

# Алгоритмы и структуры данных

# Литература:

- Д. Кнут. Искусство программирования для ЭВМ. Т. 1-3, М.: Мир, 1978, 1995 и др..
- Н. Вирт. Алгоритмы и структуры данных. М.: Мир, 1989.

# Концепция типа данных

Информация, которая должна обрабатываться на компьютере является абстракцией, отображением некоторого фрагмента реального мира. А именно того фрагмента, который является предметной областью решаемой задачи. Для ее решения вначале строится *информационная*, а в общем случае *математическая модель* изучаемой предметной области и выбирается существующий или строится новый алгоритм решения задачи.

Информация всегда *материализуется*, представляется в форме сообщения. Сообщение в общем случае представляет собой некоторый *зарегистрированный физический сигнал*. Сигнал — это *изменение во времени или пространстве некоторого объекта*, в частности, параметра некоторой физической величины, например индукции магнитного поля (при хранении информации, *точнее сообщения* на магнитных носителях) или уровня напряжения в электрической цепи (в микросхемах процессора или оперативной памяти).

*Дискретное* сообщение — это последовательность *знаков* (значений сигнала) из некоторого *конечного алфавита* (конечного набора значений параметра сигнала), в частности, для компьютера это *последовательность знаков двоичного алфавита, то есть последовательность битов*.

Компьютерные данные это дискретные сообщения, которые представлены в форме, используемой в компьютере, *понятной компьютеру*. Для процессора компьютера любые данные представляют собой **неструктурированную** последовательность битов (иногда используют термин поток битов).

Конкретная интерпретация этой последовательности зависит от программы, от *формы представления и структуры данных*, которые выбраны *программистом*. Это выбор, в конечном счёте, зависит от решаемой задачи и удобства выполнения действий над данными.

К данным в программах относятся:

- ❖ **Непосредственные значения** это *неизменные* объекты программы, которые представляют сами себя: числа (25, 1.34E-20), символы ('A', '!') , строки ('Введите элементы матрицы');
- ❖ **Константы** – это имена, закрепляемые за некоторыми значениями (const pi=3.1415926).
- ❖ **Переменные** это объекты, которые могут принимать значение, сохранять его без изменения, и изменять его при выполнении определенных действий (var k:integer, x:real, a:array[1..3,1..5]).
- ❖ **Значения выражений и функций**. Выражения и функции– это записанные определённым способом правила вычисления значений:  $k*x + \text{sqrt}(x)$ .

Для отображения особенностей представления в компьютере данных различной природы в информатике, в компьютерных дисциплинах используется важнейшая концепция *типа данных*.

***Тип данных представляет собой важнейшую характеристику, которая определяет:***

- ❖ множество допустимых значений;
- ❖ множество операций, которые могут выполняться над значением;
- ❖ структуру значения (скаляр, вектор и т.д.);
- ❖ способ машинного представления значения.

**Основные принципы концепции типа данных  
в языках программирования:**

- ❖ Тип константы, переменной или выражения может быть определен по внешнему виду (по изображению) или по описанию без выполнения каких-либо вычислений.
- ❖ Любая операция или функция требует аргументов и возвращает результат вполне определенного типа. Типы аргументов и результатов операций определяется по вполне определенным правилам языка.

# Разновидности типов и структур данных

В информатике используется большое количество различных *типов*, различных *структур данных*, которые применяются для моделирования объектов, встречающихся в рассматриваемых задачах.



Значение скалярного (*простого, атомарного*) типа представлено ровно одним компонентом ( пример: время, температура).

Значение структурированного (*составного*) типа представлено более чем одним компонентом (пример: вектор, матрица, таблица и т.д.).

Если структура данного по ходу выполнения алгоритма не изменяется, то такая структура считается статической, Статические структуры данных существуют в неизменном виде в течение **всего времени исполнения алгоритма**.

Динамические структуры создаются, изменяются и уничтожаются по мере необходимости **в любой момент исполнения алгоритма**.

Различают **предопределенные** (предварительно определенные) - **стандартные** и **определяемые в программе** типы. Для *стандартных* типов в описании языка программирования заданы все его характеристики – множество значений, множество операций, структура и машинное представление значения. Для *вновь определяемых* типов в языке предусмотрен механизм указания в программе множества значений и структура значения. Обычно новый тип строится на базе имеющихся стандартных. Поэтому множество операций и машинное представление таких типов фиксировано в описании языка.

