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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: «Условия, циклы, оператор switch»

Студент гр. 7381 _____ Минуллин М. А.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

Цель работы.

Познакомиться с массивами.

Познакомиться с оператором выбора `switch`.

Познакомиться с циклами `for (;;)`, `while ()`, `do while ()`.

Познакомиться с операторами `continue`, `break` и `return`.

В текущей директории создать 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.

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 нулевого элемента. (“`index_first_zero.c`”)

`1`: индекс последнего нулевого элемента. (“`index_last_zero.c`”)

`2`: Найти сумму модулей элементов массива, расположенных от первого нулевого элемента и до последнего. (“`sum_between.c`”)

`3`: Найти сумму модулей элементов массива, расположенных до первого нулевого элемента и после последнего. (“`sum_before_and_after.c`”)

Иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения.

Заголовочные файлы, необходимые для создания проекта:

1. `<stdlib.h>` – содержит прототип функции "`int abs(int n);`", возвращающей абсолютное значение числа. Используется в определениях функций "`int sum_between(int [], int);`" и "`int sum_before_and_after(int [], int);`".

Описание:

Функция вычисляет абсолютную величину (модуль) значения, передаваемого в качестве аргумента через параметр `n`.

Параметры:

`n` – целое значение.

Возвращаемое значение:

Модуль числа `n`.

2. `<stdio.h>` – содержит прототипы функций "`int printf(const char* format [, argument]...);`" и "`int scanf(const char* format [, argument]...);`", которые используются для ввода из потока ввода и вывода в поток вывода, а так же прототипы функций "`int getc (FILE* filestream);`" и "`int ungetc(int character, FILE* filestream);`", которые возвращают символ из потока или обратно в обратно поток. Используются в определении функции "`int main();`".

"`int getc(FILE* filestream);`":

Описание:

Функция возвращает символ из потока `filestream`, на который ссылается внутренний индикатор позиции файла. Внутренний индикатор позиции в файле, после срабатывания этой функции сдвигается на один символ, таким образом, теперь он указывает на следующий символ.

Функция `getc` эквивалентна функции `fgetc` и также принимает поток через параметр, но `getc` может быть определена как макрос, поэтому, передаваемый ей аргумент, не должен быть выражением.

Параметры:

`filestream` – указатель на объект типа `FILE`, который идентифицирует поток, для выполнения с ним различных операций.

Возвращаемое значение:

Считанный символ возвращается в виде целого значения.

Если конец файла достигнут или в процессе чтения происходит ошибка, функция возвращает `EOF` и соответствующие индикаторы ошибки или конца файла устанавливаются. Вы можете использовать любую функцию `ferror` или `feof` чтобы определить, произошла ошибка или был достигнут конец файла.

"`int ungetc(int character, FILE* filestream);`":

Описание:

Функция `ungetc` возвращает только что прочитанный символ обратно в поток ввода `filestream`, через параметр `character`. Внутренний индикатор позиции файла уменьшается обратно, на предыдущее положение, так что этот символ возвращается при следующем вызове операции чтения для этого потока.

Параметр `character` может содержать любой символ, например, последний символ прочитанный из потока в предыдущей операции или любой другой. В обоих случаях, значение, полученное по следующей операции чтения является значением функции `ungetc`, независимо от символа `character`.

Обратите внимание, что данная функция влияет только на следующую операцию чтения для данного потока, а не на содержание файла, связанного с потоком, который не изменяется при любом вызове этой функции.

Если внутренний показатель конца файла `EOF` был установлен, то после вызова этой функции он очищается.

Вызовы функций `fseek`, `fsetpos` или `rewind` для потока `stream` совместно с функцией `ungetc`, будут отбрасывать любые символы назад.

Если аргумент параметра `character` — `EOF`, функция завершается, не изменяя входной поток.

Параметры:

`character` – символ, возвращаемый обратно в поток. Символ передается, как значение типа `int`.

`filestream` – указатель на объект типа `FILE`, который идентифицирует входной поток.

Возвращаемое значение:

В случае успеха, возвращается целочисленное значение символа, который был перенесен обратно в поток.

В противном случае, возвращается значение `EOF`, и поток остается неизменным.

