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. 3384 _____ Рудаков А.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2023

Цель работы.

Изучить основные управляющие конструкции языка C: условия, циклы, оператор switch. Получить практические навыки путем написания программного кода.

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 число в массиве. (max)

1 : минимальное число в массиве. (min)

2 : разницу между максимальным и минимальным элементом. (diff)

3 : сумму элементов массива, расположенных до первого минимального элемента. (sum)

иначе необходимо вывести строку "Данные некорректны".

Ошибкой в данном задании считается дублирование кода!
При выводе результата, не забудьте символ переноса строки.

Выполнение работы.

В начале программы была подключена библиотека *stdio.h*, подключающая функции стандартного ввода и вывода.

Далее была написана основная функция *int main()*. Внутри нее определены 3 переменные:

- *int array[102]* — массив, содержащий 102 элемента типа *int*, в который будут записаны считанные значения.

- *int n* — переменная типа int, в которую будет записано первое считанное число
- *int len = 0* — переменная типа *int*, в которой будет храниться количество считанных значений.

Функция ввода *scanf("%d",&n)* считывает первое поданное на вход число типа *int*, тип определяется благодаря спецификации *%d*, и передает значение переменной *n*, передача происходит благодаря *&*.

Далее вызывается функция *scanf_array(array).*

Следующим действием выполняется цикл *for (int i=0;i<101;i++)*, в котором определяется внутренняя переменная *int i=0* типа *int*, имеющая начальное значение 0, условие продолжения цикла *i<101* (цикл будет выполняться до момента, пока условие верно) и действие, которое будет происходить после каждого прохода цикла *i++* - увеличение переменной *i* на 1. Данный цикл нужен для того, чтобы определить количество введенных значений. Внутри цикла находиться условный оператор *if (array[i]==101)*, который проверяет *i*-ый элемент массива *array* на его равенства числу 101. Так как в массиве *array* существует единственный элемент равный 101, который определяется в функции *scan_array*, идущий после всех введенных элементов, его индекс будет обозначать количество поданных на вход чисел. Если *i*-ый элемент равен 101, то выполняются действия внутри условного оператора, а именно присваивание переменной *len* значения, равного *i*, и выход из цикла, реализованный благодаря *break*.

Далее находиться условный оператор *if (array[101]!=102)*, который проверяет неравенство 101 элемента массива *array* значению 102. Если элемент равен данному значению, значит данные некорректны. 102-ой элемент определяется в функции *scan_array*, и становиться равным только в том случае, если введенные данные некорректны. В таком случае выполняется ветка *else*, в которой при помощи функции вывода *printf("Данные некоректны\n")* на экран выводиться сообщение о некорректности данных, *\n* в конце выводимого сообщения обозначает символ переноса строки.

Если же 101-ый элемент массива *array* не равен 102, то выполняется действие внутри условного оператора. Внутри оператора *if* находиться оператор *switch(n)*, зависящий от переменной *n*.

- Если *n=0*, то выполняются действия внутри *case 0*. Вызывается функция *def_max(array,len)*, после чего происходит выход из условного оператора, реализованный через *break*.
- Если *n=1*, то выполняются действия внутри *case 1*. Вызывается функция *def_min(array,len)*, после чего происходит выход из условного оператора, реализованный через *break*.
- Если *n=2*, то выполняются действия внутри *case 2*. Вызывается функция *def_diff(array,len)*, после чего происходит выход из условного оператора, реализованный через *break*.
- Если *n=3*, то выполняются действия внутри *case 3*. Вызывается функция *def_sum(array,len)*, после чего происходит выход из условного оператора, реализованный через *break*.
- Если *n* не равно ни одному из предыдущих значений, то выполняются действия внутри *default*. Функция вывода *printf("Данные некорректны\n")* выводит на экран сообщение о некорректности данных.

Далее была реализована функция *void scan_array(int arr[])*, предназначенная для ввода массива элементов. Она реализована при помощи типа функции *void*, невозвращающей значения. В ее аргументах содержится *int arr[]* - ссылка на массив *int array[]* в функции *int main()*.

