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. 1382 _____ Коренев Д.А.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Изучение процесса сборки программ на языке си при помощи утилиты Make.

Задание (Вариант 4).

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.

0 : индекс первого четного элемента. (index_first.even)

1 : индекс последнего нечётного элемента. (index_last_odd.c)

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.c)

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.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.

Основные теоретические положения.

Препроцессор - это программа, которая подготавливает код программы для передачи ее компилятору. Команды препроцессора называются директивами и имеют следующий формат: #ключевое_слово параметры Основные действия, выполняемые препроцессором:

• Удаление комментариев

• Включение содержимого файлов (#include)

• Макроподстановка (#define)

• Условная компиляция (#if, #ifdef, #elif, #else, #endif)

Компиляция - процесс преобразования программы с исходного языка высокого уровня в эквивалентную программу на языке более низкого уровня (в частности, машинном языке).

Компилятор - программа, которая осуществляет компиляцию.

Линковщик (компоновщик) принимает на вход один или несколько объектных файлов и собирает по ним исполняемый модуль. Работа компоновщика заключается в том, чтобы в каждом модуле определить и связать ссылки на неопределённые имена.

Сборка проекта - это процесс получения исполняемого файла из исходного кода.

Сборка проекта вручную может стать довольно утомительным занятием, особенно, если исходных файлов больше одного и требуется задавать некоторые параметры компиляции/линковки. Для этого используются Makefile - список инструкций для утилиты make, которая позволяет собирать проект сразу целиком.

Если запустить утилиту make, то она попытается найти файл с именем Makefile в текущей директории и выполнить из него инструкции.

Выполнение работы.

Файл menu.c:

В функции main объявляются необходимые в дальнейшем переменные: task, ans, c, len, arr, а так же с помощью директивы #define определено значение для SIZE. При помощи scanf считывается значение: 0, 1, 2 или 3, и присваивается переменной task, в зависимости от которого будет вызываться та или иная функция. При помощи цикла for считываются данные, которые будут помещены в массив arr. Каждую итерацию к переменная len увеличивается на единицу, а при помощи scanf

элемент массива arr с индексом i и переменная с равны некоторому значению и знаку табуляции после него соответственно во входных данных . Если знак после значения — перенос строки, то цикл for досрочно прерывается при помощи оператора break. Далее используется оператор switch по значению task.

- Если task равен 0, выводится значение возвращаемое функцией index_first_even. Файл index_first_even.h содержит объявление функции index_first_even. В index_first_even.c содержится функция index_first_even, которая находит индекс первого четного элемента в данном нам массиве. Принимает на вход массив (int arr[]) , а так же длину (int len) — количество элементов этого массива, среди которых надо найти четный элемент. Она возвращает индекс первого четного элемента, который она находит путем перебора элементов массива, пока не встретить искомое, циклом for.
- Если - 1, выводится значение возвращаемое функцией index_last_odd. Файл index_last_odd.h – содержит объявление функции index_last_odd. В index_last_odd.c содержится функция index_last_odd, которая находит индекс последнего нечетного элемента в данном массиве. Принимает на вход массив (int arr[]) , а так же длину (int len) — количество элементов этого массива, среди которых надо найти последний нечетный элемент. Она находит путем перебора всех элементов массива, присваивая к переменной odd индекс нечетного элемента, каждый раз, когда находит такой в массиве. После завершения цикла for, переменная odd хранит в себе значение индекса последнего нечетного элемента, которое возвращает функция.
- Если 2, выводится значение возвращаемое функцией sum_between_even_odd. Файл sum_between_even_odd.h содержит объявление функции sum_between_even_odd. sum_between_even_odd.c включает содержимое файлов index_first_even.c и index_last_odd.c с помощью команд #include "index_first_even.h" и #include "index_last_odd.h". В sum_between_even_odd.c содержится функция sum_between_even_odd,

которая находит сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний. Принимает на вход массив (int arr[]), а так же длину (int len) — количество потенциальных элементов среди которых надо найти сумму их модулей. Внутри функции вызываются функции index_first_even и index_last_odd, принцип которых описан выше, и присваивает возвращаемые значения в переменные if_even и il_odd соответственно. Используя цикл for суммирует модули элементы массива начиная от элемента с индексом равным значению if_even и заканчивая элементом массива с индексом il_odd не включительно. Возвращает искомую сумму.

