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.0382 _____ Диденко Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение способов сборки программ.

Задание.

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 чётного элемента. (*index_first_even.c*)

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".

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.

Любой make-файл состоит из

- списка целей
- зависимостей этих целей

- команд, которые требуется выполнить, чтобы достичь эту цель

цель: зависимости

[tab] команда

Выполнение работы.

Исходный код решения задачи см.в приложении А.

Создается несколько файлов:

index_first_even.h – содержит объявление функции *index_first_even*.

index_first_even.c – включает содержимое файла *index_first_even.h* с помощью команды *#include "index_first_even.h"* и содержит определение функции *index_first_even*:

Функция *index_first_even* с параметрами *int A[]*, *int n* принимает в качестве аргументов массив *arr* и переменную *arr_size*. В теле функции объявляется целочисленная переменная *i = 0*, отвечающая за индекс элемента массива. Цикл *while* поочередно проверяет каждый элемент массива на четность с помощью условного оператора *if* , и заканчивает свою работу, если встречает четное число (*if(A[i]%2 == 0)*) с помощью оператора *break*, если же такое число не находится, то производится операция *i*++ (добавление единицы) и цикл повторяется. Функция возвращает значение *i* –индекс первого четного элемента массива.

index_last_odd.h – содержит объявление функции *index_last_odd*.

index_last_odd.c – включает содержимое файла *index_last_odd.h* с помощью команды *#include "index_last_odd.h"* и содержит определение функции *index_last_odd*:

Функция *index_last_odd* с параметрами *int A[]*, *int n* принимает в качестве аргументов массив *arr* и переменную *arr_size*. В теле функции объявляется целочисленная переменная *i = n-1*, отвечающая за индекс элемента массива (в данном случае логичней начинать с последнего элемента массива, имеющего индекс n-1). Цикл *while* поочередно проверяет каждый элемент массива на четность с помощью условного оператора *if* , и заканчивает свою работу, если встречает нечетное число (*if(abs(A[i])%2 == 1)*) с помощью оператора *break*,

если же такое число не находится, то производится операция *i--* (вычитание единицы) и цикл повторяется. Функция возвращает значение *i* –индекс последнего нечетного элемента массива.

sum_between_even_odd.h - содержит объявление функции *sum_between_even_odd*.

sum_between_even_odd.c – включает содержимое файла *sum_between_even_odd.h* с помощью команды *#include "sum_between_even_odd.h"* и содержит определение функции *sum_between_even_odd*:

Функция *sum_between_even_odd* с параметрами *int A[]*, *int n* принимает в качестве аргументов массив *arr* и переменную *arr_size*.В теле функции объявляются целочисленные переменные *Begin* и *End*, получающие значения из функций *index_first_even* и *index_last_odd* соответственно, которые получают аргументы *A,n* равные массиву *arr* и переменной *arr_size*. Целочисленная переменная *summ = 0*, в которую с помощью цикла *for* будет записана сумма модулей элементов массива с *Begin* включительно до *End* невключительно. Функция возвращает значение *summ*.

sum_before_even_and_after_odd.h – содержит объявление функции *sum_before_even_and_after_odd*.

sum_before_even_and_after_odd.c – включает содержимое файла *sum_before_even_and_after_odd.h* с помощью команды *#include " sum_before_even_and_after_odd.h"* и содержит определение функции *sum_before_even_and_after_odd*:

Функция *sum_before_even_and_after_odd* с параметрами *int A[]*, *int n* принимает в качестве аргументов массив *arr* и переменную *arr_size*. В теле функции объявляются целочисленные переменные *i = 0, j = 0*(счетчики); *total_sum = 0* (сумма модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент)), *first_even = index_first_even(A,n)* (хранит индекс первого четного элемента), *last_odd = index_last_odd(A,n)* (хранит индекс последнего нечетного элемента). В цикле *while* проверяется условие *i<n* и выполняется

действие *total_sum += abs(A[i])*. Если *i == first_even,* то с помощью вложенного цикла *while* пропускаем все элементы массива до *last_odd* и складываем оставшиеся. Функция возвращает значение *total_sum*.

menu.c – содержит главную функцию *main()* и связывает все вышеописанные функции с помощью директивы *#include:*

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

Подключаются библиотеки – *stdio.h* для ввода и вывода данных через функции *scanf()* и *printf()* соответственно и *stdlib.h* для нахождения модуля через функцию *abs()*. В главной функции программы объявляется целочисленный массив *arr* размерностью 100, целочисленная переменная *arr_size*, задача которой – показывать программе текущую заполненность массива значимыми элементами, целочисленная переменная *V* (version), отвечающая за дальнейшие действия программы над массивом *arr*. С помощью функции *scanf()* записываем значение в переменную V,объявляется переменная *S* типа *char*, следящая за введением данных в массив (если значение «пробел», ввод продолжается, иначе - прекращается).В следующих двух строках через цикл *while* с условиями *arr_size < 100* и *S == ' '* заполняется массив *arr.* В условном операторе *switch* проверяется переменная *V*: в случае 0 – выводится на консоль индекс первого четного элемента массива с помощью функции *index_first_even*, в случае 1 – индекс последнего нечетного элемента массива с помощью функции *index_last_odd*, в случает 2 - сумма модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний с помощью функции *sum_between_even_odd*, в случае 3 - сумма модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент) с помощью функции *sum_before_even_and_after_odd*, при любом другом значении V выводится «Данные некорректны».

