МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

по дисциплине: Основы программирования
тема: «Стандартный ввод и вывод»

Выполнил: ст. группы ПВ-201
Машуров Дмитрий Русланович

Проверил:
Притчин Иван Сергеевич

Белгород 2021 г.

Лабораторная работа № 2

Основные понятия языка Си

Цель работы: знакомство с основными типами данных, операциями, операторами языка Си.

Задания для подготовки к работе:

1. Изучить базовые типы данных в языке Си и сравнить их с основными типами данных языка Паскаль.
2. Изучить арифметические операции и операции присваивания в Си.
3. Ознакомиться с операторами в Си.
4. Изучить стандартные математические функции библиотеки `math`.
5. Изучить правила описания функций и обращения к ним.
6. Разработать алгоритм и составить программу, состоящую, по крайней мере, из двух функций, для решения задачи соответствующего варианта. Результаты должны быть выведены в наиболее естественном виде. Например, если требуется многочлен $x^2 + 3x - 4$ разложить на множители, то результат должен быть выведен следующим образом: $x^2 + 3x - 4 = (x - 1)(x + 4)$
7. Подобрать тестовые данные

Задание варианта №17

Дана вещественная последовательность $a_1, a_2, \dots, a_n$. Определить максимальное количество идущих подряд положительных членов последовательности.
Вывести найденный фрагмент.

Выполнение:

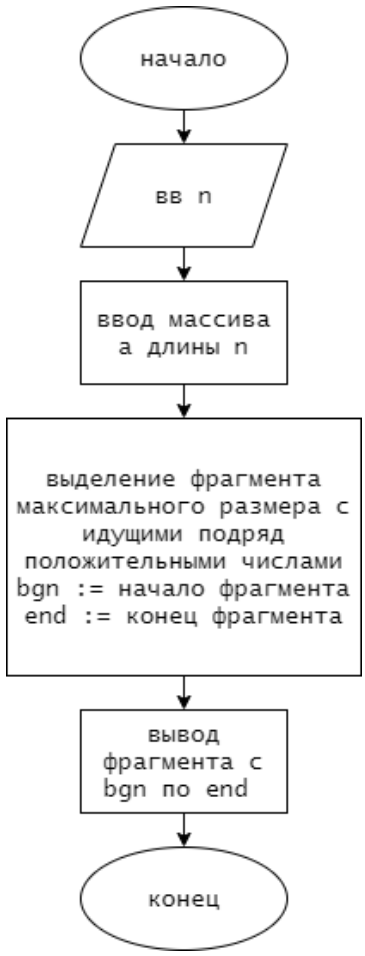
1. **Описание алгоритма и выделение подзадач**

   Исходя из того, что нам нужно определить максимальный фрагмент идущих подряд положительных чисел, можно запоминать индекс начала фрагмента и индекс конца, а затем выводить фрагмент находящийся между этих двух индексов (сами индексы включаются в вывод)

   Выделение подзадач:
   - Ввод последовательности
   - Выделение фрагмента с наибольшим количеством подряд идущих положительных элементов
   - Вывод фрагмента последовательности

   Далее алгоритм описан в укрупнённых блок в терминах выделенных подзадач

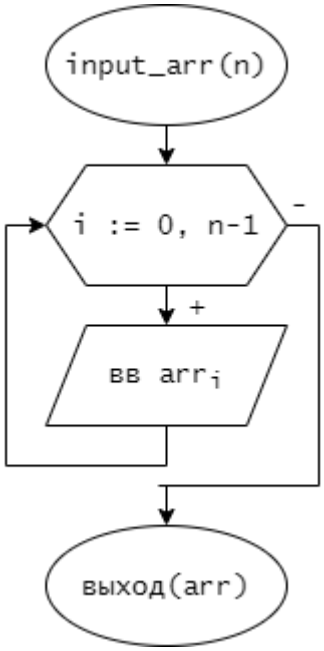
2. **Блок-схема с укрупнёнными блоками**



3. Спецификации подпрограмм

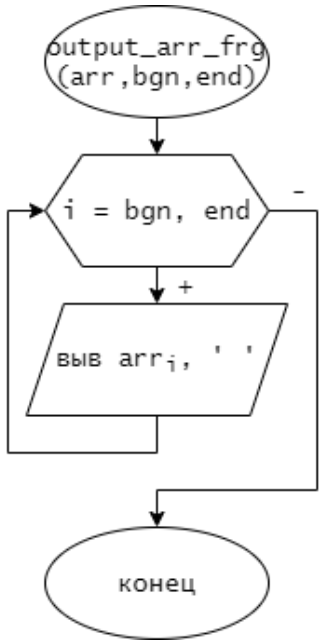
Спецификация процедуры `input_arr`

1) Заголовок: `void input_arr(float arr[], const size_t n)`
2) Назначение: ввод вещественного массива `arr` длины `n`
   Блок-схема:



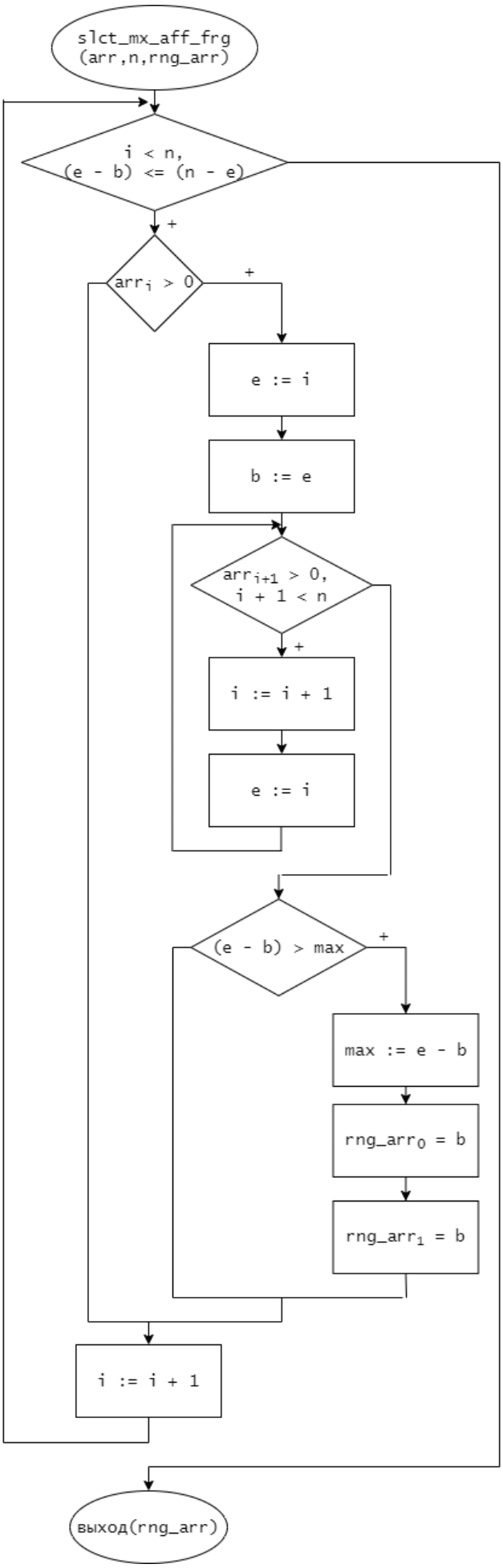
Спецификация процедуры `output_arr_frg`

1) Заголовок: `void output_arr_frg(const float arr[], const size_t bgn, const size_t end)`
2) Назначение: вывод фрагмента массива `arr` с индекса `bgn` по индекс `end`
   Блок-схема:



Спецификация процедуры `slct_mx_aff_frg`

1) Заголовок: `slct_mx_aff_frg(const float arr[], const size_t n, int rng_arr[])`
2) Назначение: выделяет фрагмент максимального размера с идущими подряд положительными элементами в массиве `arr` длины `n`. Начало и конец фрагмента заносит в массив `rng_arr` в под индексами 0 и 1 соответственно
   Блок-схема:


slct_mx_aff_frg
(arr,n,rng_arr)
i < n,
(e - b) <= (n - e)
+
arr$_i$ > 0
+
e := i
b := e
arr$_{i+1}$ > 0,
i + 1 < n
+
i := i + 1
e := i
(e - b) > max
+
max := e - b
rng_arr$_0$ = b
rng_arr$_1$ = b
i := i + 1
выход(rng_arr)

4. Тестовые данные

№	Вход	Выход
1	9 1.0 -2.22 3.4 4.1 5.01 –12.0 -32.02 9.9 10.1	3.4 4.1 5.0
2	6 -12.22 23.3 -1.1 13.37 2.28 14.48	13.4 2.28 14.48
3	5 2.3 -1.1 -23.6 -66.6 -5.5	2.3

5. Текст программы:

```c
#include <stdio.h>

///ввод массива
void input_arr(float arr[], const size_t n)
{
    for (size_t i = 0; i < n; ++i)
        scanf("%f",&arr[i]);
}

///выбор фрагмента (начало фрагмента и конец фрагмента) с максимальным
///количеством идущих подряд положительных чисел.
void slct_mx_aff_frg(const float arr[],const size_t n, int rng_arr[])
{
    size_t i = 0;
    int b = 0, e = 0, max = -1;

    while ((i < n) && ((e - b) <= (n - e)))
    {
        if (arr[i] > 0)
        {
            b = e = i;

            while ((arr[i + 1] > 0) && (i + 1 < n))
                e = ++i;

            if ((e - b) > max)
            {
                max = e - b;
                rng_arr[0] = b;
                rng_arr[1] = e;
            }
        }

        i++;
    }
}

///вывод массива.
void output_arr_frg(const float arr[], const size_t bgn, const size_t end)
{
    for (int i = bgn; i <= end; i++) {
        printf("%2.1f ",arr[i]);
    }
}

int main()
{
```

```
    printf("Input length of numbers sequence\n");
    size_t n;
    scanf("%ud",&n);

    float arr[n];
    printf("Input elements of numbers sequence\n");
    input_arr(arr, n);

    int rng_vals[] = {0, 0};

    slct_mx_aff_frg(arr, n, rng_vals);

    if ((rng_vals[0] == 0) && (rng_vals[1] == 0) && (arr[0] < 0))
        printf("There is no such fragment");
    else
    {
        printf("Length of fragment is %u\n",(rng_vals[1]-rng_vals[0]+1));
        output_arr_frg(arr,rng_vals[0],rng_vals[1]);
    }
}
```

6. Результаты работы программы

Пример №1

```
Input length of numbers sequence
9
Input elements of numbers sequence
1.0 -2.22 3.4 4.1 5.01 -12.0 -32.02 9.9 10.1
3.4 4.1 5.0
```

Пример №2

```
Input length of numbers sequence
6
Input elements of numbers sequence
-12.22 23.3 -1.1 13.37 2.28 14.48
13.4 2.3 14.5
```

Пример №3

```
Input length of numbers sequence
5
Input elements of numbers sequence
2.3 -1.1 -23.6 -66.6 -5.5
2.3
```

7. Анализ допущенных ошибок

- Выход за пределы массива из-за чего происходил захват «мусора» при выделении фрагмента