В данной функции определяются 3 переменные:

- *char symbol* — переменная типа *char*, которая в последующем будет хранить символ, стоящий в строке после очередного числа из введенного массива.
- *int i = 0* — переменная типа *int*, отвечающая за индекс считываемого элемента. Имеет начальное значение, равное 0.
- *int proverka = 0* — переменная типа *int*, благодаря которой определяется корректность введенных данных. Имеет начальное значение, равное 0.

Далее идет цикл *do { } while (symbol!='\n' && proverka==0)* с постусловием, который повторяется, пока верны оба условия. Первое условие *symbol!='\n'* проверяет наличие символа переноса строки после считанного значения. Второе условие *proverka==0* проверяет введенные данные на корректность. Пока значение равно 0, данные верны. Внутри цикла первым стоит условный оператор *if (scanf("%d%c",&arr[i],&symbol)==0),* внутри которого находиться функция ввода *scanf*, которая отвечает за считывание числа типа *int* и последующего символа. Спецификация *%d* — указывает на считывание числа типа *int* и передает значение в переменную *array[i]* через *arr[i] (*ссылку на него внутри функции), которая принимает значение благодаря *&*. Спецификация *%c* — указывает на считывание символа типа *char* и передает значение в переменную *symbol*, которая принимает значение, благодаря *&*. В условном операторе проверяется корректность введенных значений, если введенные данные не корректны, то выполняются действия внутри *if*. Первым идет присваивание переменной *proverka* значения 1, вторым действием элементу массива *array* с индексом *i* через *arr[i] (*ссылку на него внутри функции) присваивается значение 102. Далее в цикле идет второй условный оператор *if*, имеющий условие *(arr[i]>100)*, который проверяет корректность введенных данных, а конкретно их непревосходство 100. Если введенное число больше 100, то данные некорректны, значит выполняются действия внутри условного оператора, идентичные действиям, внутри предыдущего условного оператора.

Последним действием внутри функции выполняется присваивание переменной *array[i]* значения 101 через *arr[i] (*ссылку на него внутри функции).

Следующей реализованной функцией стала *int max(int first, int second),* которая получает 2 числа типа *int*, записанные в переменные *first* и *second*. Данная функция предназначена для нахождения наибольшего значения среди двух чисел. Функция *max* возвращает значение типа *int*. В теле функции содержится условный оператор *if (first>second)*, выполняющий проверку условия, что значение переменной *first* больше значения переменной *second*. Если условие верно, функция возвращает значение переменной *first*, благодаря оператору

return. Если условие ложно, функция возвращает значение переменной *second*, благодаря оператору *return*.

Далее была реализована функция *int min(int first, int second)*, которая получает 2 числа типа *int*, записанные в переменные *first* и *second*. Данная функция предназначена для нахождения наименьшего значения среди двух чисел. Функция *min* возвращает значение типа *int*. В теле функции содержится условный оператор *if (first<second)*, выполняющий проверку условия, что значение переменной *first* меньше значения переменной *second*. Если условие верно, функция возвращает значение переменной *first*, благодаря оператору *return*. Если условие ложно, функция возвращает значение переменной *second*, благодаря оператору *return*.

Следующей созданной функцией стала *void def_max(int arr[],int len)*, предназначенная для нахождения и вывода на экран максимального значения внутри массива. Она реализована благодаря типу функции *void*, невозвращающей значения. В аргументах она получает *int arr[]* - ссылка на массив *array[]*, и переменную *len* типа *int*, содержащую значение количества введенных чисел. Внутри функции определена переменная *int maximum=arr[0]* типа *int*, имеющая начальное значение, равное *0*-му элементу массива *array* через *arr* (ссылку на него внутри функции), в которой будет храниться текущее значение наибольшего элемента массива. Далее идет цикл *for (int i=0;i<len;i++)*, в котором определена внутренняя переменная *int i=0*, условие выполнения цикла *i<len* и действие, выполняемое после каждого прохода цикла *i++*. Внутри цикла вызывается функция *max(maximum,arr[i])*, вернувшееся значение присваивается переменной *maximum*. После выполнения цикла выполняется вывод на экран, организованный благодаря функции вывода *printf("%d\n",maximum)*, которая выводит значение переменной *maximum* целого типа, указанного благодаря спецификации *%d*.

