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 её 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*.

4. *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 *.c>

#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
1.	0 -5 -3 -5 -8 3 -9 -3	0
2.	2 1 2 3 4 -2 5 6 -3 7 8	-60
3.	3 0 4 -6 -5 -7 0 -6 -4	0
4.	3 1 5 -4 -7 4 3 -1	-5
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
```