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 языке Си

Студент гр. 3384 _____ Горский К. Д.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2023

Цель работы.

Изучить процесс сборки программ из нескольких файлов с исходным кодом на языке Си при помощи утилиты make, разработав некоторую программу и Makefile для ее сборки.

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".

Выполнение работы.

Для организации ввода-вывода была использована стандартная библиотека Си (подключен заголовочный файл <*stdio.h*>).

Для вычисления абсолютного значения целого числа была использована функция *int abs(int)* из <*stdlib.h*>

Для размера массива и индексов был использован тип данных *size_t* из *<stdint.h>*.

Файл *menu.c* содержит функцию *int main()* – точку входа в программу. В начале файла додключаются заголовочные файлы *index_first_negative.h, index_last_negative.h*, *sum_before_and_after_negative.h*, *sum_between_negative.h*. Они объявляют одноименные функции, использующиеся далее.

В *main* сначала производится считывание входных данных: в переменную *choice* считывается целое число, определяющее команду из условия задания; далее с помощью цикла *while* элементы массива поочередно считываются в переменную *tmp*, после чего добавляются в конец массива *arr*. В переменную *separator* считывается символ, стоящий после очередного введенного числа. Если считывается символ переноса строки, то ввод массива завершается. В переменной *n_read* содержится результат очередного вызова *scanf* – количество полей, значения которых были действительно присвоены переменным. Если функции не удается считать число или символ, стоящий за числом (т. е. значение *n_read* не равно 2), ввод считается некорректным. Ввод всех входных данных сопровождается проверками на их корректность. Если данные некорректны, то программа сообщает об этом и завершает работу. Переменная *len* содержит длину массива.

Далее происходит проверка массива на наличие как минимум одного отрицательного числа при помощи цикла f*or*. В нем при обнаружении отрицательного числа переменной is_valid, инициализированной нулем (ложь), присваивается значение 1 (истина). Эта переменная показывает, содержится ли в массиве как минимум одно отрицательное число, необходимое для выполнения программы, или нет. Если отрицательных чисел не обнаружено, программа сообщает о некорректности данных и завершает работу.

Далее при помощи оператора *switch* программа определяет команду, которую ей необходимо выполнить. В случае *default* в консоль выводится строка «Данные некорректны» и программа завершает работу. Во всех остальных

случаях программа выводит целое число – результат функции, соответствующей данной команде, после чего также завершает работу.

В заголовочном файле *range_abs_sum.h* объявленена вспомогательная функция *int calculate_range_abs_sum(int* arr, size_t first, size_t last)*, вычисляющая сумму абсолютных значений некоторых идущих подряд элементов. В качестве аргументов принимает массив целых чисел *arr*, индекс первого элемента подпоследовательности *first*, индекс последнего элемента подпоследовательности *last*. Функция вычисляет сумму от *first* включительно до *last* не включительно. Чтобы избежать повторного объявления функций при многократном включении файлов, их объявляющих, в начале всех заголовочных файлов, включая и *range_abs_sum.h*, находится директива *#pragma once*. Она позволяет пропустить включение файла, если во время компиляции он уже был включен. Также, поскольку все **.c* файлы компилируются в объектные по-отдельности и все они используют тип данных *size_t*, будет недостаточно включить *<stdlib.h>* лишь в одном из них. Поэтому после директивы *#pragma once* в кажом заголовочном файле следует *#include <stdlib.h>*. Реализация функции *calculate_range_abs_sum* находится в файле *range_abs_sum.c*. Чтобы компилятору был известен прототип реализуемой функции (в данной лабораторной работе это не обязательно, но делать так – это хорошая практика), сначала в этом файле происходит подключение соответствующего ему заголовочного файла. После идет реализация функции: локальная переменная sum содержит промежуточную сумму, которая вычисляется обходом массива от *first* до *last* циклом *for*; конечное значение *sum* является результатом и возвращается функцией.