Далее была написана функция *void def_min(int arr[],int len)*, предназначенная для нахождения и вывода на экран минимального значения внутри массива. Она реализована благодаря типу функции *void*,

невозвращающей значения. В аргументах она получает *int arr[]* - ссылка на массив *array[]*, и переменную *len* типа *int*, содержащую значение количества введенных чисел. Внутри функции определена переменная *int minimum=arr[0]* типа *int*, имеющая начальное значение, равное *0*-му элементу массива *array* через *arr* (ссылку на него внутри функции), в которой будет храниться текущее значение наименьшего элемента массива. Далее идет цикл *for (int i=0;i<len;i++)*, в котором определена внутренняя переменная *int i=0*, условие выполнения цикла *i<len* и действие, выполняемое после каждого прохода цикла *i++*. Внутри цикла вызывается функция *min(minimum,arr[i])*, вернувшееся значение присваивается переменной *minimum*. После выполнения цикла выполняется вывод на экран, организованный благодаря функции вывода *printf("%d\n",minimum)*, которая выводит значение переменной *minimum* целого типа, указанного благодаря спецификации *%d*.

Следующей была реализована функция *void def_diff(int arr[],int len)*, созданная для нахождения разницы между максимальным и минимальным элементом массива. Она реализована благодаря типу функции *void*, невозвращающей значения. В аргументах она получает *int arr[]* - ссылка на массив *array[]*, и переменную *len* типа *int*, содержащую значение количества введенных чисел. Внутри функции определена переменная *int maximum=arr[0]* типа *int*, имеющая начальное значение, равное *0*-му элементу массива *array* через *arr* (ссылку на него внутри функции), в которой будет храниться текущее значение наибольшего элемента массива, а так же переменная *int minimum=arr[0]* типа *int*, имеющая начальное значение, равное *0*-му элементу массива *array* через *arr* (ссылку на него внутри функции), в которой будет храниться текущее значение наименьшего элемента массива. Далее идет цикл *for (int i=0;i<len;i++)*, в котором определена внутренняя переменная *int i=0*, условие выполнения цикла *i<len* и действие, выполняемое после каждого прохода цикла *i++*. Внутри цикла вызывается функция *max(maximum,arr[i])*, вернувшееся значение присваивается переменной *maximum*. Далее вызывается функция *min(minimum,arr[i])*, вернувшееся значение присваивается переменной *minimum*. После выполнения

цикла определена переменная *int diff=maximum-minimum* типа *int*, имеющая начальное значение, равное разнице значений переменных *maximum* и *minimum*. После чего выполняется вывод на экран, организованный благодаря функции вывода *printf("%d\n",diff)*, которая выводит значение переменной *diff* целого типа, указанного благодаря спецификации *%d*.

Далее была создана функция *void def_sum(int arr[],int len)*, предназначенная для нахождения суммы элементов до первого минимального элемента массива. Она реализована благодаря типу функции *void*, невозвращающей значения. В аргументах она получает *int arr[]* - ссылка на массив *array[]*, и переменную *len* типа *int*, содержащую значение количества введенных чисел. Внутри функции определена переменная *int minimum=arr[len]* типа *int*, имеющая начальное значение, равное *len*-ому элементу массива *array* через *arr* (ссылку на него внутри функции), в которой будет храниться текущее значение наименьшего элемента массива. Также определена переменная *int minimum_old=minimum* типа *int*, имеющая начальное значение равное значению переменной *minimum*, в которой будет храниться предыдущее значение наименьшего элемента массива, и переменная *int sum=0* типа *int*, имеющая начальное значение *0*, в которой будет храниться текущее значение суммы. Далее идет цикл *for (int i=len;i>-1;i--)*, в котором определена внутренняя переменная *int i=len*, условие выполнения цикла *i>-1* и действие, выполняемое после каждого прохода цикла *i*--. Внутри цикла вызывается функция *min(minimum,arr[i])*, вернувшееся значение присваивается переменной *minimum*. Далее содержится условный оператор *if (minimum==minimum_old && arr[i]!=minimum)*, внутри которого содержатся 2 условия. Первое условие проверяет равенство нового и старого значения наименьшего элемента массива, второе условие проверяет неравенство данного элемента массива и его текущего наименьшего элемента. Если оба условия верны, значение переменной *sum* увеличивается на значение переменной *arr[i]*. В противном случае переменной *sum* присваивается значение 0, переменной *minimum_old* присваивается значение переменной *minimum*. После выполнения цикла выполняется вывод на экран, организованный благодаря

