МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им.

Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий

Кафедра «Информационные радиосистемы»

**Индуктивные операции и функции высших порядков**

**(Вариант 28)**

Выполнил:

Студент гр. 22-Рз Наумов А.А.

Проверил:

к.т.н., доцент кафедры ИРС Сидоров С.Б.

Нижний Новгород,

2024 г.

Оглавление

[Задание по варианту 3](#__RefHeading___Toc2465_4070117755)

[Контрольная работа № 1. 4](#__RefHeading___Toc2035_2758904751)

[1.1. Постановка задачи 4](#__RefHeading___Toc2041_2758904751)

[1.2. Алгоритм индуктивной обработки 4](#__RefHeading___Toc11489_2758904751)

[1.3. Архитектура программной реализации вычислителя. 7](#__RefHeading___Toc6530_2651826584)

[АТД Application 8](#__RefHeading___Toc11491_2758904751)

[application.h 8](#__RefHeading___Toc11493_2758904751)

[АТД Vector 9](#__RefHeading___Toc6532_2651826584)

[Приложение 1. 10](#__RefHeading___Toc3283_4070117755)

[Приложение 2. 16](#__RefHeading___Toc2053_2758904751)

[Контрольная работа №2 21](#__RefHeading___Toc2047_2758904751)

[2.1 Архитектура программной системы 21](#__RefHeading___Toc11495_2758904751)

[2.2 Использование индуктивного вычислителя 22](#__RefHeading___Toc11497_2758904751)

[Приложение 1 23](#__RefHeading___Toc11499_2758904751)

[Приложение 2 29](#__RefHeading___Toc11501_2758904751)

# Задание по варианту

Обнаружение наиболее длинного участка монотонного возрастания значений последовательных элементов, при условии что разность значений последнего и первого элементов участка не менее чем D. Результатом является интервал: (начальный номер, конечный номер). Тип элемента — целочисленный.

# Контрольная работа № 1.

**«Реализация индуктивной обработки**

**последовательности элементов»**

## 1.1. Постановка задачи

Освоение способов разработки алгоритмов выполнения индуктивных операций над последовательностью данных, построение индуктивных функций методом индуктивных расширений. Изучение общей схемы программной реализации индуктивной функции и схемы обработки последовательности элементов с использованием индуктивной функции.

На вход системы последовательно и неограниченно во времени поступают элементы xi, где i — порядковый номер элемента, начиная с 1. Реализовать указанную в варианте обработку f(X) последовательности элементов X=⟨x1,x2,x3,…⟩ с использованием схемы индуктивной обработки на пространстве последовательностей. Полученный набор выходных значений Y=⟨y1,y2,y3,…⟩ рассматривается как результирующая последовательность. Значения элементов исходной последовательности должны запрашиваться у пользователя (приниматься со стандартного устройства ввода) по одному за раз. Сформированные выходные значения требуется выдавать сразу после их формирования на стандартное устройство вывода. Реализация обработки должна быть приведена в отдельном модуле.  
 В качестве примера рассмотрим следующий набор данных:

## 1.2. Алгоритм индуктивной обработки

Основной алгоритм программы содержится в функции appProcess, имея значение , для осуществления пересчета необходимо соблюсти следующие условия

Рассмотрим каждое из условий:

## 1.3. Архитектура программной реализации вычислителя.

Разберём используемые абстрактные типы данных.

## 

АТД Application

Определяется структурный тип данных **Application**, содержащий пять полей: Vector valueArray, int constD, int finalLeft, int finalRight, int finalConsequenceStreak.

Первое — вектор, собираемый из введенной пользователем последовательности.

Второе — константа D, так же, получаемая через стандартное устройство ввода. Три последних элемента, начинающихся с слова final, хранят результирующие данные, соответственно своим названиям. Финальный левый и правый индекс, а так же, длину конечной, совпавшей, последовательности. Как Вы успели заметить, я написал код на плюсах в camelCase, да. На следующем курсе все будет в snake\_case.

application.h

В заголовочном файле application.h объявляется глобальная функция int appRun(Application& app). Эта функция отвечает за исполнение работу программы, а соответственно, напрямую взаимодействует с АТД A*pplication,* а именно: получает данные от пользователя, обрабатывает их, и выводит результат в стандартное устройство вывода. При успешном выполнении возвращает значение 0, если произошла ошибка на одном из этапов, возвращает значение 1.

Для выполнения поставленных, условием, задач, в .h файле объявляется прототип функции, а в .cpp файле, определяются под-функции appRun, а именно:

bool appInitializeData(Application &app);  
 bool appGetConstantD(Application &app);  
 bool appProcessDataIntoFinalResult(Application &app);  
 bool appGetOutputToUser(Application &app);

Рассмотрим каждую функцию в отдельности:

bool appInitializeData(Application &app)- отвечает за сборку вектора исходных данных, полученных от пользователя.  
 bool appGetConstantD(Application &app)- Получение числа D.  
 bool appProcessDataIntoFinalResult(Application &app)- Содержит основную логическую функцию обработки.  
 bool appGetOutputToUser(Application &app)- выводит конечный результат на стандартное устройство вывода.

