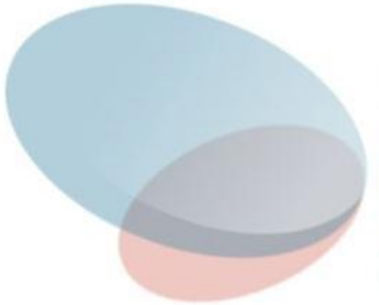


Алгоритмы и структуры данных
Лекция 1
Основные понятия

Понятие алгоритма и его свойства

Алгоритм – точное предписание, определяющее вычислительный процесс, ведущий от **варьируемых** начальных данных к искомому результату.

Алгоритм – строгая, четкая последовательность математических и логических операций, приводящая к решению задачи.

Свойства алгоритма

- ***Детерминированность*** (*определенность, точность, однозначность*). При задании одних и тех же исходных данных несколько раз алгоритм должен выполняться абсолютно одинаково и всегда будет получен один и тот же результат.

  На каждом шаге выполнения алгоритма всегда точно известно, что делать дальше, каждое действие однозначно понятно исполнителю и не может быть истолковано неопределенно. Благодаря этому свойству выполнение алгоритма носит механический характер.

Понятие алгоритма и его свойства

- ***Массовость.*** С помощью алгоритма можно решать не одну конкретную задачу, а любую задачу из некоторого класса однотипных задач при **всех допустимых** значениях исходных данных.
- ***Результативность*** (*направленность*). Выполнение алгоритма обязательно должно привести к решению поставленной задачи, либо к сообщению о том, что при заданных исходных величинах задачу решить невозможно. Алгоритмический процесс не может обрываться безрезультатно.
- ***Дискретность.*** Алгоритм состоит из последовательности отдельных шагов – элементарных действий, выполнение которых не представляет сложности. Благодаря этому свойству алгоритм может быть реализован на компьютере.

Понятие алгоритма и его свойства

- ***Конечность*** (*финитность*). Последовательность элементарных действий алгоритма не может быть бесконечной, неограниченной, хотя может быть очень длинной (если требуется, например, больш*ая* точность вычислений).
- ***Корректность.*** Если алгоритм создан для решения определенной задачи, то для всех исходных данных он должен всегда давать правильный результат и ни для каких исходных данных не будет получен неправильный результат. Если хотя бы один из полученных результатов противоречит хотя бы одному из ранее установленных и получивших признание фактов, алгоритм нельзя признать корректным.

Если разработанная последовательность действий не обладает хотя бы одним из перечисленных выше свойств, то она не может считаться алгоритмом

