

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего
образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО
ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент

Л.Н. Бариков

Отчет

по лабораторной работе № 8

по дисциплине Информатика

на тему: «Суммирование рядов»

Работу выполнил

студент гр. 4342

С.М.Иванов

Санкт-Петербург

2024

16.

Дан числовой массив $b_0, b_1, \dots, b_{2n-1}$. Определить сумму значений элементов массива с нечётными номерами, лежащих между последним элементом с наибольшим среди неположительных элементов значением и последним элементом с положительным значением, имеющим номер меньше, чем n .

код на гитхабе для более удобного просмотра

https://github.com/crimson-catfish/SUAI_labs/blob/main/lab8.cpp

Программа считывает массив целых чисел от пользователя, находит индексы последнего неположительного числа и последнего положительного числа в первой половине массива, а затем вычисляет сумму нечетных элементов. индексы между этими двумя индексами.

разбивка кода:

Программа включает библиотеки `<iostream>` и `<limits>` и определяет константу `ARRAY_SIZE`, равную 32. Она также определяет тип `my_element` как целое число и тип `my_array` как массив `my_element` длиной `ARRAY_SIZE`.

Функция `read_array_length()` предлагает пользователю ввести половину длины массива (которая должна быть больше 0) и возвращает это значение.

Функция `read_user_array()` принимает целое число n и выходной массив и считывает $2*n$ целых чисел от пользователя в массив.

Функция `find_max_nonpositive_index()` принимает массив `b` и целое число `half_array_length` и возвращает индекс последнего неположительного числа в первой половине массива. Если неположительные числа не найдены, возвращается -1.

Функция `find_last_before_n_unsigned_index()` принимает массив `b` и целое число `half_array_length` и возвращает индекс последнего

положительного числа в первой половине массива. Если положительные числа не найдены, возвращается -1.

Функция `main()` предлагает пользователю ввести длину массива и считывает элементы массива. Затем он вызывает функции `find_max_nonpositive_index()` и `find_last_before_n_unsigned_index()`, чтобы найти индексы последнего неположительного числа и последнего положительного числа в первой половине массива. Он вычисляет сумму элементов по нечетным индексам между этими двумя индексами и распечатывает левый и правый связанные индексы и сумму. Затем пользователю предлагается ввести «R» для повторного запуска программы или любую другую букву для выхода.

```
Enter half of array length (greater than 0): 3
4 8 -2 53 8 0
Left bound index: 1
Right bound index: 5
Sum: 61

Enter "R" to run again, or enter any other letter to quit programm: r

Enter half of array length (greater than 0): 3
1 2 3 4 5 6
No nonpositive numbers found!

Enter half of array length (greater than 0): 3
-1 -2 -3 -4 -5 -6
No unsigned numbers found in first half of array!

Enter half of array length (greater than 0): 5
0 -1 2 -3 4 -5 6 -7 8 -9
Left bound index: 0
Right bound index: 4
Sum: -4

Enter "R" to run again, or enter any other letter to quit programm: q
```

```
#include <iostream>
```

```
#include <limits>
```

```
using namespace std;
```

```
const int ARRAY_SIZE = 32;
```

```

typedef int my_element;
typedef my_element my_array[ARRAY_SIZE];

int read_array_length()
{
    int n;
    cout << "\nEnter half of array length (greater than 0): ";
    cin >> n;
    if (n < 1)
    {
        cout << "length should be greater than 0";
    }
    return n;
}

void read_user_array(int n, my_array out)
{
    for (int i = 0; i < 2 * n; i++)
    {
        cin >> out[i];
    }
}

int find_max_nonpositive_index(my_array b, int half_array_length)
{
    int max_nonpositive;
    int max_nonpositive_index = -1;
    for (int i = 0; i < half_array_length * 2; i++)
    {
        if (b[i] <= 0)
        {
            max_nonpositive = b[i];
            max_nonpositive_index = i;
            break;
        }
        if (i == half_array_length * 2 - 1)
            return -1;
    }

    for (int i = 0; i < half_array_length * 2; i++)
    {
        if (b[i] <= 0 && b[i] > max_nonpositive)
        {

```

```
max_nonpositive = b[i];
max_nonpositive_index = i;
}
}
return max_nonpositive_index;
}
```

```
int find_last_before_n_unsigned_index(my_array b, int half_array_length)
{
for (int i = half_array_length - 1; i >= 0; i--)
{
if (b[i] > 0)
{
return i;
}
}
return -1;
}
```

```
int main()
{
char quitOrRestart = 'r';
```

```
do
{
int n = read_array_length();
my_array b;
read_user_array(n, b);
```

```
int max_nonpositive_index = find_max_nonpositive_index(b, n);
if (max_nonpositive_index == -1)
{
cout << "No nonpositive numbers found!\n";
continue;
}
```

```
int last_before_n_unsigned_index = find_last_before_n_unsigned_index(b, n);
if (last_before_n_unsigned_index == -1)
{
cout << "No unsigned numbers found in first half of array!\n";
continue;
}
```

```
int left_bound = min(max_nonpositive_index, last_before_n_unsigned_index);
int right_bound = max(max_nonpositive_index, last_before_n_unsigned_index);

cout << "Left bound index: " << left_bound << endl;
cout << "Right bound index: " << right_bound << endl;

if (left_bound % 2 == 0)
    left_bound++;

int sum = 0;
for (int i = left_bound; i < right_bound; i += 2)
{
    sum += b[i];
}

cout << "Sum: " << sum;

cout << "\n\nEnter \"R\" to run again, or enter any other letter to quit programm: ";
cin >> quitOrRestart;
} while (quitOrRestart == 'r' || quitOrRestart == 'R');

return 0;
}
```