## **Статические типы (структуры данных)**

□ скалярные (простые, атомарные) типы:

- ❖ целый;
- ❖ вещественный;
- ❖ логический (булевский);
- ❖ символьный;

□ структурированные (составные) типы:

- ❖ массив;
- ❖ запись;
- ❖ файл (последовательность);
- ❖ множество;
- ❖ объектовый (класс) тип;

□ всевозможные комбинации скалярных и структурированных типов;

□ ссылочный тип.

Наиболее часто используемые **предопределенные скалярные** типы: целый (*integer*), вещественный (*real*), символьный (*char*), логический (*boolean*).

### Тип *integer*

Целочисленные точные значения. Примеры: **73, -98, 5, 19674**.

Машинное представление: формат с фиксированной точкой. Диапазон значений определяется длиной поля. Операции: **+, -, \*, div, mod, =, <**, и т.д.

### Тип *real*

Нецелые приближенные значения. Примеры: **0.195, -91.84, 5.0**

Машинное представление: формат с плавающей точкой. Диапазон и точность значений определяется длиной поля. Операции: **+, -, \*, /, =, <**, и т.д.

### Тип *char*

Одиночные символы текстов. Примеры: **'a', '!', '5'**.

Машинное представление: формат ASCII. Множество значений определяется кодовой таблицей и возможностями клавиатуры. Операции: **+, =, <**, и т.д.

### Тип *boolean*

Два логических значения **false** и **true**. Причем, **false < true**.

Машинное представление — нулевое и единичное значение бита: false кодируется 0, true — 1. Операции: **¬, ∨, ∧, =, <** и т.д.



Различают **дискретные** и **непрерывные** скалярные типы. Множество значений *дискретного* типа конечное или счетное. Множество значений *непрерывного* типа более чем счетное. К дискретным стандартным типам относятся целый, символьный и логический. К непрерывным стандартным типам относится вещественный.

Основные механизмы построения новых скалярных дискретных типов: **перечисление**, **ограничение**. В определении *перечисляемых* типов фиксируется список всех возможных значений, множество операций определяется в языке заранее. В определении *ограниченных* типов в качестве множества допустимых значений фиксируется *подмножество* множества значений некоторого дискретного типа, который в этом случае называется базовым типом по отношению к определяемому.

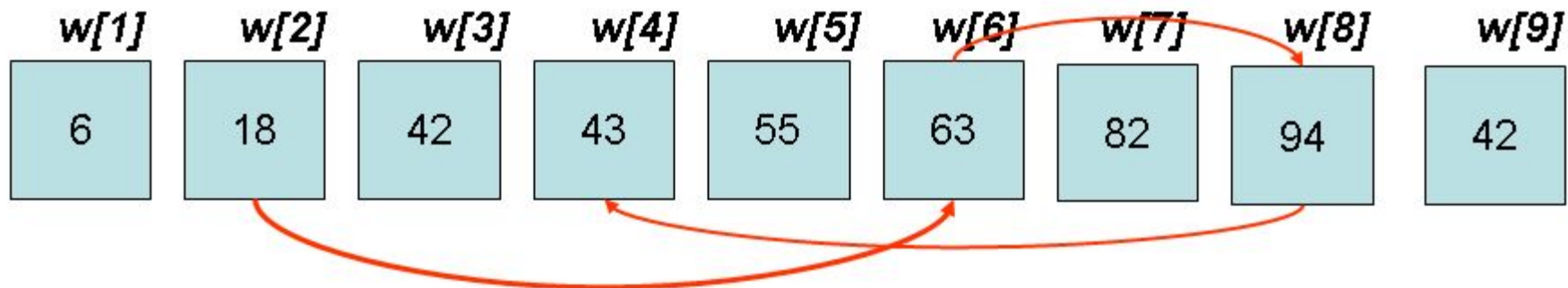
**Структурированные (составные) типы** характеризуются: количеством и возможным типом компонентов значения, а также способом доступа к отдельному компоненту значения.

### ***Массив или регулярный тип***

Структуры аналогичные векторам и матрицам в информатике принято называть **массивами**. Все элементы массива должны быть одного и того же типа.

$$\bar{x} = \{x_1, x_2, x_3\}; \bar{w} = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9); A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Для доступа (обращения) к отдельному элементу массива используется индекс или несколько индексов (**w[5]; w[i+2]; A[1,2]**). Индексы могут быть выражениями, значения которых могут произвольным образом изменяться в заранее заданных границах. Поэтому говорят, что к элементам массивов имеется **прямой доступ**.