В файлах *index_first_negative.h* и *index_last_negative.h* аналогичным образом определены функции *index_first_negative и index_last_negative.* В соответствующих им **.c* файлам, помиимо файлов с прототипами функций, также подключается *<stdint.h>* для определения макроса *SIZE_MAX*.

size_t index_first_negative(int arr, size_t len)* – функция, соответутвующая команде «0». В качестве аргументов принимает массив целых чисел *arr* и его

длину *len*. Возвращает индекс первого отрицательного элемента массива. Если такого не существует, возвращает максимальное значение типа данных *size_t* (*SIZE_MAX*, определен в <*stdint.h*>). Функция производит обход массива слева-направо циклом *for*. Если она встречает элемент с отрицательным значением, то возвращает его индекс.

size_t index_last_negative(int arr, size_t len)* – функция, соответствующая команде «1». В качестве аргументов принимает массив целых чисел *arr* и его длину *len.* Возвращает индекс последнего отрицательного элемента массива. Если такого не существует, возвращает максимальное значение типа данных size_t. Функция происходит обход массива справо-налево циклом *for*. Если она встречает элемент с отрицательным значением, то возвращает его индекс. Стоит заметить, что переменная *i* внутри цикла меняет свое значение не от *len - 1* до *0,* а от *len* до *1*. Это сделано для того, чтобы условие внутри цикла имело смысл для беззнакового целого, которым является *i.* В первом случае условие было бы *i >= 0*, что всегда верно для любого беззнакового числа. Если цикл не завершится другим путем, то программа зависнет. Поэтому мной был использован второй вариант.

Идентично вышеупомянутым заголовочным файлам, в файлах *sum_between_negative.h* и *sum_before_and_after_negative.h* объявляются функции *sum_between_negative* и *sum_before_and_after_negative*. Реализация этих функций находися в соответствующих **.c* файлах.

int sum_between_negative(int arr, size_t len)* – функция, соответствующая команде «2». В качестве аргументов принимает массив целых чисел *arr* и его длину *len*. Возвращает сумму абсолютных значений элементов от первого отрицательного включительно до последнего отрицательного не включительно. Эта функция вычисляет сумму элементов от *first* включительно до *last* не включительно с помощью вспомогательной функции *calculate_range_abs_sum*, описанной выше. Значения *first* и *last* вычисляются вышеупомянутыми функциями *index_first_negative* и *index_last_negative*. Перед реализацией функции в файле *sum_between_negative.c* подключаются файлы с прототипами

всех необходимых функций: *sum_between_negative.h*, *index_first_negative.h*, *index_last_negative.h*, *range_abs_sum.h*.

int get_sum_before_and_after_negative(int arr, size_t len)* – функция, соответствующая команде «3». В качестве аргументов принимает массив целых чисел *arr* и его длину *len*. Возвращает сумму абсолютных значений элементов от первого элемента включительно до первого не отрицательного включительно и от последнего отрицательного включительного до последнего элемента массива включительно. Функция возвращает сумму результатов двух вызовов функций *calculate_range_abs_sum*. Первый вызов соответствует первому диапазону, второй вызов соответствует второму. Границы диапазонов вычисляются, как и в *get_sum_between_negative*, с помощью функций *index_first_negative* и *index_last_negative*. Перед реализацией функции в файле *sum_before_and_after_negative.c* подключаются файлы с прототипами всех необходимых функций: *sum_before_and_after_negative.h*, *index_first_negative.h*, *index_last_negative.h*, *range_abs_sum.h*.

Makefile начинается с опрелеления переменной *CC=gcc*. Эта переменная будет использоваться всякий раз, когда необходимо преобразовать файл и исходным кодом в объектный или собрать объектные файлы в исполняемый. Перед сборкой эту переменную можно изменить на, допустим, *clang* (компиляция пройдет без ошибок, полученная программа тесты проходит, проверено).

Далее следует строка *.PHONY: clean*. Она задает список целей, для которых не существует одноименных файла. Таким образом, файл *clean* не существует, но цель с таким именем существует и может быть выполнена.

menu – самая первая цель файла. По умолчанию выполняется именно она. Она собирает все объектные файлы в исполняемый файл *menu*. Следовательно, список ее зависимостей состоит из всех объектных файлов. Далее следует серия однотипных целей вида:

obj/.o: obj *.c *.h…*

Это цели, соответствующие объектным файлам. Каждому файлу **.c* соответствует один файл **.o* с таким же именем. Зависимостями цели в таком случае будут сам файл с исходным кодом **.c,* а также все заголовочные файлы, подключенные в нем. Если не добавить заголовочные файлы в качестве зависимости, то может получиться так, что сигнатура функции поменялась (тип аргументов, их количество или возвращаемое значение функции), а объектный файл остался неизменным и содержит функцию, принимающую на вход совсем другие аргументы, или вызов функции с совсем другими аргументами. Еще одна важная зависимость для каждого объектного файла – это директория obj. Если её не будет, тогда путь, по которому объектный файл должен быть сохранен, будет некорректным.

