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

Студент гр. 0382 Литягин С.М.

Преподаватель Чайка К.В., Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение управляющих конструкций языка Си

Задание.

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: максимальное по модулю число в массиве. (abs_max)
1: минимальное по модулю число в массиве. (abs_min)
2: разницу между максимальным по модулю и минимальным по модулю элементом. (diff)
3: сумму элементов массива, расположенных после максимального по модулю элемента (включая этот элемент). (sum)

иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения.

В программе использовались следующие управляющие конструкции языка:

Функции библиотеки **stdio.h**:

- **printf()** – функция вывода на консоль;
- **scanf()** – функция ввода данных из консоли.

Функция библиотеки **stdlib.h**:

- **abs()** – функция получения модуля числа.

Циклы:

- **while(){}** – каждая итерация проверяет, выполняется ли условие в круглых скобках, если оно верно, то выполняется код в фигурных скобках, а если неверно, то происходит выход из цикла;
- **for(){<переменная>; <условие>; <выражение_1>}** – код в теле цикла будет исполнятся до тех пор, пока объявленная в цикле переменная будет удовлетворять условию цикла, выражение_1 каким-либо способом меняет значение этой переменной.

Операторы:

- **if(){} … else{}** – если выполняется условия, указанное в круглых скобках, то выполняется код в фигурных скобках после if, иначе – в фигурных скобках после else (else не является обязательной частью конструкции)
- **switch(<переменная>){case x:… break; … default:…break;}** – от значения переменной в круглых скобках зависит, какой кейс будет выполняться (например, если переменная имеет значение x – выполнится case x). Если же не будет кейса с таким значением, то выполнится код из блока default.

Функции:

- **<тип_функции> имя_функции(<аргумент_1>, … , <argument_n>) {}** – при вызове данной функции в главной(main) функции выполняется код в фигурных скобках, а затем возвращает значение оператором return (если тип функции не void)

Выполнение работы.

Решение задачи заключается в считывании данных, их обработке и выводе результата.

Чтобы считать данные, были созданы следующие переменные:

1. Массив **numbers** типа int, в котором хранятся вводимые целые числа;
2. Переменная **value** типа int. В ней хранится значение (0, 1, 2 или 3), от которого зависит способ обработка целых чисел массива;
3. Переменная **index** типа int, которая считает, сколько чисел ввели в массив.

Также для работы с созданными функциями были сделаны следующие переменные:

1. Переменная **max_n** типа int для хранения максимального по модулю числа(элемента массива);

2. Переменная **min_n** типа int для хранения минимального по модулю числа(элемента массива);
3. Переменная **ind_max** типа int для хранения индекса максимального по модулю числа;
4. Переменная **summa** типа int для хранения суммы чисел, расположенных после максимального по модулю числа(элемент массива), включая его;
5. Переменная **different** типа int для хранения разницы между максимального по модулю числа(элемент массива) и минимального по модулю числа(элемент массива).

Для реализации программы были созданы следующие функции:

1. Функция **void abs_max(int *numbers, int index, int *max_n, int *ind_max)**. Она принимает целочисленный массив numbers, целочисленную переменную index, целочисленные переменные max_ n и ind_max.

   Цель этой функции – найти максимальный по модулю элемент массива и его индекс. Для этого переменной max_n присваивается первый элемент массива numbers[0]. Далее в цикле for все элементы массива с индексами от 0 до значения переменной index сравниваются оператором if на удовлетворение условию abs(max_n) < abs(numbers[k]) (k – счетчик цикла). Если условие выполняется, то переменная max_n принимает значение элемента numbers[k], а также переменная ind_max принимает значение k.

   В результате мы получаем максимальный по модулю элемент массива и его индекс. Функция передает эти значения в переменные max_n и ind_max, которые указаны в аргументах функции.
2. Функция **void abs_min(int *numbers, int index, int *min_n)**. Она принимает целочисленный массив numbers, целочисленную переменную index, целочисленные переменные min_ n.

Цель этой функции – найти минимальный по модулю элемент массива. Для этого переменной min_n присваивается первый элемент массива numbers[0]. Далее в цикле for все элементы массива с индексами от 0 до значения переменной index сравниваются оператором if на удовлетворение условию abs(min_n) > abs(numbers[k]) (k – счетчик цикла). Если условие выполняется, то переменная min_n принимает значение элемента numbers[k].