функции вывода *printf("%d\n",sum)*, которая выводит значение переменной *sum* целого типа, указанного благодаря спецификации *%d*.

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 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	93	Верно
2.	1 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	0	Верно
3.	2 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	93	Верно
4	3 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	504	Верно
5	skjfsakj	Данные некорректны	Верно

Выводы.

Были изучены основные управляющие конструкции языка С: условия, циклы, оператор switch.

Был разработан программный код, благодаря которому выполняется считывание массива целых чисел и вывод на экран результата в зависимости от

первого введенного значения. Для обработки данного значения использовались операторы sitch,case. В случае некорректных данных программа выводит "Данные некорректны"

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb1.c

```
#include <stdio.h>

void scan_array(int arr[]){
      char symbol;
      int i=0;
      int proverka=0;
      do{
            if (scanf("%d%c",&arr[i],&symbol)==0){
                  proverka=1;
                  arr[101]=102;
            }
            if (arr[i]>100){
                  proverka=1;
                  arr[101]=102;
            }
            i++;
      }while(symbol!='\n' && proverka==0);
      arr[i]=101;
}

int max(int first, int second){
      if (first>second){
            return first;
      } else{
            return second;
      }
}

int min(int first, int second){
      if (first<second){
            return first;
      } else{
            return second;
      }
}

void def_max(int arr[],int len){
      int maximum=arr[0];
         for (int i=0;i<len;i++){
                  maximum=max(maximum,arr[i]);
      }
         printf("%d\n",maximum);
}

void def_min(int arr[],int len){
      int minimum=arr[0];
         for (int i=0;i<len;i++){
            minimum=min(minimum,arr[i]);
         }
         printf("%d\n",minimum);
}
```

```
void def_diff(int arr[],int len){
	int minimum=arr[0];
	int maximum=arr[0];
	for (int i=0;i<len;i++){
		maximum=max(maximum,arr[i]);
		minimum=min(minimum,arr[i]);
	}
	int diff=maximum-minimum;
	printf("%d\n",diff);
}

void def_sum(int arr[],int len){
	int minimum=arr[len];
	int minimum_old=minimum;
	int sum=0;
	for (int i=len;i>-1;i--){
		minimum=min(minimum,arr[i]);
		if (minimum==minimum_old && arr[i]!=minimum){
			sum+=arr[i];
		}else{
			sum=0;
			minimum_old=minimum;
		}
	}
	printf("%d\n",sum);
}

int main(){
	int array[102];
	int n;
	scanf("%d",&n);
	scan_array(array);
	int len=0;
	for (int i=0;i<101;i++){
		if (array[i]==101){
			len=i;
			break;
		}
	}
	if (array[101]!=102){
		switch(n){
			case 0:
				def_max(array,len);
				break;
			case 1:
				def_min(array,len);
				break;
			case 2:
				def_diff(array,len);
				break;
			case 3:
				def_sum(array,len);
				break;
			default:
				printf("Данные некорректны\n");
		}
	}else{
		printf("Данные некоректны\n");
```

```
        }
    }
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Если результаты тестирования велики (больше 1 страницы), то их выносят в приложение.

Процесс тестирования можно представить в виде таблицы, например:

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
4.			
5.			
6.			
...			

Обратите внимание, что в нумерации таблицы в приложении обязательно должен быть в качестве префикса номер самого приложения: А.