# Приложение 1.

# Приложение 2.

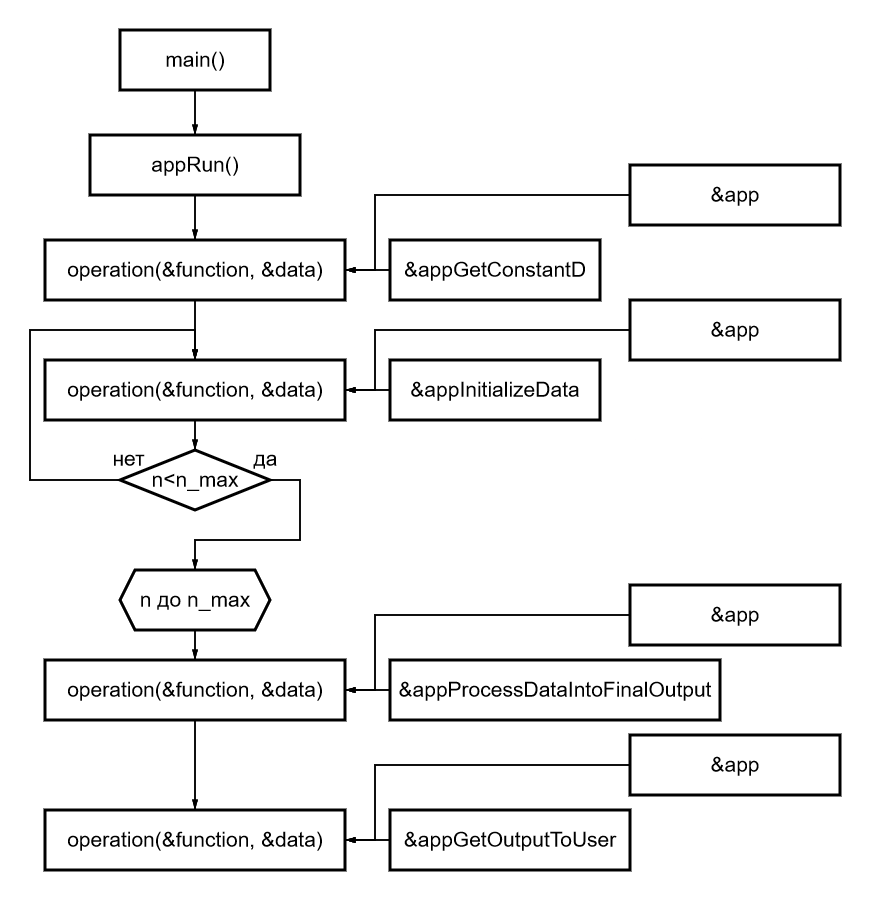
# Контрольная работа №2

**«Настройка индуктивного вычислителя**

**с использованием функции обратного вызова»**

## 2.1 Архитектура программной системы

Основная логика вычисления не претерпела изменений, с контрольной работы 1. Изменился стиль вызова и способ обмена данными между функциями. Теперь мы объявляем АТД Application в основной функции appRun, вместо main.

В следствие внедрения данной методологии вызова функций, изменилась структура программы. Представлена ниже.

## 2.2 Использование индуктивного вычислителя

Функция operation, которая соответствует назначению — callback из условия работы. Она принимает ссылку на функцию и ссылку на объявленный тип данных. В результате мы избегаем копирования одного и того же типа данных, по несколько раз.

В следствие использования данной структуры, можно наладить общение нескольких программ между друг другом, вследствие унифицированного элемента функции и набора данных.

//application.h

typedef bool (\*Callback)(void \*object);  
 bool operation(Callback callback, void \*data);

|  |  |
| --- | --- |
| Элемент | Назначение |
| typedef | аллиас на функцию |
| bool | тип функции |
| (\*Callback) | указатель на тип функции |
| (void \*object) | указатель на тип принимаемого аргумента, и его название |

// application.cpp

bool operation(Callback callback, void \*data) {  
 return (\*callback)(data);  
 }

|  |  |
| --- | --- |
| Элемент | Назначение |
| bool | тип функции |
| operataion | название |
| (Callback callback | тип и имя принимаемого аргумента |
| void \*data) | указатель на любой тип данных, имя аргумента |
| return | возвращаемое значение |
| (\*callback) | результат работы вызываемой функции |
| (data) | данные с которыми работала программа. |

# Приложение 1

# Приложение 2