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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Программирование Си»
Тема: Условия, циклы, оператор switch.

Студентка гр. 0382 _____ Кривенцова Л.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучить условия, циклы, оператор switch и закрепить знания на практике посредством написания кода программы для решения конкретной задачи.

Задание.

Вариант 3.

Напишите программу, выделив каждую подзадачу в отдельную функцию.

Реализуйте программу, 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)
1 : индекс последнего нулевого элемента. (index_last_zero)
2 : Найти сумму модулей элементов массива, расположенных от первого нулевого элемента и до последнего. (sum_between)
3 : Найти сумму модулей элементов массива, расположенных до первого нулевого элемента и после последнего. (sum_before_and_after)
иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения.

Функция нахождения модуля числа находится в заголовочном файле stdlib.h стандартной библиотеки языка Си.

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.

Выполнение работы.

Ход решения:

Используется стандартная библиотека языка си и её заголовочный файл stdlib.h (для вычисления модуля)числа.

На вход программы задаётся символ(command=*getchar()*), обозначающий комманду, которую выбирает пользователь. Далее с помощью цикла вводится массив и считывается его длина в переменную *length*. При помощи оператора *switch*, в котором описаны 5 случаев (4 из которых — команда, заданная символом, и *default*, в случае если данные были введены некорректно), и в зависимости от *command* вызывается одна из четырех заданных функций.

Чтобы найти индекс первого нуля массива, в функции void *index_first_zero* организован цикл *for*, перестающий читать массив при встрече первого нуля. Тогда индекс i цикла и является индексом первого нуля — он печатается с помощью *printf*.

Чтобы найти индекс последнего нуля массива, в функции *void index_last_zero* организован цикл *for*, читающий массив от начала до конца, и перезаписывающий в переменную *last* индекс каждого последующего встреченного нуля. Конечное значение *last* печатается с помощью *printf*.

Чтобы посчитать сумму модулей элементов массива, находящихся между первым и последним нулями, в функции *void sum_between*

организовано несколько циклов. Первый цикл *for* просто читает массив с начала до первого нуля. Следующий цикл *while* читает массив с остановленного места до конца. В нём исполняется вложенный цикл *for*, который записывает в переменную *sum2* сумму модулей всех читающихся в массиве элементов. Следует проверка, если цикл *for* закончился из-за условия (встречен ноль), а не из-за окончания массива, то в переменную *sum1*, где хранится интересующая нас сумма, записывается значение переменной *sum2*, вычисленное в цикле *for*. После окончания цикла *while* печатается значение переменной *sum1*.

Чтобы посчитать сумму модулей элементов массива, находящихся до первого и после последнего нулей, в функции *void sum_before_and_after* организовано два цикла *for*. Первый цикл прибавляет к переменной *sum1* модуль значения каждого элемента массива, пока не встречен 0. Второй цикл выполняется с момента первого нуля и до конца массива. Он прибавляет к переменной *sum2* модуль значения каждого элемента массива, но если встречен 0, он обнуляет эту переменную *sum2*. После окончания второго цикла печатается сумма переменных *sum1+sum2*.

Переменные и константы:

1.Глобальная константа, задана через #define.

#define max 100 — максимальная длина массива по условию задачи.

2. Главной функции main():

char command; - переменная, в которую считывается номер команды.

int arr[max]; - массив, в который записываются числа (входные данные) для решения задачи.

int length=0; - переменная, в которую считается фактическая длинна введённого массива. Используется как счётчик, применяемая в цикле для считывания элементов массива.

char sym = ' '; вспомогательная символьная переменная для считывания элементов массива с клавиатуры через пробел.

3.Переменные функции void index_first_zero(int num[max],int length):

int i; - переменная счётчик для цикла.

4.Переменные функции void index_last_zero(int num[max], int length):

int i; - переменная счётчик для цикла.

int last=0; - переменная, хранящая индекс последнего встретившегося нуля.

5.Переменные функции void sum_between(int num[max], int length):

int i; - переменная счётчик для цикла.

int sum1=0; -переменная, в которой сохраняется сумма модулей элементов от первого, до последнего встретившегося нуля.

int sum2=0; - переменная, в которой записывается сумма модулей элементов от первого нуля, до окончания цикла.

6.Переменные функции void sum_before_and_after(int num[max], int length);

int i; - переменная счётчик для цикла.

int sum1=0; - переменная, в которой сохраняется сумма модулей элементов от начала массива, до первого нуля.

int sum2=0; - переменная, в которой сохраняется сумма модулей элементов от последнего нуля, до конца массива.