- Если 3, выводится значение возвращаемое функцией sum_before_even_and_after_odd. Файл sum_before_even_and_after_odd.h содержит объявление функции sum_before_even_and_after_odd. sum_before_even_and_after_odd включает содержимое файлов index_first_even.c и index_last_odd.c с помощью команд #include "index_first_even.h" и #include "index_last_odd.h". В sum_before_even_and_after_odd.c содержится функция sum_before_even_and_after_odd, которая находит сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент). Принимает на вход массив (int arr[]), а так же длину (int len) — количество потенциальных элементов среди которых надо найти сумму модулей этих элементов. Внутри функции вызываются функции index_first_even и index_last_odd, принцип которых описан выше, и присваивает возвращаемые значения в переменные if_even и il_odd соответственно. Используются два цикла for: первый - суммирует модули элементов массива начиная с элемента под нулевым индексом (т.е. Первый в массиве) и заканчивая элементом с индексом равным значению в переменной If_even не включительно, второй - суммирует модули элементов массива

начиная с элемента под индексом равным значению в переменной il_odd и заканчивая последним (т.е. элемент под индексом равным значению переменной len). Возвращает искомую сумму.

В случае по умолчанию, т.е. если значение task не было равно ни 0, ни 1, ни 2, ни 3, выводится строчка "Данные некорректны". На этом программа завершает работу.

Makefile:

1. Инструкция all:
   1) Зависимости: menu.o index_first_even.o, index_last_odd.o, sum_between_even_odd.o, sum_before_even_and_after_odd.o
   2) Команда: gcc menu.o index_first_even.o index_last_odd.o sum_between_even_odd.o sum_before_even_and_after_odd.o -o menu
   3) Что происходит: Выполнение данной инструкции приводит к сборке исполняемого файла с именем menu из объектных
2. Инструкция menu.o:
   1) Зависимости: menu.c, index_first_even.h, index_last_odd.h, sum_between_even_odd.h, sum_before_even_and_after_odd.h
   2) Команда: gcc -c menu.c
   3) Что происходит: Создаётся объектный файл menu.o
3. Инструкция index_first_even.o:
   1) Зависимости: index_first_even.c
   2) Команда: gcc –c index_first_even.c
   3) Что происходит: Создаётся объектный файл index_first_even.o
4. Инструкция index_last_odd.o:
   1) Зависимости: index_last_odd.c
   2) Команда: gcc –c index_first_even.c
   3) Что происходит: Создаётся объектный файл index_first_even.o
5. Инструкция sum_between_even_odd.o

1) Зависимости: sum_between_even_odd.c, index_last_odd.h, index_first_even.h

2) Команда: gcc –c sum_between_even_odd.c

3) Что происходит: Создаётся объектный файл sum_between_even_odd.o

6. Инструкция sum_before_even_and_after_odd.o

1) Зависимости: sum_before_even_and_after_odd.c index_first_even.h, index_last_odd.h

2) Команда: gcc –c sum_before_even_and_after_odd.o

3) Что происходит: Создаётся объектный файл sum_before_even_and_after_odd.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 -8 -23 -30 -11 -28 15 -20 - 24 -27 5 -13 5 21 -5 16 30 - 12 15 -14 -28 -27 -11 -5 4 29 -5\n	0	Программа работает правильно
2.	1 -8 -23 -30 -11 -28 15 -20 - 24 -27 5 -13 5 21 -5 16 30 - 12 15 -14 -28 -27 -11 -5 4 29 -5\n	25	Программа работает правильно
3.	2 -8 -23 -30 -11 -28 15 -20 - 24 -27 5 -13 5 21 -5 16 30 - 12 15 -14 -28 -27 -11 -5 4 29 -5\n	426	Программа работает правильно
4.	3 -8 -23 -30 -11 -28 15 -20 - 24 -27 5 -13 5 21 -5 16 30	5	Программа работает

	- 12 15 -14 -28 -27 -11 -5 4 29 -5\n		правильно
5.	4 3 -8 -23 -30 -11 -28 15 - 20 - 24 -27 5 -13 5 21 -5 16 30 - 12 15 -14 -28 -27 -11 - 5 4 29 -5\n	Данные некорректны	Программа работает правильно

Выводы.

В ходе работы был изучен процесс сборки программы на языке Си при помощи утилиты Make.