Цель *obj*, если директории с таким именем не существует, выполняется командой *mkdir obj*.

Самая последняя цель – это *clean.* Она используется, чтобы удалить всё, что было создано целью *menu*. Она удаляет сам исполняемый файл *menu,* а также директорию, содержащую все созданные объектные файлы.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Мной был изучен процесс сборки программ из нескольких файлов с исходным кодом на языке Си при помощи утилиты make. Была раззаботана программа и Makefile для ее сборки.

Разработанная программа выполняет считывание с клавиатуры некоторого массива целых чисел и вычисляет сумму абсолютных значений некоторых идущих подряд элементов этого массива. Для обработки команд пользователя использовался оператор *switch-case*. Для лучшей читаемости, структурированности кода и меньшего его дублирования каждой команде была отведена отдельная функция, а также была создана одна вспомогающая функция.

Каждый файл с исходным кодом компилируется отдельно от остальных, после чего происходит линковка (сборка) всех полученных объектных файлов в исполняемый.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

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

const char  err_msg[] = "Данные некорректны";
int         arr[100];
size_t      len = 0;
size_t      first, last;

int calculate_range_abs_sum(int* arr, size_t first, size_t last) {
    int sum = 0;
    for (size_t i = first; i < last; i++)
        sum += abs(arr[i]);

    return sum;
}

/* 0 */
size_t get_index_first_negative(int* arr, size_t len) {
    for (size_t i = 0; i < len; i++) {
        if (arr[i] < 0)
            return i;
    }

    return SIZE_MAX;
}

/* 1 */
size_t get_index_last_negative(int* arr, size_t len) {
    for (size_t i = len; i > 0; i--) {
        if (arr[i - 1] < 0)
            return i - 1;
    }

    return SIZE_MAX;
}

/* 2 */
int get_sum_between_negative(int* arr, size_t len) {
    return calculate_range_abs_sum(arr, first, last);
}

/* 3 */
int get_sum_before_and_after_negative(int* arr, size_t len) {
    return calculate_range_abs_sum(arr, 0, first)
         + calculate_range_abs_sum(arr, last, len);
}

int main() {
    int     choice;
    int     tmp;
    char    separator;
```

```
    int     is_valid = 0;

    if (!scanf("%d", &choice)) {
        puts(err_msg);
        return 0;
    }

    while (1) {
        int n_read = scanf("%d%c", &tmp, &separator);

        if (n_read != 2) {
            puts(err_msg);
            return 0;
        }

        arr[len++] = tmp;
        if (separator == '\n') break;
    }

    for (size_t i = 0; i < len; i++) {
        if (arr[i] < 0) {
            is_valid = 1;
            break;
        }
    }

    if (!is_valid) {
        puts(err_msg);
        return 0;
    }

    first = index_first_negative(arr, len);
    last = index_last_negative(arr, len);

    switch (choice) {
        case 0:
            printf("%ld\n", first);
            break;
        case 1:
            printf("%ld\n", last);
            break;
        case 2:
            printf("%d\n", sum_between_negative(arr, len));
            break;
        case 3:
            printf("%d\n", sum_before_and_after_negative(arr, len));
            break;
        default:
            puts(err_msg);
            break;
    }
}
```

Название файла: index_first_negative.h

```
#pragma once
#include <stdlib.h> // size_t

size_t index_first_negative(int* arr, size_t len);
```

Название файла: index_last_negative.h

```
#pragma once
#include <stdlib.h> // size_t

size_t index_first_negative(int* arr, size_t len);
```

Название файла: sum_before_and_after_negative.h

```
#pragma once
#include <stdlib.h> // size_t

int sum_before_and_after_negative(int* arr, size_t len);
```

Название файла: sum_between_negative.h

```
#pragma once
#include <stdlib.h> // size_t

int sum_between_negative(int* arr, size_t len);
```

Название файла: sum_between_negative.h

```
#pragma once
#include <stdlib.h> // size_t

int calculate_range_abs_sum(int* arr, size_t first, size_t last);
```

Название файла: sum_between_negative.h

