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

ОТЧЕТ

по лабораторной работе № 2

по дисциплине «Программирование»

Тема: Обработка одномерных массивов

Студент(ка) гр. 3311 _____ Баймухамедов Р.Р

Преподаватель _____ Хахаев И.А.

Санкт-Петербург

2023

Цель работы.

Целью работы является изучение и освоение основных методов обработки одномерных массивов на языке программирования С.

Задание (вариант 9)

Дан массив, упорядоченный по возрастанию, и число а, о котором известно следующее: оно не равно ни одному из элементов массива, больше первого и меньше последнего элемента.

а) Вывести все элементы массива, меньшие а.
б) Найти два элемента массива (их порядковые номера и значения), в интервале, между которыми находится значения а.
в) Найти элемент массива (его порядковый номер и значение), ближайшее к а.

Условный оператор (if) не использовать.

Постановка задачи и описание решения

Для выполнения данной лабораторной работы необходимо разработать программу, которая будет выводить все элементы массивы, меньшие а, находить два ближайшие к а элемента массива между которыми заключено значение а и находить элемент массива, наиболее ближайший к а.

Для выполнения поставленных задач необходимо выполнить следующие шаги:

1. Поочередно запросить у пользователя значения элементов массива, упорядоченные по возрастанию, и сохранить эти значения b[i] в массиве с соответствующими порядковыми номерами.
2. Запросить у пользователя значение а, не равное ни одному из элементов массива, а также больше первого и меньше последнего элемента.
3. Для вывода всех элементов массива, меньших а, следует поочередно проверять условие (значение элемента массива меньше а), начиная с

нулевого порядкового номера и увеличивая его на один, и выводить значение элемента массива при успешной проверке условия, до тех пор, пока проверка не окажется проваленной. Так как массив упорядочен по возрастанию, значит, что все значения элементов массива, идущие после элемента, провалившего проверку, также провалят её.

4. Для нахождения элементов массива, в интервале между которыми находится значение a, следует сначала найти меньшее значение интервала, затем большее.
5. Для нахождения меньшего значения интервала необходимо постепенно отнимать значение, начиная с 1, у значения a и проверять каждый элемент на равенство его значения с этой разностью. При успешной проверке сохраняем значение элемента массива (lw) и его порядковый номер (lwi) и прерываем поиск меньшего значения интервала. Если ни одно значение элемента не оказалось равно полученной разности, то значение, отнимаемое от a, увеличиваем на единицу и начинаем проверку каждого элемента на равенство полученной разности заново.
6. Для нахождения большего значения интервала необходимо постепенно прибавлять значение, начиная с 1, к значению a и проверять каждый элемент на равенство его значения с этой суммой. При успешной проверке сохраняем значение элемента массива (up) и его порядковый номер (upi) и прерываем поиск большего значения интервала. Если ни одно значение элемента не оказалось равно полученной сумме, то значение, прибавляемое к a, увеличиваем на единицу и начинаем проверку каждого элемента на равенство полученной сумме заново.
7. Вывести порядковые номера и значения элементов массива, в интервале, между которыми находится значение a.
8. Среди значений элементов массива, меньших a, ближайшим будет lw с порядковым номером lwi, вычисленное в пункте (б). Среди значений элементов массива, больших a, ближайшим будет up с порядковым номером upi, вычисленное в пункте (б). Значит, чтобы найти элемент,

значение которого наиболее близкое к a, необходимо сравнить значение a лишь с lw и up. Чтобы вычислить значение, наиболее близкое к a, необходимо сравнить модули разности a и значения элемента массива. Наиболее близким к a значением будет являться то, чья по модулю разность будет наименьшей. Если модуль разности a и lw больше модуля разности a и up, то значение up является наиболее близким, сохраняем значение up в df и upi в dfi. Иначе сохраняем значение lw в df и lwi в dfi.

9. Вывести порядковый номер и значение элемента массива, наиболее близкое к a.

