МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

ТЕМА: Условия, циклы, оператор switch

Студентка гр. 9383 Габибов Э.Р

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2019

Цель работы.

Изучить основные управляющие конструкции языка Си, такие как: условия, циклы, оператор *switch.*

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".

Выполнение работы.

Использованные библиотеки:

1. <*stdio.h*> - стандартный заголовочный файл ввода-вывода.

2. <*stdlib.h*> - заголовочный файл, который содержит в себе функции, занимающиеся выделением памяти (использованная функция – *abs*, нахождение модуля числа).

Использованные переменные:

1. *array* (тип int) – массив элементов, оперируемых в программе. 2. *N* (тип int) – количество элементов, оперируемых в программе.

3. *sum* (тип int) – сумма элементов массива.
4. *first_zero* (тип int) – первый нулевой элемент.
5. *last_zero* (тип int) – последний нулевой элемент.
6. *fun* (тип int) – переменная, которая определяет значение функции (хранение значений от 0 до 3)
7. *C* (тип char) – переменная, которая определяет введѐнные символы после элементов (пробел).

Описание функции *index_first_zero*:

- Функция принимает на вход массив целых чисел (*array*), а также количество элементов этого массива (*N*).
- С помощью цикла for происходит перебор элементов, проверяющий каждый элемент на 0. Как только будет найден нулевой элемент, функция прекратит работу, передав в основную функцию индекс первого нулевого элемента. Если нулевого элемента в массиве нет, то функция передаст -1.

```
int index_first_zero(int array[], int N){
for (int i = 0; i < N; i++) {         if
(array[i] == 0){             return i;
        }
}       return
-1;
}
```

Описание функции *index_last_zero*:

- Функция принимает на вход массив целых чисел (*array*), а также количество элементов этого массива (*N*).
- С помощью цикла for происходит перебор элементов, проверяющий каждый элемент на 0. Как только будет найден нулевой элемент, функция прекратит работу, передав в основную функцию индекс последнего нулевого элемента. Если нулевого элемента в массиве нет, то функция передаст -1.

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

Описание функции *sum_between*:

- Функция принимает на вход массив целых чисел (*array*), а также количество элементов этого массива (*N*).
- В переменную *first_zero* присваивается индекс первого нулевого элемента, а в переменную *last_zero* присваивается индекс последнего нулевого элемента. Переменная *sum*, отвечающая за сумму элементов, обнуляется.
- С помощью цикла for происходит перебор элементов, начиная с первого нулевого элемента (не включительно) и заканчивая последним нулевым элементом (не включительно). Модули элементов, входящих в данный промежуток, суммируются. В завершении функция передаст в основную функцию значение суммы модулей элементов.

```
int sum_between(int array[], int N){
int sum = 0;
    int first_zero = index_first_zero(array, N);
int last_zero = index_last_zero(array, N);     for
(int i = first_zero; i < last_zero; i++)
{          sum += abs(array[i]);
    }
    return sum;
}
```

Описание функции *sum_before_and_after*:

- Функция принимает на вход массив целых чисел (*array*), а также количество элементов этого массива (*N*).
- В переменную *first_zero* присваивается индекс первого нулевого элемента, а в переменную *last_zero* присваивается индекс последнего нулевого элемента. Переменная *sum*, отвечающая за сумму элементов, обнуляется.
- С помощью цикла for происходит перебор элементов, начиная с первого элемента массива (включительно) и заканчивая первым нулевым элементов массива (не включительно). Модули элементов, входящие в данный промежуток суммируются.
- С помощью цикла for происходит перебор элементов, начиная с последнего нулевого элемента массива (не включительно) и заканчивая последним элементом массива (включительно). Модули элементов, входящие в данный промежуток также продолжают суммироваться в переменную *sum*.
- В завершении функция передаст в основную функцию значение суммы модулей.

```
int sum_before_and_after(int array[], int N){
int sum = 0;
    int first_zero = index_first_zero(array, N);
int last_zero = index_last_zero(array, N);
for (int i = 0; i < first_zero; i++)
{          sum += abs(array[i]);
    }
```

```
    for ( int i = N-1; i > last_zero; i--){
sum += abs(array[i]);
    }
return sum;
}
```

Описание функции *main*:

- Создается массив *array* размером в 100 элементов.
- Создается переменная *num*, в которой будет храниться количество нулей массива.
- С помощью функции *scanf* вводится значение *fun* и *C*. Если переменная *C* не будет содержать пробела (то есть числа не разделены пробелами), тогда функция выведет строку “Данные некорректны”.
- С помощью цикла *do while* происходит заполнения массива элементами, пока строка не закончится символом перевода строки (*‘\n’*). Находим количество нулевых элементов, используя условный оператор if, и записываем их в переменную *num*. Если нулевых элементов окажется меньше двух, то функция выведет строку “Данные некорректны”. Значение индекса массива увеличивается на 1.
- С помощью оператора *switch* происходит печать соответствующей функции, основываясь на значении переменной *fun*. Если же значение *fun* не соответствуют значениям от 0 до 3, тогда функция выведет строку “Данные некорректны”.

```
int main ()
{     int
array[100];     int
N = 0;     int num
= 0;     int fun;
char C;

    scanf("%d%c", &fun, &C);
if (C != ' '){
        printf("Данные некорректны\n");
return 0;

    }
do {
        scanf("%d%c", &array[N], &C);
if (array[N]==0){             num++;
        }
        N++;
    } while (C != '\n');
if (num < 2){
fun = -1;
    }
switch(fun) {
case 0:
printf("%d\n",
index_first_zero(
array, N));
break;
case 1:
printf("%d\n",
index_last_zero(a
rray, N));
break;
case 2:
printf("%d\n",
sum_between(array
, N));
break;
case 3:
printf("%d\n",
sum_before_and_af
ter(array, N));
break;
default:
printf("Данные
некорректны");
break;
    }
return 0;
```

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 2 6 7 3 0 2 1 0	4	Программа выводит индекс первого нулевого элемента
2.	1 4 -2 7 0 8 9 -5 0	7	Программа выводит индекс последнего нулевого элемента
3.	2 1 4 0 -5 3 9 -1 0 0	18	Программа выводит сумму модулей элементов массива расположенных от первого нулевого элемента и до последнего
4.	3 9 2 0 3 7 0 4 0 -2 6	19	Программа выводит сумму модулей элементов массива, расположенных до первого нулевого элемента и до последнего.

Выводы.

Я изучил основные управляющие конструкции языка Си: условия, циклы,

оператор *switch*.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для обработки команд пользователя использовались условные операторы *if* и *switch*, а также циклы *do while* и *for*. Для ввода и вывода данных использовались команды *scanf* и *printf*. Для поиска модуля числа использовалась функция *abs* из библиотеки *<stdlib.h>*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
 int index_first_zero(int array[], int N)
{     for (int i = 0; i < N; i++) {
if (array[i] == 0){             return i;
        }
}      return
-1;
}  int index_last_zero(int array[], int
N){     for (int i = N-1; i >= 0; i--){
if (array[i] == 0){             return
i;
        }
}      return
-1;
}
int sum_between(int array[], int N){
int sum = 0;
    int first_zero = index_first_zero(array, N);
int last_zero = index_last_zero(array, N);     for
(int i = first_zero; i < last_zero; i++)
{         sum += abs(array[i]);
    }
return sum;
}
int sum_before_and_after(int array[], int N){
int sum = 0;
    int first_zero = index_first_zero(array, N);
int last_zero = index_last_zero(array, N);
for (int i = 0; i < first_zero; i++)
{         sum += abs(array[i]);
    }
    for ( int i = N-1; i > last_zero; i--){
sum += abs(array[i]);
    }
return sum;
}  int main () {
int array[100];
int N = 0;     int
num = 0;     int
fun;     char C;
scanf("%d%c", &fun,
&C);     if (C != '
'){
        printf("Данные некорректны\n");
return 0;


    }
do {
```

```
        scanf("%d%c", &array[N], &C);
if (array[N]==0){             num++;
        }
        N++;
    } while (C != '\n');
if (num < 2){
fun = -1;
    }          switch(fun) {         case 0:
printf("%d\n", index_first_zero(array, N));
break;         case 1:             printf("%d\n",
index_last_zero(array, N));             break;         case
2:             printf("%d\n", sum_between(array, N));
break;         case 3:             printf("%d\n",
sum_before_and_after(array, N));             break;
default:             printf("Данные некорректны");
break;
    }
return 0;

}
```