## Запись или комбинированный тип

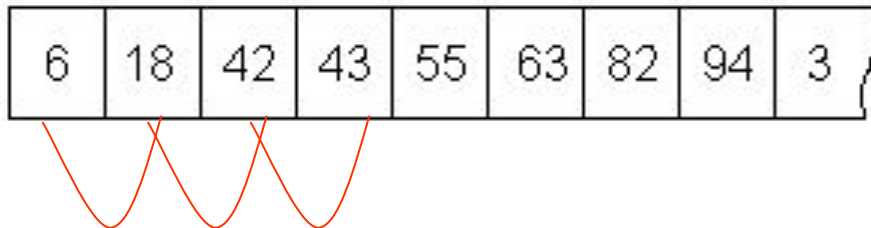
День Победы:  
Полёт Гагарина:

День	Месяц	Год
9	май	1945
12	апрель	1961

Структуры, аналогичные строкам таблицы, называют **записями**. Компоненты записей принято называть **полями**. Различные поля (столбцы таблицы) могут быть разных типов. Для доступа к отдельным полям записи используются их фиксированные и неизменные имена. Например: **День Победы. Месяц := май**. Поля могут выбираться для обработки в произвольном порядке, поэтому говорят, что доступ к компонентам записи прямой.

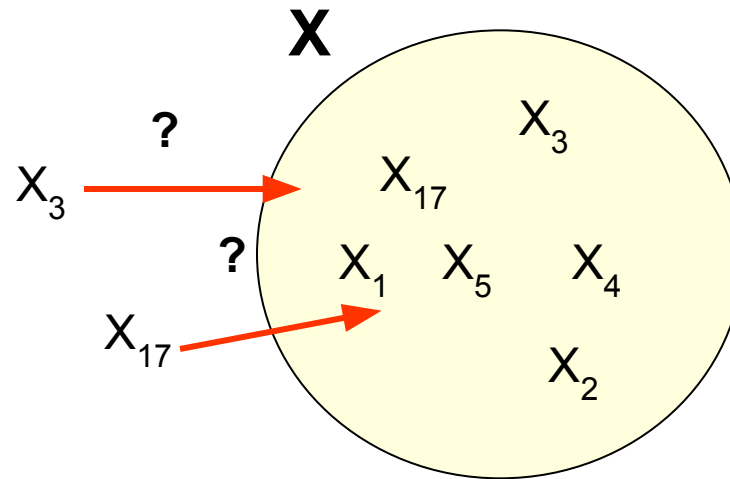
### Файл (последовательность)

Основной структурой данных, которая используется для хранения информации на внешних устройствах (магнитных дисках, лентах и т.д.) являются **файлы** или **последовательности**. Считается, что файл всегда находится на внешнем устройстве. При этом количество компонентов файла неизвестно, все компоненты должны быть одного и того же типа. Доступ к компонентам — **последовательный**.



# Множество

Во многих математических и информационных задачах возникает необходимость в прямом или косвенном использовании основного математического объекта множества. Соответствующая множеству тип данных по определению относится к структурированным, так как в общем случае множество может состоять более чем из одного элемента, и при этом со всеми элементами множества приходится выполнять операции как с единым целым. Количество элементов в множестве заранее не определяется, и с течением времени оно может изменяться. Все элементы множества должны быть одного и того же типа. Доступа к отдельным элементам множества нет. Можно только узнать принадлежит элемент множеству или нет, включить элемент в множество или исключить его из множества. Предусмотрены также стандартные операции над множествами: объединение, пересечение, вычитание и т.д.



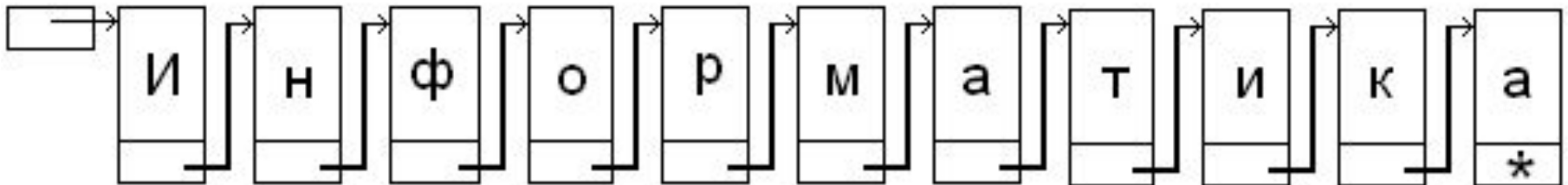
## ***Динамические структуры данных***

У данных с динамической структурой с течением времени изменяется сама структура, а не только количество элементов, как у файлов или последовательностей. Базовыми динамическими структурами данных являются:

- объект;
- линейный список;
- дерево;
- граф.