Вывод:

Познакомился с массивами в `C`: синтаксисом, использованием, расположением в памяти, и т. д.

Познакомился с оператором выбора `switch`: синтаксисом, использованием, операторами `case`, `break` и `default`.

Познакомился с циклами `for (;;)`, `while ()`, `do while ()`: синтаксисом, использованием, операторами `continue` для перехода к следующей итерации, `break` для выхода из цикла.

Исходный код проекта.

- Файл "index_first_zero.h":

```
#pragma once

int index_first_zero(int [], int);
```

- Файл "index_first_zero.c":

```
#include "index_first_zero.h"

int index_first_zero(int arr[], int arr_size) {
    int i;
    for (i = 0; i < arr_size; ++i)
        if (arr[i] == 0)
            return i;
}
```

- Файл "index_last_zero.h":

```
#pragma once

int index_last_zero(int [], int);
```

- Файл "index_last_zero.c":

```
#include "index_last_zero.h"

int index_last_zero(int arr[], int arr_size) {
    int i;
    for (i = arr_size - 1; i >= 0; --i)
        if (arr[i] == 0)
            return i;
}
```

- Файл "sum_between.h":

```
#pragma once

int sum_between(int [], int);
```

- Файл "sum_between.c":

```
#include "sum_between.h"
```

```c
#include "index_first_zero.h"
#include "index_last_zero.h"

#include <stdlib.h>

int sum_between(int arr[], int arr_size) {
    int from = index_first_zero(arr, arr_size);
    int to = index_last_zero(arr, arr_size);
    int result = 0;
    int i;
    for (i = from; i < to; ++i)
        result += abs(arr[i]);
    return result;
}
```

- Файл "sum_before_and_after.h":

```c
#pragma once

int sum_before_and_after(int [], int);
```

- Файл "sum_before_and_after.c":

```c
#include "sum_before_and_after.h"

#include "index_first_zero.h"
#include "index_last_zero.h"

#include <stdlib.h>

int sum_before_and_after(int arr[], int arr_size) {
    int to = index_first_zero(arr, arr_size);
    int from = index_last_zero(arr, arr_size);
    int result = 0;
    int i;
    for (i = 0; i < to; ++i)
        result += abs(arr[i]);
    for (i = from; i < arr_size; ++i)
```

```
        result += abs(arr[i]);
    return result;
}
```

- Файл "menu.c":

```
#include "index_first_zero.h"
#include "index_last_zero.h"
#include "sum_between.h"
#include "sum_before_and_after.h"

#include <stdio.h>

int arr[100];
int arr_size = 0;
int query;
int value;
int c;

int main() {
    scanf("%d", &query);
    while ((c = getc(stdin)) != '\n') {
        ungetc(c, stdin);
        scanf("%d", &value);
        arr[arr_size++] = value;
    }
    switch (query) {
    case 0:
        printf("%d", index_first_zero(arr, arr_size));
        break;
    case 1:
        printf("%d", index_last_zero(arr, arr_size));
        break;
    case 2:
        printf("%d", sum_between(arr, arr_size));
        break;
```

```
        case 3:
            printf("%d", sum_before_and_after(arr, arr_size));
            break;
        default:
            printf("Данные некорректны");
        }
        return 0;
    }
```

- Файл "Makefile":

```
    objects = menu.o index_first_zero.o index_last_zero.o sum_between.o
sum_before_and_after.o
    executable = menu

    all: $(objects)
        gcc -o $(executable) $(objects)

    menu.o: index_first_zero.h index_last_zero.h sum_between.h
sum_before_and_after.h
        gcc -c menu.c

    index_first_zero.o: index_first_zero.h index_first_zero.c
        gcc -c index_first_zero.c

    index_last_zero.o: index_last_zero.h index_last_zero.c
        gcc -c index_last_zero.c

    sum_between.o: index_first_zero.h index_last_zero.h sum_between.h
sum_between.c
        gcc -c sum_between.c

    sum_before_and_after.o: index_first_zero.h index_last_zero.h
sum_before_and_after.h sum_before_and_after.c
        gcc -c sum_before_and_after.c
```

```
clean:
	rm $(objects) $(executable)
```