Понятие структуры данных

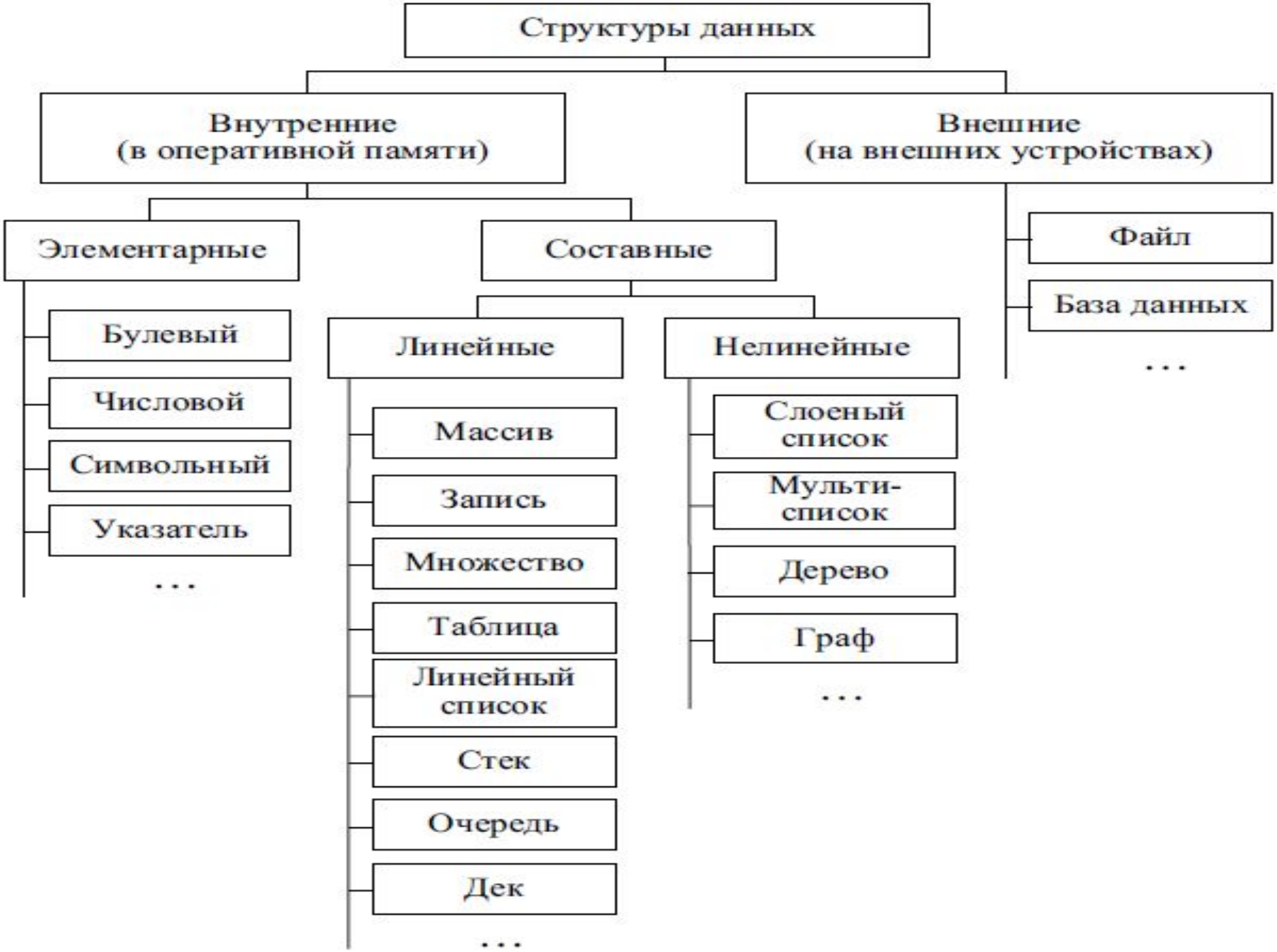
Под **структурой данных** в общем случае понимают множество элементов данных и множество связей между ними.

Понятие «**физическая структура данных**» отражает способ физического представления данных в памяти компьютера и называется еще структурой хранения, внутренней структурой или структурой памяти.

Структура данных, рассматриваемая без учета ее представления в машинной памяти, называется **абстрактной** или **логической структурой**.

В зависимости от размещения физических структур, а соответственно, и доступа к ним, различают **внутренние** (находятся в оперативной памяти) и **внешние** (на внешних устройствах) структуры данных.

Классификация структур данных



Классификация структур данных

Различаются **элементарные (простые, базовые, примитивные**) структуры данных и **составные (интегрированные, композитные, сложные**).

- **Элементарными** называются такие структуры данных, которые **не могут быть расчленены на составные части, большие, чем биты**.

  В конкретной системе программирования всегда можно заранее сказать, каков будет размер элементарного данного и каково его размещение в памяти. **С логической точки зрения** элементарные данные **являются неделимыми единицами.**

- **Составными называются такие структуры данных, составными частями которых являются другие структуры данных – элементарные или в свою очередь составные.**

  Важный признак составной структуры данных – **характер упорядоченности ее частей**. По этому признаку структуры можно делить на **линейные** и **нелинейные** структуры.