```
#pragma once
#include <stdlib.h> // size_t

int calculate_range_abs_sum(int* arr, size_t first, size_t last);
```

Название файла: sum_between_negative.h

```
#pragma once
#include <stdlib.h> // size_t

int calculate_range_abs_sum(int* arr, size_t first, size_t last);
```

Название файла: index_first_negative.c

```
#include "index_first_negative.h"
#include <stdint.h> // SIZE_MAX

size_t index_first_negative(int* arr, size_t len) {
    for (size_t i = 0; i < len; i++) {
        if (arr[i] < 0)
            return i;
    }

    return SIZE_MAX;
}
```

Название файла: index_last_negative.c

```
#include "index_last_negative.h"
#include <stdint.h> // SIZE_MAX

size_t index_last_negative(int* arr, size_t len) {
    for (size_t i = len; i > 0; i--) {
        if (arr[i - 1] < 0)
            return i - 1;
    }

    return SIZE_MAX;
}
```

Название файла: sum_before_and_after_negative.c

```
#include "sum_before_and_after_negative.h"

#include "index_first_negative.h"
#include "index_last_negative.h"
#include "range_abs_sum.h"

int sum_before_and_after_negative(int* arr, size_t len) {
    size_t first = index_first_negative(arr, len);
    size_t last = index_last_negative(arr, len);

    return calculate_range_abs_sum(arr, 0, first)
         + calculate_range_abs_sum(arr, last, len);
}
```

Название файла: sum_between_negative.c

```
#include "sum_between_negative.h"

#include "index_first_negative.h"
#include "index_last_negative.h"
#include "range_abs_sum.h"

int sum_between_negative(int* arr, size_t len) {
    size_t first = index_first_negative(arr, len);
    size_t last = index_last_negative(arr, len);

    return calculate_range_abs_sum(arr, first, last);
}
```

Название файла: range_abs_sum.c

```
#include "range_abs_sum.h"

int calculate_range_abs_sum(int* arr, size_t first, size_t last) {
    int sum = 0;
    for (size_t i = first; i < last; i++)
        sum += abs(arr[i]);

    return sum;
}
```

Название файла: Makefile

```
CC=gcc

.PHONY: clean

menu: obj/menu.o obj/index_first_negative.o obj/index_last_negative.o obj/sum_between_negative.o obj/sum_before_and_after_negative.o obj/range_abs_sum.o
	$(CC) obj/menu.o obj/index_first_negative.o obj/index_last_negative.o obj/sum_between_negative.o obj/sum_before_and_after_negative.o obj/range_abs_sum.o -o menu

obj/index_first_negative.o: obj index_first_negative.c index_first_negative.h
	$(CC) -c index_first_negative.c -o obj/index_first_negative.o

obj/index_last_negative.o: obj index_last_negative.c index_last_negative.h
	$(CC) -c index_last_negative.c -o obj/index_last_negative.o

obj/sum_between_negative.o: obj sum_between_negative.c sum_between_negative.h index_first_negative.h index_last_negative.h
	$(CC) -c sum_between_negative.c -o obj/sum_between_negative.o

obj/sum_before_and_after_negative.o: obj sum_before_and_after_negative.c sum_before_and_after_negative.h index_first_negative.h index_last_negative.h
	$(CC) -c sum_before_and_after_negative.c -o obj/sum_before_and_after_negative.o

obj/range_abs_sum.o: obj range_abs_sum.c range_abs_sum.h
	$(CC) -c range_abs_sum.c -o obj/range_abs_sum.o

obj/menu.o: obj menu.c index_first_negative.h index_last_negative.h sum_between_negative.h sum_before_and_after_negative.h range_abs_sum.h
	$(CC) -c menu.c -o obj/menu.o

obj:
	mkdir obj

clean:
	rm -rf obj menu
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.2 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	3	Команда «0». Ввод корректный.
2.	1 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	20	Команда «1». Ввод корректный.
3.	2 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	226	Команда «2». Ввод корректный.
4.	3 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	30	Команда «3». Ввод корректный.
5.	0 i dont care and type whatever i want	Данные некорректны	Массив задан неверно.
6.	i dont care and type whatever i want	Данные некорректны	Команда задана не целым числом.
7.	2 2 4 8 10 1 3 7 9	Данные некорректны	В массиве нет отрицательных элементов.
8.	8 1 2 3 4	Данные некорректны	Команды с таким номером не существует.