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 максимальное количество идущих подряд положительных членов последовательности, можно запоминать индекс начала фрагмента с максимальным количеством идущих подряд положительных членов последовательности и индекс конца.

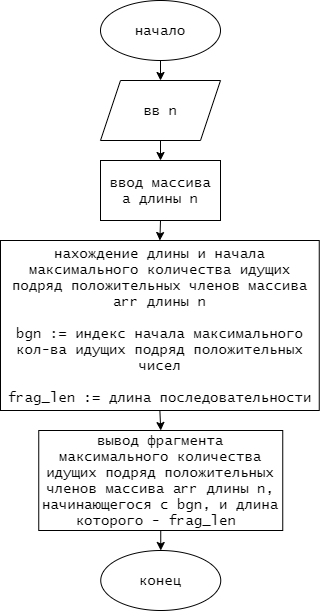
Затем выводим количество элементов: . А после сами элементы, которые располагаются между индексами и .

Выделим следующие подзадачи:

* Ввод последовательности
* Нахождение максимального количества идущих подряд положительных членов последовательности
* Вывод максимального количества идущих подряд положительных членов последовательности

Далее алгоритм описан в укрупнённых блок в терминах выделенных подзадач

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

****

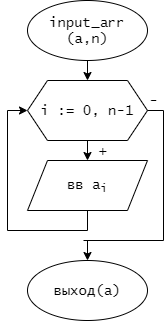
1. **Спецификации подпрограмм**

Ввод последовательности

Спецификация процедуры input\_arr

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

Блок-схема:

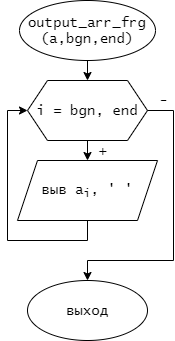


Вывод максимального количества идущих подряд членов последовательности

Спецификация процедуры output\_arr\_frg

1. Заголовок: void output\_arr\_frg(const float a[], const size\_t bgn, const size\_t end)
2. Назначение: вывод фрагмента массива a с индекса bgn по индекс end

Блок-схема:

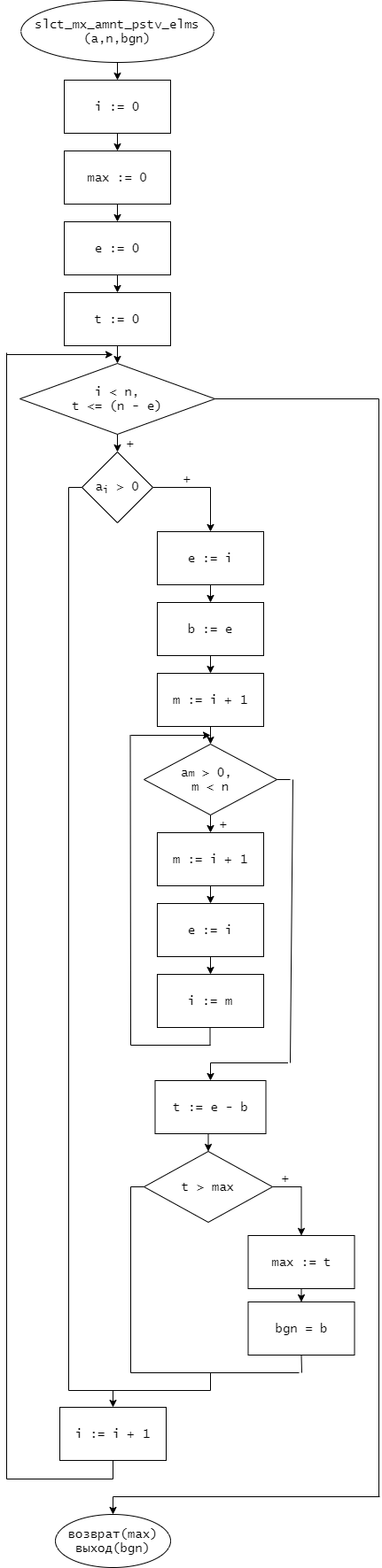


Нахождение максимального количества идущих подряд положительных членов последовательности

Спецификация процедуры get\_len\_mx\_amnt\_pstv\_elms

1. Заголовок: get\_len\_mx\_amnt\_pstv\_elms(const float a[], const size\_t n, size\_t bgn)
2. Назначение: возвращает длину максимального количества идущих подряд положительных членов массива a длины n. Побочный эффект: в bgn помещается индекс начала фрагмента.

Блок-схема:



1. **Тестовые данные**

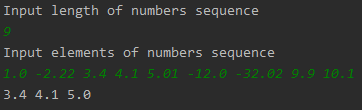
|  |  |  |
| --- | --- | --- |
| № | Вход | Выход |
| 1 | 9  1.0 -2.22 3.4 4.1 5.01 –12.0 -32.02 9.9 10.1 | 3.4 4.1 5.0 |
| 2 | 6  -12.22 23.3 -1.1 13.37 2.28 14.48 | 13.4 2.28 14.48 |
| 3 | 5  2.3 -1.1 -23.6 -66.6 -5.5 | 2.3 |

1. **Текст программы:**

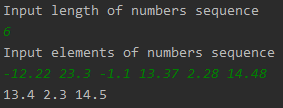
#include <stdio.h>  
  
///ввод массива arr длины n.  
void input\_arr(float a[], const size\_t n)  
{  
 for (size\_t i = 0; i < n; ++i)  
 scanf("%f", &a[i]);  
}  
  
///вывод проивзольного количества идущих подряд элементов  
///массива arr длины n начинающегося с bgn и  
///оканчивающегося на end.  
void output\_arr\_frg(const float a[], const size\_t bgn, const size\_t end)  
{  
 for (size\_t i = bgn; i <= end; i++)  
 {  
 printf("%f ",a[i]);  
 }  
}  
  
///возвращает длину максимального количества идущих подряд  
///положительных членов массива a длины n.  
///Побочный эффект: в bgn помещается индекс начала фрагмента.  
size\_t get\_len\_mx\_amnt\_pstv\_elms(const float a[], const size\_t n, size\_t \*bgn)  
{  
 size\_t i = 0,  
 b,  
 e = 0,  
 t = 0,  
 max = 0;  
  
 while ((i < n) && (t <= (n - e)))  
 {  
 if (a[i] > 0)  
 {  
 b = e = i;  
  
 while ((a[i+1] > 0) && (i + 1 < n))  
 e = ++i;  
  
 t = e - b;  
  
 if (t >= max)  
 {  
 max = t;  
 \*bgn = b;  
 }  
 }  
  
 i++;  
 }  
  
 return max;  
}  
  
int main()  
{  
 printf("Input length of numbers sequence\n");  
 size\_t n;  
 scanf("%ud", &n);  
  
 float a[n];  
 printf("Input elements of numbers sequence\n");  
 input\_arr(a, n);  
  
 size\_t bgn = 0,  
 frag\_len;  
  
 frag\_len = get\_len\_mx\_amnt\_pstv\_elms(a, n, &bgn);  
  
 if ((bgn == 0) && (frag\_len == 0) && (a[0] < 0))  
 {  
 printf("There is no such fragment");  
 }  
 else  
 {  
 printf("Length of fragment is %u\n", frag\_len+1);  
 output\_arr\_frg(a, bgn, bgn + frag\_len);  
 }  
}

1. **Результаты работы программы**

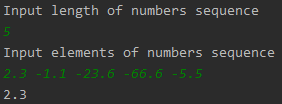
*Пример №1*



*Пример №2*



*Пример №3*



1. **Анализ допущенных ошибок**

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