Функции:

1. main().

Функция осуществляет ввод команды и массива с клавиатуры, в зависимости от введённой команды вызывает нужную функцию. Возвращает 0 при корректной работе. Не имеет аргументов.

2. void index_first_zero(int num[max],int length).

Функция выполняет поиск первого нуля в массиве и печатает его индекс. Ничего не возвращает в качестве значения. На вход подаются два аргумента:

int num[max]; - массив, заданный в функции main() передается для обработки (поиска индекса элемента).

int length; - длина массива, посчитанная в функции main() передается для удобства выхода из цикла.

3. void index_last_zero(int num[max], int length).

Функция выполняет поиск последнего нуля в массиве и печатает его индекс. Ничего не возвращает в качестве значения. На вход подаются два аргумента:

int num[max]; - массив, заданный в функции main() передается для обработки (поиска индекса элемента).

int length; - длина массива, посчитанная в функции main() передается для удобства выхода из цикла.

4. void sum_between(int num[max], int length).

Функция считает сумму модулей элементов массива, располагающихся между первым и последним нулями. Ничего не возвращает в качестве значения. На вход подаются два аргумента:

int num[max]; - массив, заданный в функции main() передается для обработки (поиска индекса элемента).

int length; - длина массива, посчитанная в функции main() передается для удобства выхода из цикла.

5. void sum_before_and_after(int num[max], int length).

Функция считает сумму модулей элементов массива, располагающихся до первого нуля и после последнего. Ничего не возвращает в качестве значения. На вход подаются два аргумента:

int num[max]; - массив, заданный в функции main() передается для обработки (поиска индекса элемента).

int length; - длина массива, посчитанная в функции main() передается для удобства выхода из цикла.

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
1.	0 -21 10 0 -23 -7 -15 -14 8 -9 10 -13 -14 -27 0 -7 12 -18	2	Последний элемент массива ранвый нулю, имеет индекс два.
2.	2 4 -5 0 7 -2 -9 1 0 3 6	19	Сумма модулей элементов массива, стоящих между первым и последним нулём равна 19.
3.	3 91 -12 0 88 45 6 43 20 0 16 39	158	Сумма модулей элементов массива, стоящих до первого и после последнего нулей равна 158.

Выводы.

Мной были изучены основы языка Си.

Мной были изучены основные управляющие конструкции языка: условия, циклы, оператор *switch*.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для выбора команды, номер которой считывался с помощью оператора getchar, использовался оператор множественного выбора switch. Для обработки команд пользователя также использовались условные операторы *if-else и циклы while, for*. Для

отлавливания некорректных данных был отведён раздел множественного выбора *default.*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

```
#include <stdio.h>
#include <stdlib.h>
#define max 100
void index_first_zero(int num[max],int length)
{
int i;
for (i = 0; i< length && num[i]!=0; i++)
    {
    }
printf ("\n%d", i);
}

void index_last_zero(int num[max], int length)
{
int i,last=0;
for (i = 0; i < length; i++)
    {if (num[i]==0) last=i;
    }
printf ("\n%d", last);
}

void sum_between(int num[max], int length)
{
int i,sum1=0,sum2=0;
for (i = 0; num[i]!=0; i++)
    {
    }
while(i<length){
    for (i++; num[i]!=0 && i<length; i++)
        {sum2+=abs(num[i]);
        }
    if (i<length && num[i]==0){
    sum1=sum2;
    }
}
printf ("\n%d", sum1);
}

void sum_before_and_after(int num[max], int length)
{
int i,sum1=0,sum2=0;
for (i = 0; num[i]!=0; i++)
    {sum1+=abs(num[i]);
    }
for (i++; i<length; i++)
    {
    if (num[i]!=0){
        sum2+=abs(num[i]);
        }
```

```
        else {
            sum2=0;
            }
    }
printf ("\n%d", sum1+sum2);
}

int main ()
{
char command;
int arr[max],length=0;
char sym = ' ';

command = getchar ();
while (length < max && sym == ' ')
    {
    scanf ("%d%c", &arr[length++], &sym);
    }
switch(command){
case '0': index_first_zero(arr,length);
    break;
case '1': index_last_zero(arr,length);
    break;
case '2': sum_between(arr,length);
    break;
case '3': sum_before_and_after(arr,length);
    break;
default:printf ("Данные некорректны");
    break;
}

return 0;
}
```