**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 ПРОЕКТОВ В ЯЗЫКЕ СИ

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Анисимов Д.А |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2023

## Цель работы

Целью данной работы является изучение процесса сборки программ, написанных на языке Си на примере использования make-файлов.

Для достижения поставленной цели требуется решить следующие задачи:

1) Изучить как происходит процесс компиляции и линковки с использованием компилятора gcc.

2) Изучить структуру и правила составления make-файлов.

3) Написать make-файл для сборки заданной программы.

## Задание

Вариант 1

​ 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 20. 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 произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). (multi\_between\_negative.c)

3 : Найти произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). (multi\_before\_and\_after\_negative.c)

иначе необходимо вывести строку "Данные некорректны".

## Основные теоретические положения

Make-файл - это текстовый файл, который содержит правила сборки приложения. Утилита make используется для работы с make-файлами и позволяет эффективно компилировать только измененные файлы. При запуске утилита ищет файл с именем Makefile в текущем каталоге, но можно также явно указать другой make-файл с помощью ключа -f. Основные части make-файла включают в себя цель, зависимости и команды. Цель - это файл или действие, зависимости - это файлы, от которых зависит цель. Утилита make проверяет зависимости на наличие и дату изменений, и обновляет цель только в случае, если зависимости стали новее. Это позволяет избежать перекомпиляции всего проекта при изменении только одного файла.

## Выполнение работы

В файле menu.c:

Подключается стандартная библиотека *<stdio.h>*:

*#include <stdio.h>*

Подключаются остальная библиотеки:

*#include "* *index\_first\_negative.h"*

*#include "* *index\_last\_negative.h"*

*#include "* *multi\_between\_negative.h"*

*#include "* *multi\_before\_and\_after\_negative.h"*

Создаётся переменная *b* (тип int) записывается входную строку с помощью *scanf()*. Данное условие нужно для того, чтобы проверить входное значение (т.к. ожидается одна цифра 0, 1, 2 или 3.

Задаётся переменная *g*(типа *char*) в которую записывается целое число из массива с помощью *scanf()*.

Задаётся переменная *a* (массив типа *int*, размера 20), где будет храниться, введенный массив чисел.

Задаётся переменная *j* (типа *int*), где будут храниться счётчик, который используется в цикле *do{} while()* для замены элемента номера *j* в массиве *a* на переменную *g*.

Задаются переменные *c, d, v, y* (типа *int*) для вызова функций.

Далее запускается функция *switch()*, которая принимает на вход переменную *b* и в зависимости от eё значения(0,1,2,3) выводит ответ на одну из подзадач или строку “Данные некорректны”.

Все функции записаны в файлах формата \*.c. Примечание: в файлах *multi\_before\_and\_after\_negative.c* и *multi\_between\_negative.c* также подключены библиотеки *" index\_first\_negative.h "* и *" index\_last\_negative.h "*.

Функции:

1. *index\_first\_negative(int \*x)*

Получает на вход массив *a* (тип *int*) и количество символов в массиве *j* (тип *int*). Создаётся переменная *i* (тип *int*), которая будет счётчиком. Далее с помощью функции *While()* проверяет не является ли элемент массива *a* под номером *i* равным нулю. Если не равен 0 то, увеличивает счётчик *i* на 1. Если же элемент равен 0, то прерывает цикл *While()*.После чего возвращает *i*.

2. *index\_last\_negative(int \*x,int n)*

Получает на вход массив *a* (тип *int*) и количество символов в массиве *j* (тип *int*). Создаётся переменная *i* (тип *int*), которая будет счётчиком. Создаётся переменная *index\_2* (тип *int*). Далее с помощью цикла *for()* проверяет пока счётчик меньше количества символов в массиве и если элемент массива *a* под номером *i* меньше 0 заменяет значение *index\_2* на *i*. После чего возвращает *index\_2*.

3. *multi\_between\_negative(int \*x,int n,int index\_1,int index\_2)*

Получает на вход массив *a* (тип *int*), количество символов в массиве *j* (тип *int*), значение *index\_1* (тип *int*) и значение *index\_2* (тип *int*). Создаётся счётчик *index\_3* (тип *int*). Создаётся переменная *multiplication* (тип *int*). Далее с помощью цикла *for()* пока счётчик меньше *index\_2* *multiplication* будет умножаться на элемент массива *a* под номером *index\_3*. После выполнения цикла функция выводит значение переменной *index\_3*.

1. *multi\_before\_and\_after\_negative(int \*x,int n,int index\_1,int index\_2)*

Получает на вход массив *a* (тип *int*), количество символов в массиве *j* (тип *int*), значение *index\_1* (тип *int*) и значение *index\_2* (тип *int*). Создаются счётчики *index\_4* (тип *int*), *index\_5* (тип *int*) и *s* (тип *int*). Создаётся переменная *l* (тип *int*).

Далее в первом цикле *for()* пока *s* меньше *index\_4*, *l* умножается на элемент массива под номером *s*. Во втором цикле *for()*,пока *index\_5* меньше чем *n*, *l* умножается на элемент массива *x* под номером *index\_5*. После выполнения второго цикла функция выводит значение переменной *l*.

Для решения задачи использовались заголовочные файлы. Заголовочные файлы содержат объявления функций, определенных в соответствующих файлах \*.c. Они также содержат конструкцию #ifndef … #define … … #endif, чтобы избежать "бесконечного подключения" при подключении заголовочных файлов.