Статические и динамические структуры данных

Важный признак структуры данных – ее **изменчивость**, т.е. изменение **числа элементов** и/или **связей** между составными частями структуры и **адресов расположения во внешней памяти**. По признаку изменчивости различают структуры **статические и динамические**.

Для **статической структуры** данных эти характеристики остаются постоянными.

Динамическая структура данных – это структура данных, определяющие характеристики которой могут изменяться на протяжении ее существования.

Память под динамические структуры данных выделяется и освобождается по мере необходимости.

Динамические структуры, по определению, ***характеризуются отсутствием физической смежности*** элементов структуры в памяти, ***непостоянством*** и непредсказуемостью ***размера*** (числа элементов) структуры в процессе ее обработки.

Адрес элемента динамической структуры не может быть вычислен из адреса начального или предыдущего элемента. **Для установления связи между элементами динамической структуры используются указатели**, через которые устанавливаются **явные связи** между элементами.
Такое *представление* данных в памяти называется ***связным***.

Динамические структуры данных

Характеристики динамической структуры данных

- она не имеет имени;
- ей выделяется память в процессе выполнения программы;
- количество элементов структуры может не фиксироваться;
- размерность структуры может меняться в процессе выполнения программы;
- в процессе выполнения программы может меняться характер взаимосвязи между элементами структуры.

Каждой динамической структуре данных сопоставляется статическая переменная типа указатель (ее значение – адрес этого объекта), посредством которой осуществляется доступ к динамической структуре.

Сами динамические величины не требуют описания в программе, поскольку во **время компиляции память под них не выделяется**. Во *время компиляции память* выделяется только под статические величины. **Указатели – это статические величины**, поэтому они требуют описания.

Динамические структуры данных

Достоинства связного представления данных – в возможности обеспечения значительной изменчивости структур:

- размер структуры ограничивается только доступным объемом машинной памяти;
- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
- большая гибкость структуры.

Основные недостатки связного представления данных:

- на поля, содержащие указатели для связывания элементов друг с другом, расходуется дополнительная память;
- доступ к элементам *связной структуры* может быть менее эффективным по времени.

Классификация динамических структур данных

- стек;
- *дек*;
- очередь;
- однонаправленные (односвязные) списки;
- *двунаправленные* (*двусвязные*) списки;
- циклические списки;
- *бинарные деревья*
- *и т.д.*

Они отличаются способом связи отдельных элементов и/или допустимыми операциями.

Типы данных

В языках программирования понятие «**структуры данных**» тесно связано с понятием «**типы данных**».

Информация по каждому типу однозначно определяет:

– структуру (способ) хранения данных указанного типа, т. е.

- **выделение памяти**,
- **представление данных в ней и**
- **метод доступа к данным**;

– **множество допустимых значений**, которые может иметь тот или иной объект описываемого типа;

– **набор допустимых операций**, которые применимы к объекту описываемого типа.

Элементарные структуры данных

Данные **элементарных типов** представляют собой единое и неделимое целое.

В каждый момент времени они могут принимать только одно значение.

С помощью **целых чисел** может быть представлено конечное количество объектов, являющихся дискретными по своей природе. Значения **целочисленных типов** всегда представляются в памяти компьютера абсолютно точно.

Значение **вещественных типов** определяет число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

Значением **символьного типа** char являются символы из некоторого предопределенного множества. Например, множество символов в кодировке ASCII (American Standard Code for Information Interchange).

Значениями **логического типа** может быть одна из предварительно объявленных констант false (ложь) или true (истина).

Тип указателя представляет собой адрес ячейки памяти. Физическое представление адреса существенно зависит от аппаратной архитектуры вычислительной системы.

Указатели

В языках высокого уровня **указатели могут быть типизированными и нетипизированными**. При объявлении **типизированного** указателя определяется и тип данного в памяти, адресуемого этим указателем.

