МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Условия, циклы, оператор switch.

Студентка гр. 7381 _____ Мартьянова Н. М.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

Цель работы:

Познакомиться с оператором выбора switch, условным оператором if и циклом while.

Задача:

Реализовать 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".

Создать для проекта make-файл.

Основные управляющие конструкции языка С

- **Операторный блок** - несколько операторов, сгруппированные в единый блок с помощью фигурных скобок

{ [<*оператор 1*>...<*оператор N*>] }

- **Условный оператор:**

if (<*выражение*>**)** <*оператор 1*> [**else** <*оператор 2*>]
Если *выражение* интерпретируется как истина, то *оператор1* выполняется. Может иметь необязательную ветку **else**, путь выполнения программы пойдет в случае если выражение ложно. В языке С любое ненулевое выражение расценивается как истина.

- **Оператор множественного выбора**

switch (<выражение>**)**

{ case <*константное выражение 1*>**:** <*операторы 1*>
... **case** <*константное выражение N*>**:** <*операторы N*>
[**default:** <*операторы*>] **}**

Выполняет поочередное сравнение выражения со списком константных выражений. При совпадении, выполнение программы начинается с соответствующего оператора. В случае, если совпадений не было, выполняется необязательная ветка **default**. Важно помнить, что операторы после первого совпадения будут выполняться далее один за другим. Чтобы этого избежать, следует использовать оператор **break.**

- **Цикл с предусловием**

while (<*выражение*>**)** <*оператор*>;

На каждой итерации цикла происходит вычисление выражения и если оно истинно, то выполняется тело цикла

- **Цикл с постусловием**

do <*оператор*> **while** <*выражение*>**;**

На каждой итерации цикла сначала выполняется тело цикла, а после вычисляется выражение. Если оно истинно — выполняется следующая итерация.

- **Цикл со счетчиком**

for ([<*начальное выражение*>]**;** [<*условное выражение*>]**;** [<*выражение приращения*>]**)**
<*оператор*>

Условием продолжения цикла как и в цикле с предусловием является некоторое выражение, однако в цикле со счетчиком есть еще 2 блока — начальное выражение, выполняемое один раз перед первым началом цикла и выражение приращения, выполняемое после каждой итерации цикла. Любая из трех частей оператора **for** может быть опущена.

- **Оператор break** — досрочно прерывает выполнение цикла.
- **Оператор continue** — досрочный переход к следующей итерации цикла.

Вывод:

В ходе выполнения работы были освоены оператор множественного выбора switch,оператор прерывания цикла break, условный оператор if, цикл с условием while, цикл со счетчиком for.

Исходный код проекта:

Файл menu.c

```
#include <stdio.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 char i=0, a[100], n;
	char  m;
	do {scanf(“%d%c”, &a[i], &m); i++}
	while(m!=’\n’);
	switch(a[0]){
		Case 0: printf(“%d\n”, index_first_even(a, i)); break;
		Case 1:printf(“%d\n”, index_last-odd(a,i)); break;
		Case 2:printf(“%d\n”, sum_between_even_odd(a,i)); break;
		Case 3:printf(“%d\n”, sum_before_even_and_after_odd(a,i)); break;
		Default: printf(“Данные некорректны\n”);
	}
return 0;
}
```

Файл index_first_even.h

```
#pragma once
int index_first_even(int a[], int i);
```

Файл index_first_even.c

```
int index_first_even(int a[], int i)
{
	int n;
	for(n=1;n<i;++n){
		if((a[n]%2)==0)
		break;
	}
return (n-1);
}
```

Файл index_last_odd.h

```
#pragma once
int index_last_odd(int a[], int i);
```

Файл index_last_odd.c

```
int index_last_odd(int a[], int i)
{
	int k=i;
	for(k=i;k>=1;--k){
		if(a[k]%2)
		break;
	}
	return (k-1);
}
```

Файл sum_between_even_odd.h

```
#pragma once
int sum_between_even_odd(int a[], int i);
```

Файл sum_between_even_odd.c

```
#include<stdlib.h>
#include”index_first_even.h”
#include”index_last_odd.h”
int sum_between_even_odd(int a[], int i)
{
	int sum=0,t;
	int n=index_first_even(a,i)+1;
	int k=index_last_odd(a,i)+1;
	for(t=n;t<k;++t)
	{
		sum+=abs(a[t]);
	}
	return sum;
}
```

Файл sum_before_even_and_after_odd.h

```
#pragma once
int sum_before_even_and_after_odd(int a[], int i);
```

Файл 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 a[], int i)
{
	int sum=0,t,sum1=0,sum2=0;
	int n=index_first_even(a,i)+1;
	int k=index_last_odd(a,i)+1;

	for(t-1; t<n; ++t){sum1+=abs(a[t]);}

	for(t=k; t<i;++t){sum2+=abs(a[t]);}

	sum=sum1+sum2;

	return sum;

}
```

Makefile

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