Разработана программа, которая собирается из нескольких файлов с помощью Makefile и утилиты make, выполняющая считывание исходных с помощью функции scanf и цикла while в переменную n и массив arr[SIZE], условием которого было равенство переменной c, хранящей код символа между числами, коду символа пробела, написаны функции для обработки входных результатов, подробное описание которых приведено в разделе «выполнение работы», с помощью оператора switch и функции printf реализован вывод результата определённой функции в зависимости от входного управляющего значения task:

• если 0 — выводится результат функции int index_first_even;

• если 1 — выводится результат функции int index_last_odd;

• если 2 — выводится результат функции int sum_between_even_odd;

• если 3 — выводится результат функции int sum_before_even_and_after_odd;

Если значение не соответствует ни одному из перечисленных — выводится строка «Данные некорректны»

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД КРОГРАММЫ

Название файла: menu.c

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

#include "index_first_even.h"
#include "index_last_odd.h"
#include "sum_between_even_odd.h"
#include "sum_before_even_and_after_odd.h"

#define SIZE 100

int main()
{
    int task;
    int ans;
    char c;
    int len = 0;
    int arr[SIZE];

    scanf("%d", &task);
    int i = 0;
    for (i; i < SIZE; ++i
        scanf("%d%c", &arr[i], &c);
        len ++;
        if (c == '\n'){
            break;
        }
    }

    switch (task)
    {
        case 0:
            ans = index_first_even(arr, len);
            printf("%d\n", ans);
            break;
        case 1:
            ans = index_last_odd(arr, len);
            printf("%d\n", ans);
            break;
        case 2:
            ans = sum_between_even_odd(arr, len);
            printf("%d\n", ans);
            break;
        case 3:
            ans = sum_before_even_and_after_odd(arr, len);
            printf("%d\n", ans);
            break;
        default:
            printf("Данные некорректны\n");
            break;
    }

    return 0;
```

```
}
```

Название файла: index_first_even.c

```
#include <stdio.h>

int index_first_even(int arr[], int len) {
    int i = 0;
    for (i; i < len; ++i) {
        if ((arr[i] % 2) == 0) {
            return (i);
        }
    }
}
```

Название файла: index_first_even.h

```
int index_first_even(int num[], int lenth);
```

Название файл: index_last_odd.c

```
#include <stdio.h>

int index_last_odd(int arr[], int len){
    int odd = -1;
    int i = 0;
    for (i; i < len; ++i){
        if ((arr[i] % 2) != 0){
            odd = i;
        }
    }
    return(odd);
}
```

Название файла: index_last_odd.h

```
int index_last_odd(int num[], int lenth);
```

Название файла: sum_between_even_odd.c

```
#include <stdlib.h>
#include <stdio.h>
#include "index_first_even.h"
#include "index_last_odd.h"


int sum_between_even_odd(int arr[], int len){
    int sum = 0;
    int if_even = index_first_even(arr, len);
    int il_odd = index_last_odd(arr, len);
    int i = if_even;
    for (i; i < il_odd; ++i){
        sum+=abs(arr[i]);
    }
    return(sum);
}
```

Название файла: sum_between_even_odd.h

```
int sum_between_even_odd(int num[], int lenth);
```

Название файла: sum_before_even_and_after_odd.c

```
#include <stdlib.h>
#include "index_first_even.h"
#include "index_last_odd.h"

int sum_before_even_and_after_odd(int arr[], int len){
    int sum = 0;
    int if_even = index_first_even(arr, len);
    int il_odd = index_last_odd(arr, len);
    int i = 0;
    for (i; i < if_even; ++i) {
        sum += abs(arr[i]);
    }
    i = il_odd;
    for (i; i < len; ++i) {
        sum += abs(arr[i]);
    }
    return(sum);
}
```

Название файла: sum_before_even_and_after_odd.h

```
int sum_before_even_and_after_odd(int num[], int lenth);
```

Название файла: Makefile

```
all: menu.o index_first_even.o index_last_odd.o
sum_between_even_odd.o sum_before_even_and_after_odd.o
    gcc menu.o index_first_even.o index_last_odd.o
sum_between_even_odd.o sum_before_even_and_after_odd.o -o menu

menu.o: menu.c index_first_even.h index_last_odd.h
    gcc -c menu.c

index_first_even.o: index_first_even.c
    gcc -c index_first_even.c

index_last_odd.o: index_last_odd.c
    gcc -c index_last_odd.c

sum_between_even_odd.o: sum_between_even_odd.c index_first_even.o
index_last_odd.o
    gcc -c sum_between_even_odd.c

sum_before_even_after_odd.o: sum_before_even_after_odd.c
index_first_even.o index_last_odd.o
    gcc -c sum_before_even_after_odd.c
```