### ***Линейный список***

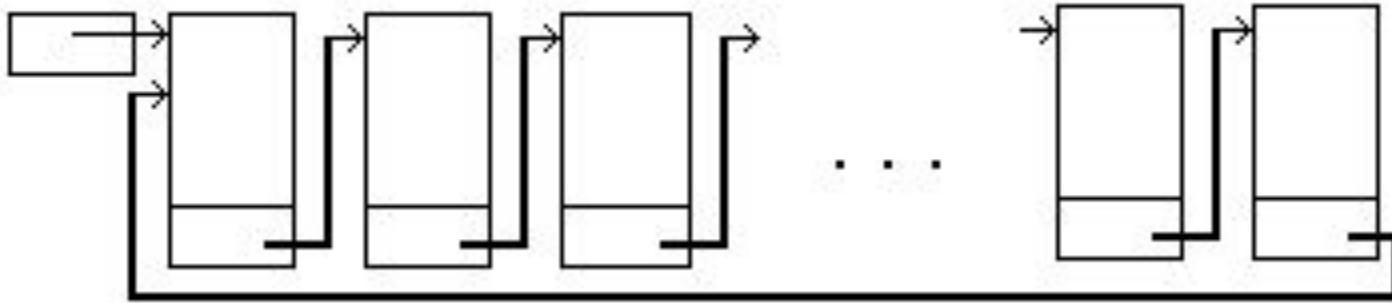
У линейного списка каждый элемент связан с предшествующим ему. У линейного списка известно, какой элемент находится в начале списка, какой в конце, а также, какой элемент стоит перед текущим. В линейном списке переходить от текущего элемента к следующему можно только с помощью указанных связей между соседними элементами.



В целом получается цепочка элементов, в которой можно осуществлять поиск, в которую можно вставлять элементы или исключать их.

На базе линейного списка организуются много других типов динамических структур. Это в частности: **кольца**, **очереди**, **деки** и **стеки**.

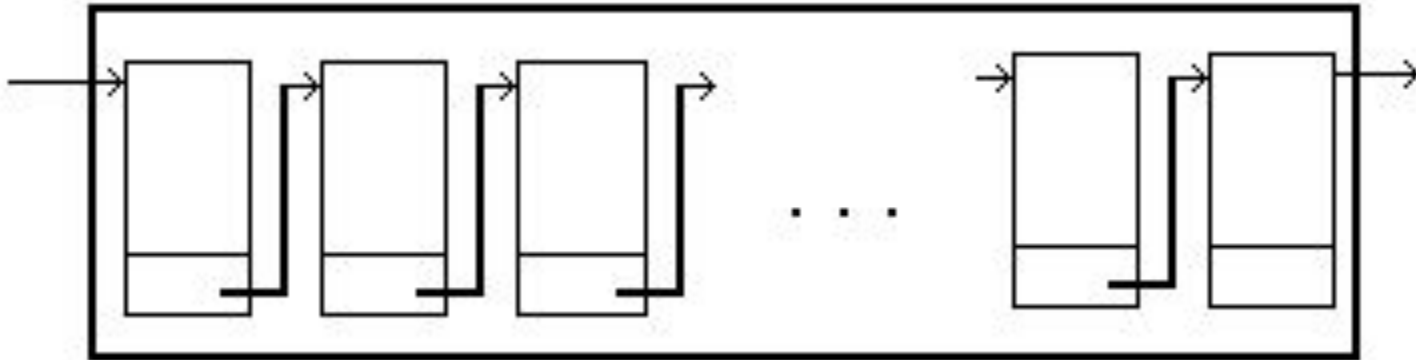
### Структура кольца



Отличие кольца от линейного списка в том, что у кольца имеется связь между последним элементом списка и его первым элементом.

