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 разложить на множители, то результат должен быть выведен следующим образом:
7. Подобрать тестовые данные

**Задание варианта №17**

Дана вещественная последовательность . Определить максимальное количество идущих подряд положительных членов последовательности. Вывести найденный фрагмент.

**Выполнение:**

1. **Описание алгоритма и выделение подзадач**

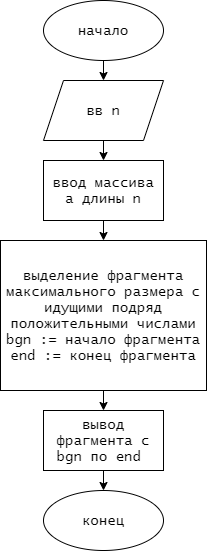
Исходя из того, что нам нужно определить максимальный фрагмент идущих подряд положительных чисел, можно запоминать индекс начала фрагмента и индекс конца, а затем выводить фрагмент находящийся между этих двух индексов (сами индексы включаются в вывод)

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

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

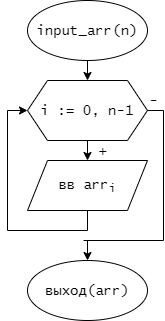


1. **Спецификации подпрограмм**

Спецификация процедуры input\_arr

1. Заголовок: void input\_arr(float arr[], const size\_t n)
2. Назначение: ввод вещественного массива arr длины n

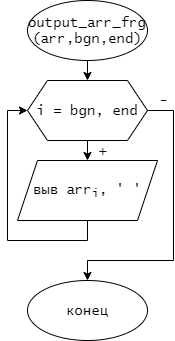
Блок-схема:



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

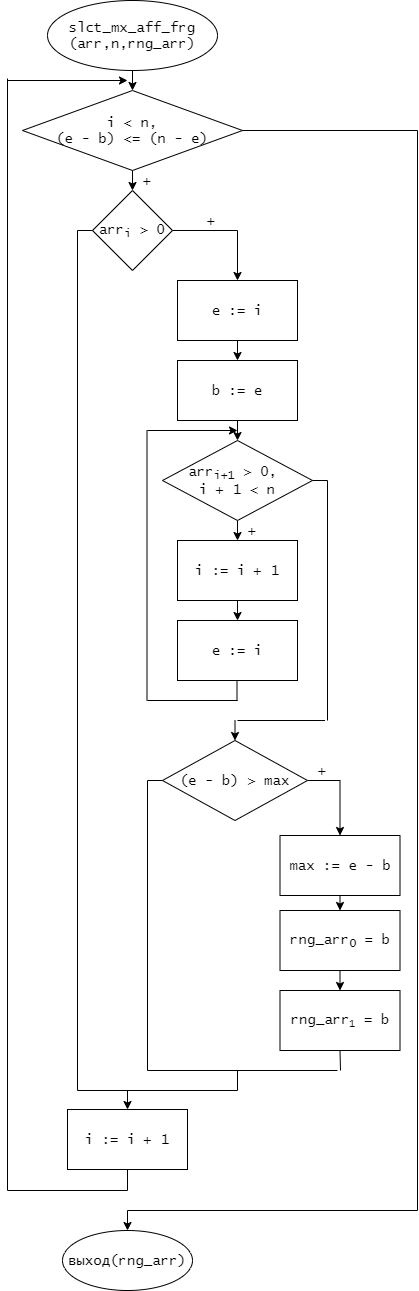
Блок-схема:



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:



1. **Тестовые данные**

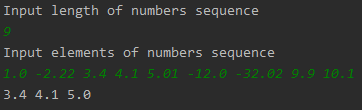
|  |  |  |
| --- | --- | --- |
| № | Вход | Выход |
| 1 | 9  1.0 -2.22 3.4 4.1 5.01 –12.0 -32.02 9.9 10.1 | 3.4 4.1 5.0 |
| 2 | 6  -12.22 23.3 -1.1 13.37 2.28 14.48 | 13.4 2.28 14.48 |
| 3 | 5  2.3 -1.1 -23.6 -66.6 -5.5 | 2.3 |

1. **Текст программы:**

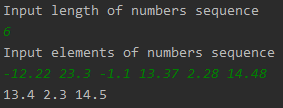
#include <stdio.h>  
  
///ввод массива  
void input\_arr(float arr[], const size\_t n)  
{  
 for (size\_t i = 0; i < n; ++i)  
 scanf("%f",&arr[i]);  
}  
  
///выбор фрагмента (начало фрагмента и конец фрагмента) с максимальным  
///количеством идущих подряд положительных чисел.  
void slct\_mx\_aff\_frg(const 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]);  
 }  
}

1. **Результаты работы программы**

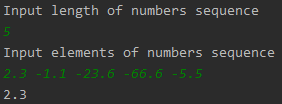
*Пример №1*



*Пример №2*



*Пример №3*



1. **Анализ допущенных ошибок**

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