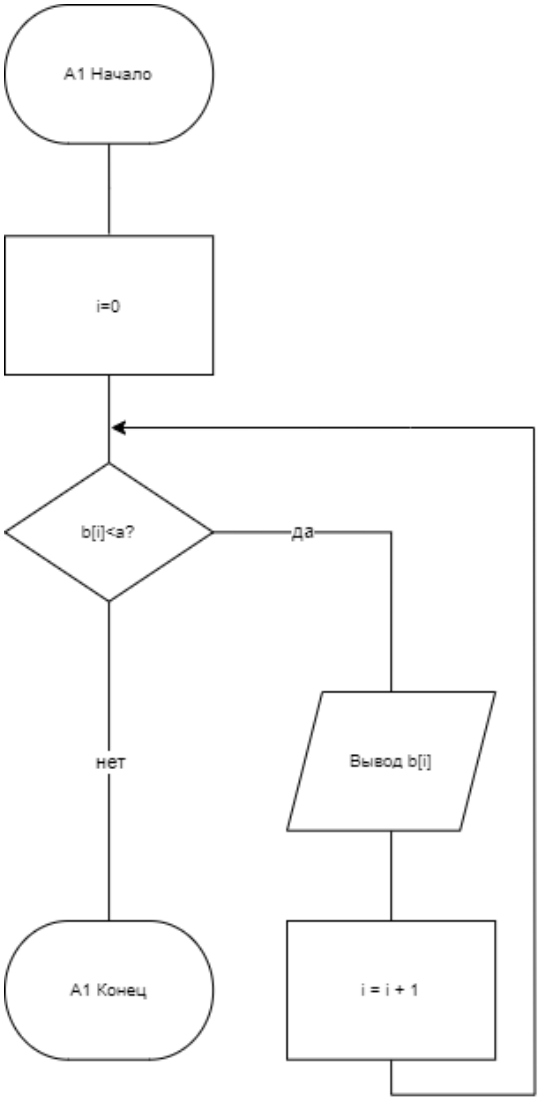
Описание переменных

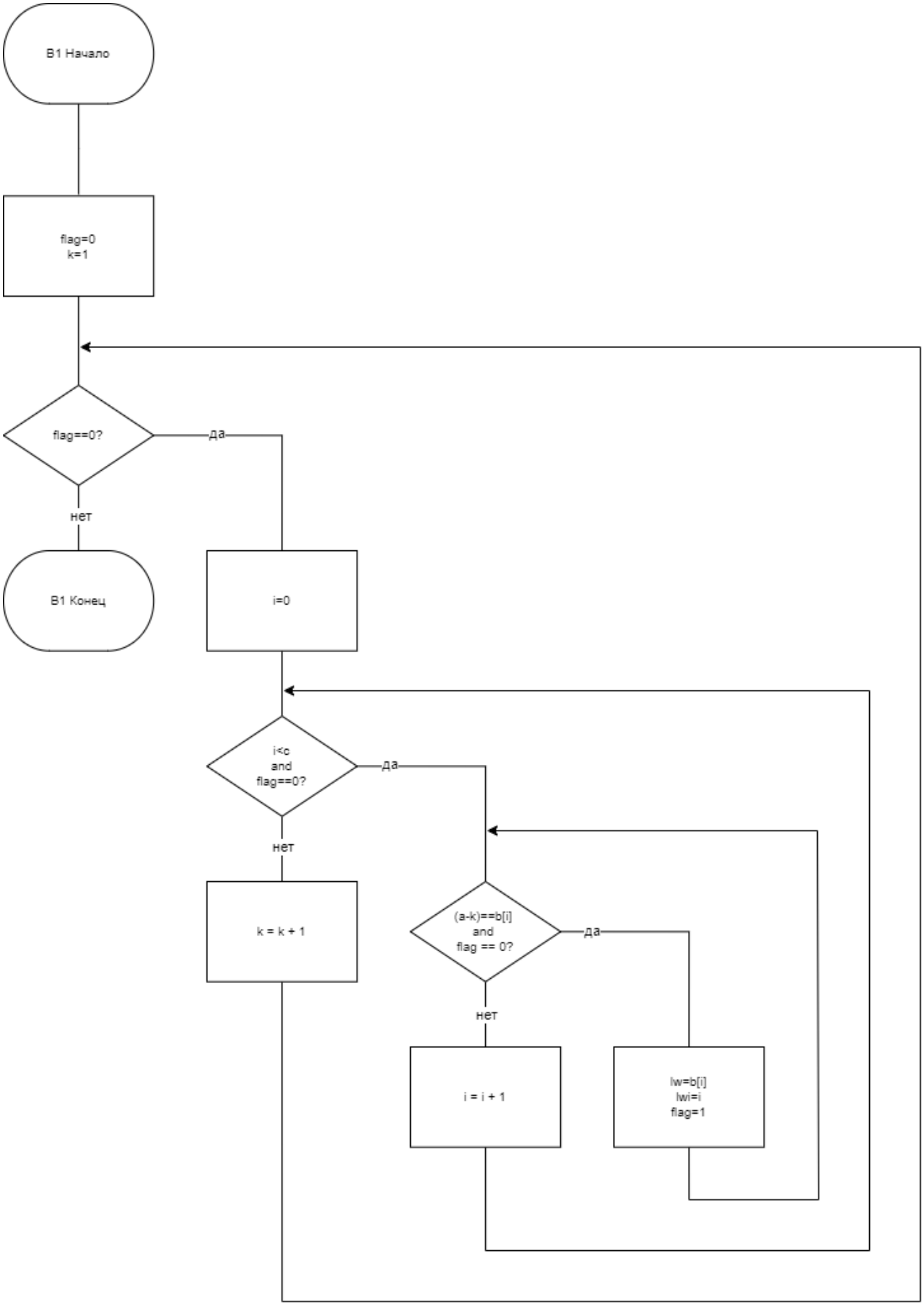
№	Имя переменной	Тип	Назначение
1	a	int	Исходное значение, относительно которого происходят вычисления программы.
2	n	int	Вводимое значение элемента массива.
3	i	int	Переменная в цикле.
4	c	int	Количество элементов в массиве, определенное в #define.
5	k	int	Переменная в цикле, которая вычитается/складывается из значения элемента массива для определения нижнего/верхнего порога интервала.
6	b[c]	int	Массив, количество элементов которого равняется c.
7	flag	int	Вспомогательная переменная, изменяющая своё значение при успешном выполнении определённого условия.