В директории со всеми вышеописанными файлами создается *Makefile*, в котором прописываются команды для компиляции программы. С помощью утилиты *make* собирается программа, названная *menu.*

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 верно
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 верно
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 верно
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	5	Программа работает верно
5.	4 -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 верно
6.	0 5 -7 -73 83 -8 5 8 7 4 2 -16 -17 8 2 6\n	4	Программа работает верно
7.	1 5 -7 -73 83 -8 5 8 7 4 2 -16 -17 8 2 6\n	11	Программа работает верно

8.	2 5 -7 -73 83 -8 5 8 7 4 2 -16 -17 8 2 6\n	50	Программа работает верно
9.	3 5 -7 -73 83 -8 5 8 7 4 2 -16 -17 8 2 6\n	201	Программа работает верно

Выводы.

Были изучены способы сборки программ на языке Си.

Разработана программа, обрабатывающая последовательность введенных чисел и выводящая на консоль результат (зависит от значения переменной V), которая собирается из нескольких файлов с помощью *Makefile* и утилиты *make*.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: index_first_even.h

```
int index_first_even();
```

Название файла: index_first_even.c

```
#include "index_first_even.h"

int index_first_even(int A[],int n){
      int i = 0;
      while (i<n){
            if(A[i]%2 == 0){
                  break;
            }else
                  i++;
      }
      return i;
}
```

Название файла: index_last_odd.h

```
int index_last_odd();
```

Название файла: index_last_odd.c

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

int index_last_odd(int A[],int n){
      int i = n-1;
      while (i>=0){
            if (abs(A[i])%2 == 1){
                  break;
            }else
                  i--;
      }
      return i;
}
```

Название файла: sum_between_even_odd.h

```
int sum_between_even_odd();
```

Название файла: sum_between_even_odd.c

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

int sum_between_even_odd(int A[],int n){
      int Begin = index_first_even(A,n);
      int End = index_last_odd(A,n);
      int summ = 0;
      for (Begin; Begin < End; Begin++){
            summ += abs(A[Begin]);
```

```
      }
      return summ;
}
```

Название файла: sum_between_even_odd.h

```
int sum_between_even_odd();
```

Название файла: sum_between_even_odd.c

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

int sum_between_even_odd(int A[],int n){
      int Begin = index_first_even(A,n);
      int End = index_last_odd(A,n);
      int summ = 0;
      for (Begin; Begin < End; Begin++){
            summ += abs(A[Begin]);
      }
      return summ;
}
```

Название файла: sum_before_even_and_after_odd.h

```
int sum_before_even_and_after_odd();
```

Название файла: sum_before_even_and_after_odd.c

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

int sum_before_even_and_after_odd(int A[],int n){
        int i = 0;
        int j = 0;
        int total_sum = 0;
        int first_even = index_first_even(A,n);
        int last_odd = index_last_odd(A,n);
        while(i<n){
                if (i == first_even){
                        while (i != last_odd){
                                i++;
                        }
                }
                total_sum += abs(A[i]);
                i++;
                }
        return total_sum;
}
```

Название файла: menu.c

```
#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"

int main(){
      int arr[100];
      int arr_size = 0;
      int V;
      scanf("%d", &V);
      char S = ' ';
      while (arr_size < 100 && S == ' '){
            scanf("%d%c",&arr[arr_size++],&S);}
      switch (V){
            case 0:
            printf("%d\n", index_first_even(arr,arr_size));
            break;
            case 1:
            printf("%d\n", index_last_odd(arr,arr_size));
            break;
            case 2:
            printf("%d\n", sum_between_even_odd(arr,arr_size));
            break;
            case 3:
            printf("%d\n", sum_before_even_and_after_odd(arr,arr_size));
            break;
            default:
            printf("Данные некорректны\n");
            break;
      }
      return 0;
}
```

Название файла: Makefile.c

```
      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 -
std=c99
      menu.o: menu.c index_first_even.h index_last_odd.h
sum_between_even_odd.h sum_before_even_and_after_odd.h
            gcc -c menu.c -std=c99
      index_first_even.o: index_first_even.c index_first_even.h
            gcc -c index_first_even.c -std=c99
      index_last_odd.o: index_last_odd.c index_last_odd.h
            gcc -c index_last_odd.c -std=c99
      sum_between_even_odd.o: sum_between_even_odd.c
sum_between_even_odd.h index_first_even.h index_last_odd.h
            gcc -c sum_between_even_odd.c -std=c99
      sum_before_even_and_after_odd.o: sum_before_even_and_after_odd.c
sum_before_even_and_after_odd.h index_first_even.h index_last_odd.h
            gcc -c sum_before_even_and_after_odd.c -std=c99
      clean:
         rm *.o menu
```