**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: Создание make-файла

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Колованов Р.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

## Цель работы.

Изучение процесса создания make-файла для компиляции программ посредством утилиты make, а также разделение файла с исходным кодом программы на несколько файлов.

## 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.

Сначала пользователь вводит число (0-3), в зависимости от значения которого программа будет выводить: 0 – индекс первого отрицательного элемента, 1 – индекс последнего отрицательного элемента, 2 – сумма модулей элементов между этими индексами (включая первый и не включая последний отрицательные элементы), 3 – сумма модулей элементов, которые не попадают в промежуток из подзадачи 2. При помощи функции *scanf()* считывается введенное пользователем число и записывается в переменную ***action*** (нужна для хранения выбранной подзадачи). Далее программа попадает в цикл *do-while*, в котором будет обрабатываться последующие введенные пользователем числа. В данном цикле заполняется массив ***array*** (нужен для хранения входных данных; размер массива равен 100, так как по условию пользователь не введет больше 100 чисел) входными данными: в цикле считывается число и следующий символ после числа. Число записывается в массив, символ – в переменную ***nextSymbol*** (хранит следующий символ после введенного числа), а переменная ***arraySize*** (хранит количество заполненных элементов массива или просто количество входных данных) увеличивается на единицу. Далее производится запись значений в массив до тех пор, пока следующий символ после числа не равен символу перевода на новую строку. После окончания ввода вызывается функция ***menu()****,* где используется оператор switch для выбора нужной подзадачи (в зависимости от значения переменной ***action*** выбирается нужная функция). Результат выполнения выбранной функции записывается в переменную ***answer*** (хранит ответ на задачу). Если answer будет равен -1, значит переданные в функцию входные данные некорректны, тогда программа выдаст “Данные некорректны”. В другом случае программа выведет переменную ***answer*** на экран.

Программа содержит 5 функций, каждая из которых предназначена для выполнения определенной подзадачи:

1. *int* ***getFirstNegativeNumberIndex****(int\* array, int arraySize);*

На вход принимает указатель на массив данных и его размер. Используется для нахождения индекса первого отрицательного элемента в массиве входных данных. Возвращает индекс найденного элемента, а если элемент не найден – возвращает -1. Принцип работы: пробегается по элементам с начала массива до тех пор, пока не найдет отрицательный элемент.

1. *int* ***getLastNegativeNumberIndex****(int\* array, int arraySize);*

На вход принимает указатель на массив данных и его размер. Используется для нахождения индекса последнего отрицательного элемента в массиве входных данных. Возвращает индекс найденного элемента, а если элемент не найден – возвращает -1. Принцип работы: пробегается по элементам с конца массива до тех пор, пока не найдет отрицательный элемент.

1. *int* ***getSumBetweenNegativeNumbers****(int\* array, int arraySize);*

На вход принимает указатель на массив данных и его размер. Используется для нахождения суммы модулей элементов между первым (включая) и последним (не включая) отрицательными элементами. Возвращает найденную сумму, а если в массиве нет первого или последнего отрицательного элемента – возвращает -1. Принцип работы: пробегается между первым (включая) и последним (не включая) отрицательными элементами, добавляя каждый раз в переменную ***sum*** модуль значения текущего элемента.

1. *int* ***getSumBeforeAndAfterNegativeNumbers****(int\* array, int arraySize);*

На вход принимает указатель на массив данных и его размер. Используется для нахождения суммы модулей элементов между началом массива и первым (не включая) отрицательным элементом, последним (включая) отрицательным элементом и концом массива. Возвращает найденную сумму, а если в массиве нет первого или последнего отрицательного элемента – возвращает -1. Принцип работы: пробегается между началом массива и первым (не включая) отрицательным элементом, последним (включая) отрицательным элементом и концом массива, добавляя каждый раз в переменную ***sum*** модуль значения текущего элемента.

5) void **menu**(int action, int array[100], int arraySize);

На вход функция принимает действие, выбранное пользователем, массив входных данных и их количество. В зависимости от переданного значения **action** вызывает определенную подзадачу 0-3 (соответствующую ей функцию 1-4) и выводит результат на экран.

Каждая из пяти функций расположена в своем файле *<название\_функии>.c*, помимо этого для каждого из них существует заголовочный файл *<название\_функии>.h*. Заголовочный файл содержит объявление функции из файла с исходным кодом, директиву ***#pragma once*** для избежания повторного включения файлов, а также директивы ***#include*** для подключения нужных библиотек. В *main.c* содержится код обработки входных данных пользователя и вызов функции ***menu()***, предназначение которой описано выше.

Помимо этого в директории был создан make-файла для утилиты make, которая будет собирать программу. В make-файле определяются переменные ***objects*** и ***compile\_flags*** для избежания многократного копирования одних и тех же параметров. Основная цель ***all*** зависит от двух целей: ***build*** (компилирует программу) и ***clean*** (удаляет оставшиеся объектные файлы). ***build*** соответственно зависит от объектных файлов, которые содержатся в переменной ***objects***. Для каждого исходного файла создана отдельная цель, которая будет отвечать за компиляцию файла исходного кода в объектный файл. В ***build***-e вызывается gcc, которому подаются все объектные файлы исходного кода, который на выходе создает исполняемый файл *menu*. В переменной ***compile\_flags*** содержатся параметры, которые передаются команде gcc. После компиляции программы выполняется цель ***clean***, которая удаляет все объектные файлы при помощи команды ***rm -f \*.o***.