В результате мы получаем минимальный по модулю элемент массива. Функция передает это значение в переменную min_n, которая указана в аргументах функции.

3. Функция **int diff(int *max_n, int *min_n)**. Она принимает целочисленные переменные max_n и min_n.

   Цель этой функции – найти разницу между максимальным по модулю элементом массива и минимальным по модулю элементом массива. Для этого обозначается переменная different, равная разнице max_n и min_n.

   В результате функция возвращает значение переменной different с помощью оператора return.

4. Функция **void sum(int *numbers, int index, int *ind_max, int *summa)**. Она принимает целочисленный массив numbers, целочисленную переменную index, целочисленные переменные ind_max и summa.

   Цель этой функции – найти сумму всех элементов массива, расположенных после максимального по модулю элемента массива, включая его. Для этого с помощью цикла for переменная summa увеличивается на величину, равную значениям элементов массива numbers[j] (j - счетчик) с индексами от 0 до index.

   В результате функция выдает сумму всех элементов массива, расположенных после максимального по модулю элемента массива, включая его. Функция передает это значение в переменную summa, которая указана в аргументах функции.

Описание библиотек:

1. **stdio.h** – используется для подключения ввода-вывода (printf(), scanf());
2. **stdlib.h –** используются для доступа к функции abs(), которая позволяет получить модуль числа.

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 -2 3 -6 50 -250\n	-250	Программа работает правильно
2	1 100 -101 50 -5 3\n	3	Программа работает правильно
3	3 250 100 1000 50 349\n	1399	Программа работает правильно
4	2 500 2 3 4 -1\n	501	Программа работает правильно
5	3 -5 20 100\n	100	Программа работает правильно

Выводы.

Были изучены основные управляющие конструкции языка Си. А также разработана программа, выполняющая считывание с клавиатуры исходных данных. Сначала программа считывает значение переменной value. Далее с помощью цикла while считываются целые числа, которые заполняют массив numbers, а также считается количество этих чисел в переменной index, до тех пор, пока не будет получен символ перевода строки.

Далее, с помощью оператора switch(){}, определяется дальнейшая обработка чисел, введенных в массив. Это зависит от значения, присвоенного переменной value:

Если value = 0, то вызывается функция abs_max, которая определяет значение переменной max_n. Затем это значение выводится.

Если value = 1, то вызывается функция abs_min, которая определяет значение переменной min_n. Затем это значение выводится.

Если value = 2, то вызывается функция diff, которая определяет значения переменной different. Затем это значение выводится.

Если value = 3, то вызывается функция sum, которая определяет значение переменной summa. Затем это значение выводится.

Если же value имеет какое-либо другое значение, программа выведет фразу “Данные некорректны”.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>

void abs_max(int *numbers, int index, int *max_n, int *ind_max){
  *max_n = numbers[0];
  for(int k=0; k<index;k++){
    if(abs(*max_n) < abs(numbers[k])){
      *max_n = numbers[k];
      *ind_max = k;
    }
  }
}

void abs_min(int *numbers, int index, int *min_n){
  *min_n = numbers[0];
  for(int p=0; p<index; p++){
    if(abs(*min_n) > abs(numbers[p])){
      *min_n = numbers[p];
    }
  }
}

int diff(int *max_n, int *min_n){
  int different = *max_n - *min_n;
  return different;
}

void sum(int *numbers, int index, int *ind_max, int *summa){
  for(int j = *ind_max; j < index; j++){
    *summa = *summa + numbers[j];
  }
}

// ----------------------------------------

int main()
{
    int numbers[100];
    int index = 0;
    int summa = 0;
    int value, max_n, ind_max, min_n, different;

    scanf("%d", &value);

    while(getchar()!='\n'){
        scanf("%d", &numbers[index]);
        index++;
    }

    switch(value){
        case 0:
          abs_max(numbers, index, &max_n, &ind_max);
```

```
            printf("%d\n", max_n);
            break;
        case 1:
            abs_min(numbers, index, &min_n);
            printf("%d\n", min_n);
            break;
        case 2:
            abs_max(numbers, index, &max_n, &ind_max);
            abs_min(numbers, index, &min_n);
            different = diff(&max_n, &min_n);
            printf("%d\n", different);
            break;
        case 3:
            abs_max(numbers, index, &max_n, &ind_max);
            sum(numbers, index, &ind_max, &summa);
            printf("%d\n", summa);
            break;
        default:
            printf("Данные некорректны\n");
            break;
    }

    return 0;
}
```