#ifndef <ИМЯ ЗАГОЛОВОЧНОГО ФАЙЛА>\_H

#define <ИМЯ ЗАГОЛОВОЧНОГО ФАЙЛА>\_H

<объявление функции, используемой в соответственном файле \*.с>

#endif

Для создания проекта был написан Makefile, использующий переменные TARGET (со значением menu), CC (со значением gcc) и CFLAGS (со значением -Wall -std=gnu99). Переменная SRC содержит имена файлов шаблона \*.c, а переменная OBJ содержит их имена с измененным разрешением на .o. Основная цель make-файла – all – описывает, что для ее достижения необходимы файлы-объектники.

Для получения цели (TARGET) необходимы файлы-объектники, которые создаются с помощью семейства компиляторов(CC) и флагов (CFLAGS). Для получения объектных файлов также используется семейство компиляторов(CC) и флаги (CFLAGS).

Также в Makefile прописана команда clean, удаляющая исполняемый файл и объектники.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 0 -5 -3 -5 -8 3 -9 -3 | 0 |
|  | 2 1 2 3 4 -2 5 6 -3 7 8 | -60 |
|  | 3 0 4 -6 -5 -7 0 -6 -4 | 0 |
|  | 3 1 5 -4 -7 4 3 -1 | -5 |
|  | 7 9 8 15 3 | Данные некорректны |

## Выводы

В ходе выполнения данной работы были получены следующие результаты:

Произведено изучение процесса компиляции и линковки программ на языке C с использованием компилятора gcc. Освоены основные этапы данного процесса и принципы функционирования компилятора.

Изучена структура и правила создания make-файлов. Освежены основные компоненты make-файла, такие как цели, зависимости и команды.

Разработан make-файл для сборки определенной программы. В данном файле определены цели сборки, взаимосвязи между файлами и необходимые команды для успешной компиляции и линковки программы.

# Приложение А Исходный код программы

Название файла: menu.c

#include <stdio.h>

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

#include "multi\_between\_negative.h"

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

int main()

{

int b;

scanf("%d",&b);

char g = 0;

int a[20];

int j = 0;

do

{

scanf("%d%c",&a[j],&g);

j++;

}

while(g!='\n');

int c,d,v,y;

c=index\_first\_negative(a);

d=index\_last\_negative(a,j);

v=multi\_between\_negative(a,j,c,d);

y=multi\_before\_and\_after\_negative(a,j,c,d);

switch(b){

case 0:

printf("%d",c);

break;

case 1:

printf("%d",d);

break;

case 2:

printf("%d",v);

break;

case 3:

printf("%d",y);

break;

default:

printf("Данные некорректны");

}

return 0;

}

Название файла: Makefile

TARGET = menu

CC = gcc

CFLAGS = -Wall -std=gnu99

SRC = $(wildcard \*.c)

OBJ = $(patsubst %.c, %.o, $(SRC))

all : $(TARGET)

$(TARGET) : $(OBJ)

$(CC) $(CFLAGS) $^ -o $@

%.o : %.c

$(CC) $(CFLAGS) -c $< -o $@

clean :

rm $(TARGET) \*.o

Название файла: index\_first\_negative.c

#include "index\_first\_negative.h"

int index\_first\_negative(int \*x){

int i = 0;

while(x[i] >= 0){

i++;

}

return i;

}

Название файла: index\_first\_negative.h

#ifndef INDEX\_FIRST\_NEGATIVE\_H

#define INDEX\_FIRST\_NEGATIVE\_H

int index\_first\_negative(int \*x);

#endif

Название файла: index\_last\_negative.c

#include "index\_last\_negative.h"

int index\_last\_negative(int \*x, int n){

int index\_2;

int i;

for (i=0;i<n;i++){

if (x[i]<0){

index\_2=i;

}

}

return index\_2;

}

Название файла: index\_last\_negative.h

#ifndef INDEX\_LAST\_NEGATIVE\_H

#define INDEX\_LAST\_NEGATIVE\_H

int index\_last\_negative(int \*x, int n);

#endif

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

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

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

#include <stdlib.h>

int multi\_before\_and\_after\_negative(int \*x,int n,int index\_1,int index\_2){

int index\_4;

index\_4=index\_1;

int s=0;

int index\_5;

int l=1;

for (s=0;s<index\_4;s++){

l=l\*x[s];

}

for (index\_5=index\_2;index\_5<n;index\_5++){

l=l\*x[index\_5];

}

return l;

}

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

#ifndef MULTI\_BEFORE\_AND\_AFTER\_NEGATIVE\_H

#define MULTI\_BEFORE\_AND\_AFTER\_NEGATIVE\_H

int multi\_before\_and\_after\_negative(int \*x,int n,int index\_1,int index\_2);

#endif

Название файла: multi\_between\_negative.c

#include "multi\_between\_negative.h"

#include "index\_first\_negative.h"

#include "index\_last\_negative.h"

#include <stdlib.h>

int multi\_between\_negative(int \*x,int n,int index\_1,int index\_2){

int index\_3;

int multiplication=1;

for (index\_3=index\_1;index\_3<index\_2;index\_3++){

multiplication=multiplication\*x[index\_3];

}

return multiplication;

}

Название файла: multi\_between\_negative.h

#ifndef MULTI\_BETWEEN\_NEGATIVE\_H

#define MULTI\_BETWEEN\_NEGATIVE\_H

int multi\_between\_negative(int \*x,int n,int index\_1,int index\_2);

#endif