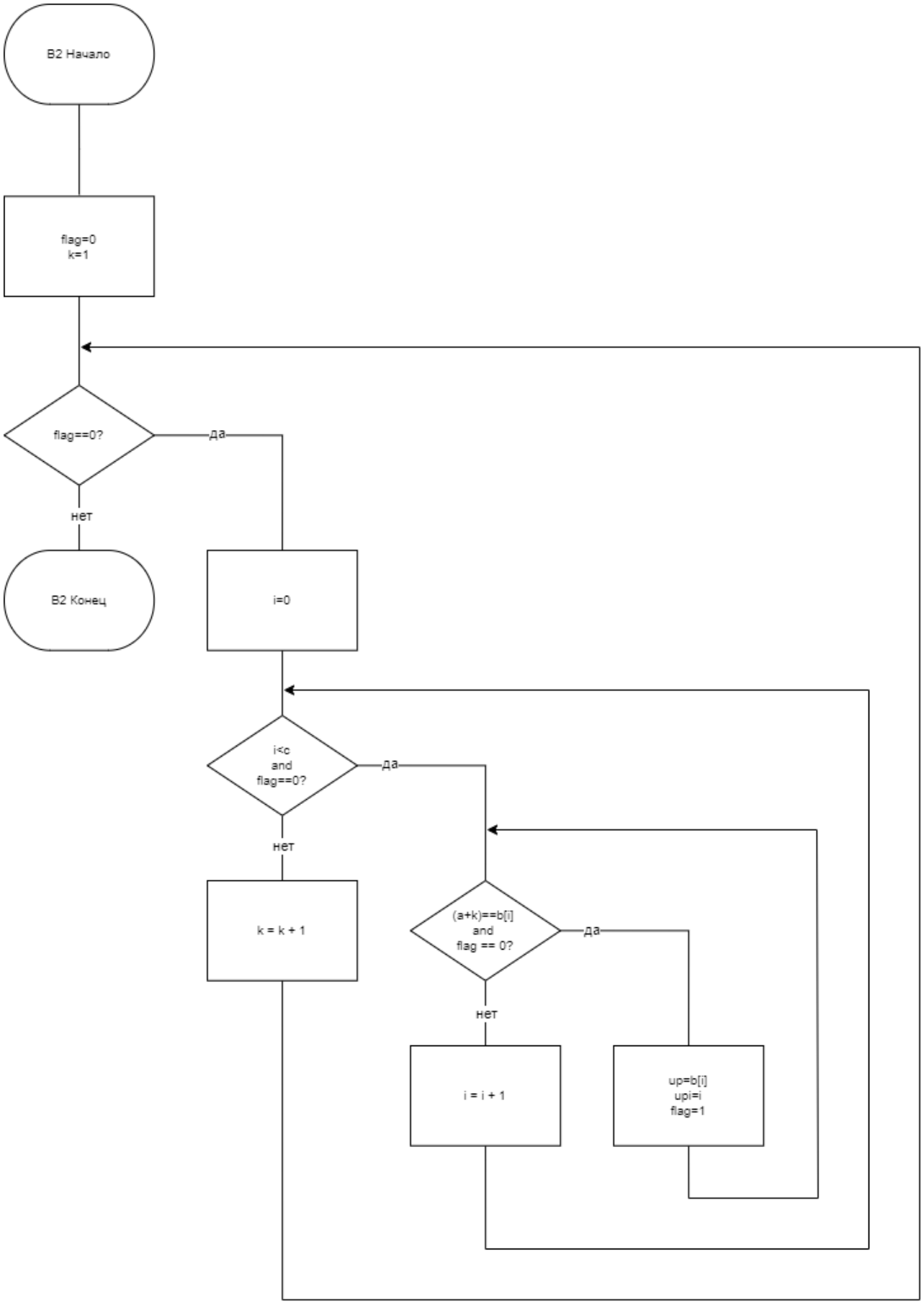
8	df	int	Значение элемента массива ближайшего к a.
9	dfi	int	Порядковый номер элемента массива, значение от которого наиболее близкое к a.
10	lw	int	Значение элемента массива, которое является нижним порогом интервала, между которым находится значение a.
11	lwi	int	Порядковый номер элемента массива, значение от которого является нижним порогом интервала, между которым находится значение a.
12	up	int	Значение элемента массива, которое является верхним порогом интервала, между которым находится значение a.
13	upi	int	Порядковый номер элемента массива, значение от которого является верхним порогом интервала, между которым находится значение a.