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 |
|  | 0 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11 | 3 |  |
|  | 1 8 0 -3 7 -3 9 223 0 0 -1 -1 -1 | 11 |  |
|  | 3 0 -9 34 -1 2 -1 3 4 5 -9 -1 | 1 |  |

## Выводы.

Были получены навыки создания make-файлов, при помощи которых можно осуществлять сборку программ посредством утилиты make.

Разработана программа, на вход которой подается одно из значений 0-3 и массив целых чисел размера не больше 100. В зависимости от введенного пользователем значения, программа может выводить следующее:

* Индекс первого отрицательного элемента.
* Индекс последнего отрицательного элемента.
* Сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент).
* Сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент).

Для обработки команд пользователя использовались условные операторы *if-else,* оператор switch. Также в программе использовались функции для разделения функционала программы на подпрограммы. Помимо этого весь исходный код программы был разделен на файлы, каждый из которых хранит определенный функционал.

# Приложение А Исходный код программы

Название файла: main.c

#include "menu.h"

int main() {

int array[100] = { 0 }, arraySize = 0, action = 0;

char nextSymbol = ' ';

scanf("%d", &action);

do {

scanf("%d%c", &array[arraySize], &nextSymbol);

arraySize++;

} while (nextSymbol != '\n');

menu(action, array, arraySize);

return 0;

}

Название файла: menu.c

#include "menu.h"

void menu(int action, int array[100], int arraySize) {

int answer = 0;

switch (action) {

case 0:

answer = index\_first\_negative(array, arraySize);

break;

case 1:

answer = index\_last\_negative(array, arraySize);

break;

case 2:

answer = sum\_between\_negative(array, arraySize);

break;

case 3:

answer = sum\_before\_and\_after\_negative(array, arraySize);

break;

default:

answer = -1;

}

if (answer == -1) {

printf("Данные некорректны\n");

} else {

printf("%d\n", answer);

}

}

Название файла: menu.h

#pragma once

#include <stdio.h>

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

#include "sum\_between\_negative.h"

#include "sum\_before\_and\_after\_negative.h"

void menu(int action, int array[100], int arraySize);

Название файла: index\_first\_negative.c

#include "index\_first\_negative.h"

int index\_first\_negative(int\* array, int arraySize) {

for (int i = 0; i < arraySize; i++)

if (array[i] < 0)

return i;

return -1;

}

Название файла: index\_first\_negative.h

#pragma once

int index\_first\_negative(int\* array, int arraySize);

Название файла: index\_last\_negative.c

#include "index\_last\_negative.h"

int index\_last\_negative(int\* array, int arraySize) {

for (int i = arraySize - 1; i >= 0; i--)

if (array[i] < 0)

return i;

return -1;

}

Название файла: index\_last\_negative.h

#pragma once

int index\_last\_negative(int\* array, int arraySize);

Название файла: sum\_before\_and\_after\_negative.c

#include "sum\_before\_and\_after\_negative.h"

int sum\_before\_and\_after\_negative(int\* array, int arraySize) {

int sum = 0;

int begin = index\_first\_negative(array, arraySize);

int end = index\_last\_negative(array, arraySize);

if (begin == -1 || end == -1)

return -1;

for (int i = 0; i < begin; i++)

sum += abs(array[i]);

for (int i = end; i < arraySize; i++)

sum += abs(array[i]);

return sum;

}

Название файла: sum\_before\_and\_after\_negative.h

#pragma once

#include <stdlib.h>

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

int sum\_before\_and\_after\_negative(int\* array, int arraySize);

Название файла: sum\_between\_negative.c

#include "sum\_between\_negative.h"

int sum\_between\_negative(int\* array, int arraySize) {

int sum = 0;

int begin = index\_first\_negative(array, arraySize);

int end = index\_last\_negative(array, arraySize);

if (begin == -1 || end == -1)

return -1;

for (int i = begin; i < end; i++)

sum += abs(array[i]);

return sum;

}

Название файла: sum\_between\_negative.h

#pragma once

#include <stdlib.h>

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

int sum\_between\_negative(int\* array, int arraySize);

Название файла: Makefile

objects = main.o menu.o index\_first\_negative.o index\_last\_negative.o sum\_between\_negative.o sum\_before\_and\_after\_negative.o

compile\_flags = -c -std=c99

all: build clean

build: $(objects)

gcc $(objects) -o menu

sum\_before\_and\_after\_negative.o:

gcc $(compile\_flags) sum\_before\_and\_after\_negative.c

sum\_between\_negative.o: sum\_between\_negative.c

gcc $(compile\_flags) sum\_between\_negative.c

index\_last\_negative.o: index\_last\_negative.c

gcc $(compile\_flags) index\_last\_negative.c

index\_first\_negative.o: index\_first\_negative.c

gcc $(compile\_flags) index\_first\_negative.c

menu.o: menu.c

gcc $(compile\_flags) menu.c

main.o: main.c

gcc $(compile\_flags) main.c

clean:

rm -rf \*.o