 1, 2, 3} и массив целых чисел. Числа разделены пробелами. В зависимости от введенного значения программа должна решить одну из следующих подзадач:

0 : найти индекс первого чётного элемента. (index_first_even)

1 : найти индекс последнего нечётного элемента. (index_last_odd)

2 : найти сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний. (sum_between_even_odd)

3 : найти сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент)

Далее программа выводит найденный результат или выводит строку “Данные некорректны”, если значение не является ни одним из чисел списка {0, 1, 2, 3}.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define ARR_SIZE 100

void fill_array(char[], int[], int*);
void allocator(char, int[], int);
int index_first_even(int[], int);
int index_last_odd(int[], int );
int sum_between_even_odd(int[], int);
int sum_before_even_and_after_odd(int[], int);

int main() {
    int buf_size = 2048;
    char buf[buf_size];
    fgets(buf, buf_size, stdin);

    char type;
    type = buf[0];
    if (buf[1] != ' ') type = '-';

    int array[ARR_SIZE];
    int size = 0;

    fill_array(buf, array, &size);
    allocator(type, array, size);
    return 0;
}

void fill_array(char buf[], int array[], int *size) {
    char *substr;
    substr = strtok(buf, " ");
    substr = strtok(NULL, " ");

    while (substr != NULL) {
        array[(*size)++] = atoi(substr);
        substr = strtok(NULL, " ");
    }
}

void allocator(char type, int array[], int size) {
    switch(type) {
    case '0':
        printf("%d\n", index_first_even(array, size));
        break;
    case '1':
        printf("%d\n", index_last_odd(array, size));
        break;
    case '2':
```

```
            printf("%d\n", sum_between_even_odd(array, size));
            break;
        case '3':
            printf("%d\n", sum_before_even_and_after_odd(array, size));
            break;
        default:
            printf("Данные некорректны\n");
        }
}

int index_first_even(int array[], int size) {
    for (int i = 0; i < size; i++) {
        if (abs(array[i]) % 2 == 0) return i;
    }
}

int index_last_odd(int array[], int size) {
    for (int i = size-1; i >= 0; i--) {
        if (abs(array[i]) % 2 != 0) return i;
    }
}

int sum_between_even_odd(int array[], int size) {
    int ife = index_first_even(array, size);
    int ilo = index_last_odd(array, size);
    int sm = 0;
    for (int i = ife; i < ilo; i++) {
        sm += abs(array[i]);
    }
    return sm;
}

int sum_before_even_and_after_odd(int array[], int size) {
    int ife = index_first_even(array, size);
    int ilo = index_last_odd(array, size);
    int sm = 0;
    for (int i = 0; i < ife; i++) {
        sm += abs(array[i]);
    }
    for (int i = ilo; i < size; i++) {
     sm += abs(array[i]);
    }
    return sm;
}
```