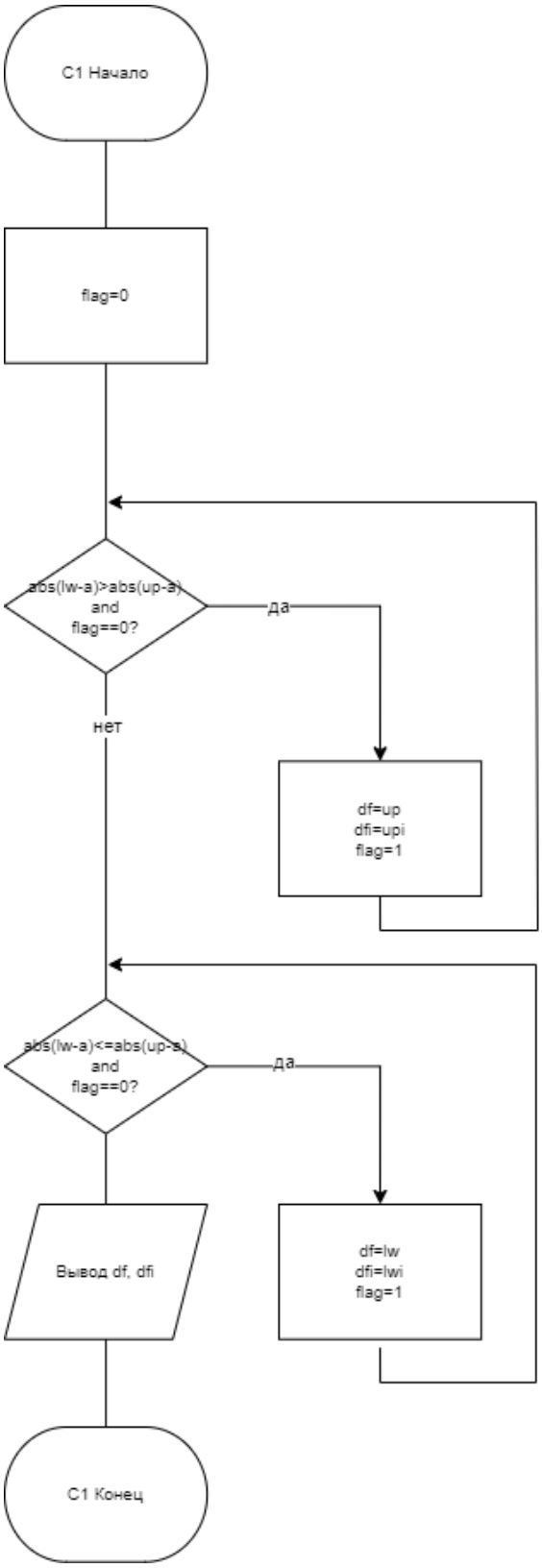
Схема алгоритма


Начало
Ввод значений элементов массива
Ввод a
A1
Выполнение пункта А
B1
Выполнение пункта Б. Нахождение меньшего значения интервала
B2
Выполнение пункта Б. Нахождение большего значения интервала
Вывод меньшего (lw, lwi) и большего (up, upi) элемента интервала
C1
Выполнение пункта В
Конец


A1 Начало
i=0
b[i]<a?
да
нет
Вывод b[i]
A1 Конец
i = i + 1


B1 Начало
flag=0
k=1
flag==0?
Да
нет
B1 Конец
i=0
i<c
and
flag==0?
Да
нет
k = k + 1
(a-k)==b[i]
and
flag == 0?
Да
нет
i = i + 1
lw=b[i]
lwi=i
flag=1