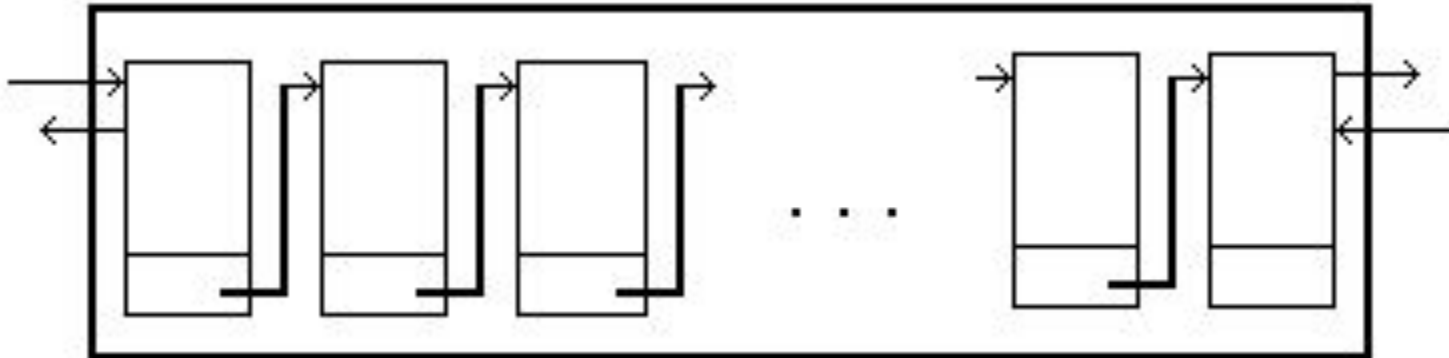
У линейного списка и у кольца возможен доступ к любому элементу структуры. Для этого нужно последовательно перемещаться от одного элемента к другому. Во многих реальных ситуациях такой доступ отсутствует. Можно взаимодействовать только с первым и последним элементами или же только с одним из них. Для моделирования таких объектов используются очереди, деки и стеки.

## Структура очереди



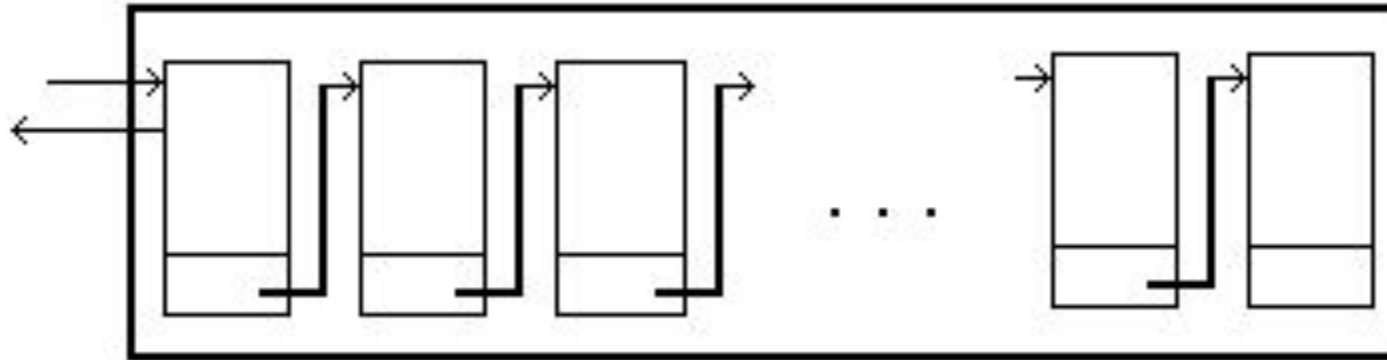
У очереди доступен для включения конец, а для исключения (выборки) — начало. Элемент, поступивший в очередь раньше и обслуживается раньше. Говорят, что очередь это структура с дисциплиной обслуживания **FIFO** (**F**irst **I**n, **F**irst **O**ut) — «первый пришёл, первый ушел».

## Структура дека



У дека оба конца доступны, как для включения, так и для выборки. Таким образом, можно сказать, что дек — это двусторонняя очередь.

## Структура стека



У стека для взаимодействия доступна только один конец структуры — вершина стека. И включение нового элемента в стек и выборка последнего ранее включенного идет через вершину стека. Таким образом на обслуживание попадает первым элемент, поступивший последним. Говорят, что стек — это структура с дисциплиной обслуживания **LIFO** (**L**ast **I**n, **F**irst **O**ut) — «последним пришёл, первым ушёл».



**СПАСИБО ЗА ВНИМАНИЕ!**