B2 Начало
flag=0
k=1
flag==0?
да
нет
B2 Конец
i=0
i<c
and
flag==0?
да
нет
k = k + 1
(a+k)==b[i]
and
flag == 0?
да
нет
i = i + 1
up=b[i]
upi=i
flag=1


C1 Начало
flag=0
abs(lw-a)>abs(up-a)
and
flag==0?
да
нет
df=up
dfi=upi
flag=1
abs(lw-a)<=abs(up-a)
and
flag==0?
да
Вывод df, dfi
df=lw
dfi=lwi
flag=1
C1 Конец

Контрольные примеры

Пример 1:

Исходные данные: c=10, b[0]=12, b[1]=17, b[2]=24, b[3]=39, b[4]=46, b[5]=54, b[6]=70, b[7]=87, b[8]=92, b[9]=102, a=30

Результаты:

a) 12;17;24;
b) Array(2)=24 < 30 < Array(3)=39
c) Array(2)=24

Пример 2:

Исходные данные: c=10, b[0]=-183, b[1]=-98, b[2]=-54, b[3]=-13, b[4]=4, b[5]=26, b[6]=55, b[7]=78, b[8]=98, b[9]=139, a=0

Результаты:

a) -183;-98;-54;-13;
b) Array(3)=-13 < 0 < Array(4)=4
c) Array(4)=4

Пример 3:

Исходные данные: c=10, b[0]=-1337, b[1]=-228, b[2]=-102, b[3]=12, b[4]=13, b[5]=15, b[6]=1337, b[7]=1488, b[8]=2025, b[9]=3000, a=14

Результаты:

d) -1337;-228;-102;12;13;
e) Array(4)=13 < 14 < Array(5)=15
f) Array(4)=13

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#define c 10
```

```
int main(){
   int i,a,n,up,upi,lw,lwi,k,b[c],flag,df,dfi;

   printf("Enter array sorted by increase.\n");
   for(i=0; i<c;i++){
      printf("Enter element %i:", i);
      scanf("%i", &n);
      b[i]=n;
   }

   printf("\nEnter value of 'a':");
   scanf("%i", &a);

   printf("\na) ");
   i=0;
   while(b[i]<a){
      printf("%i;", b[i]);
      i++;
   }

   flag=0;
   for(k=1;flag==0;k++){
      for(i=0;i<c && flag==0;i++){
         while((a-k)==b[i] && flag==0){
            lw=b[i];
            lwi=i;
            flag=1;
         }
      }
   }
```

```
    flag=0;
    for(k=1;flag==0;k++){
        for(i=0;i<c && flag==0;i++){
            while((a+k)==b[i] && flag==0){
                up=b[i];
                upi=i;
                flag=1;
            }
        }
    }
    printf("\nb) Array(%i)=%i < %i < Array(%i)=%i",lwi,lw,a,upi,up);

    flag=0;
    while(abs(lw-a)>abs(up-a) && flag==0){
        df=up;
        dfi=upi;
        flag=1;
    }

    flag=0;
    while(abs(lw-a)<=abs(up-a) && flag==0){
        df=lw;
        dfi=lwi;
        flag=1;
    }

    printf("\nc) Array(%i)=%i\n", dfi,df);
    return 0;
}
```

Примеры выполнения программы

```
Enter array sorted by increase.
Enter element 0:12
Enter element 1:17
Enter element 2:24
Enter element 3:39
Enter element 4:46
Enter element 5:54
Enter element 6:70
Enter element 7:87
Enter element 8:92
Enter element 9:102

Enter value of 'a':30

a) 12;17;24;
b) Array(2)=24 < 30 < Array(3)=39
c) Array(2)=24
```

```
Enter array sorted by increase.
Enter element 0:-183
Enter element 1:-98
Enter element 2:-54
Enter element 3:-13
Enter element 4:4
Enter element 5:26
Enter element 6:55
Enter element 7:78
Enter element 8:98
Enter element 9:139

Enter value of 'a':0

a) -183;-98;-54;-13;
b) Array(3)=-13 < 0 < Array(4)=4
c) Array(4)=4
```

```
Enter array sorted by increase.
Enter element 0:-1337
Enter element 1:-228
Enter element 2:-102
Enter element 3:12
Enter element 4:13
Enter element 5:15
Enter element 6:1337
Enter element 7:1488
Enter element 8:2025
Enter element 9:3000

Enter value of 'a':14

a) -1337;-228;-102;12;13;
b) Array(4)=13 < 14 < Array(5)=15
c) Array(4)=13
```

Выводы.

В результате выполнения работы изучены и освоены основные методы обработки одномерных массивов на языке программирования С.