При решении прикладных задач с использованием языков высокого уровня наиболее частые случаи, **когда могут понадобиться указатели**, следующие:

1) при необходимости **представить одну и ту же область памяти**, а следовательно, **одни и те же физические данные как данные разной логической структуры**. В этом случае вводятся два или более указателей, которые содержат адрес одной и той же области памяти, но имеют разный тип. Обращаясь к этой области памяти по тому или иному указателю, можно обрабатывать ее содержимое как данные того или иного типа;

2) **при работе с динамическими структурами данных**. Память под такие структуры выделяется в ходе выполнения программы, стандартные процедуры/функции выделения памяти возвращают адрес выделенной области памяти – указатель на нее. К содержимому динамически выделенной области памяти можно обращаться только через такой указатель.

Нетипизированный указатель служит для представления адреса, по которому содержатся данные неизвестного типа.

Линейные структуры данных

Статические структуры данных: массивы, строки, записи, множества.

Массив – это поименованная совокупность однотипных элементов, упорядоченных по индексам, определяющим положение элемента в массиве.

Строка – это последовательность символов (элементов символьного типа).

Запись – это агрегат, составляющие которого (поля) имеют имя и могут быть различного типа.

Множество – совокупность каких-либо однородных элементов, объединенных общим признаком и представляемых как единое целое.

Таблицы

Таблица представляет собой одномерный массив (вектор), элементами которого являются записи. Основное применение – в базах данных.

Характерная особенность: **доступ к элементам** таблицы производится не по индексу, а **по ключу**, т.е. по значению одного (или нескольких) из полей записи.

Ключ таблицы (основной, первичный) – поле (набор полей), значение которого может быть использовано для однозначной идентификации каждой записи таблицы. Ключ таблицы может быть составным – образовываться не одним, а несколькими полями данной таблицы.

Вторичный ключ – поле (набор полей) таблицы с несколькими полями, не обеспечивающее (в отличие от первичного ключа) однозначной идентификации записей таблицы. В этот ключ могут входить все поля таблицы за исключением полей, составляющих первичный ключ.

Основной операцией при работе с таблицами является **операция доступа к записи по ключу**.

Линейный однонаправленный список

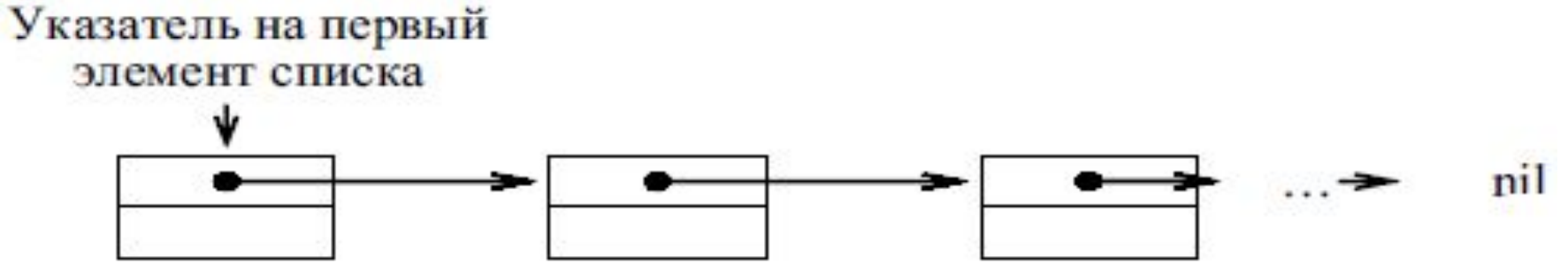
Список – это динамическая структура данных, представляющая собой логически связанную последовательность элементов.

Линейный однонаправленный список

В этом списке любой элемент имеет один указатель, который хранит адрес следующего элемента (указывает на следующий элемент в списке) или является пустым указателем (у последнего элемента).

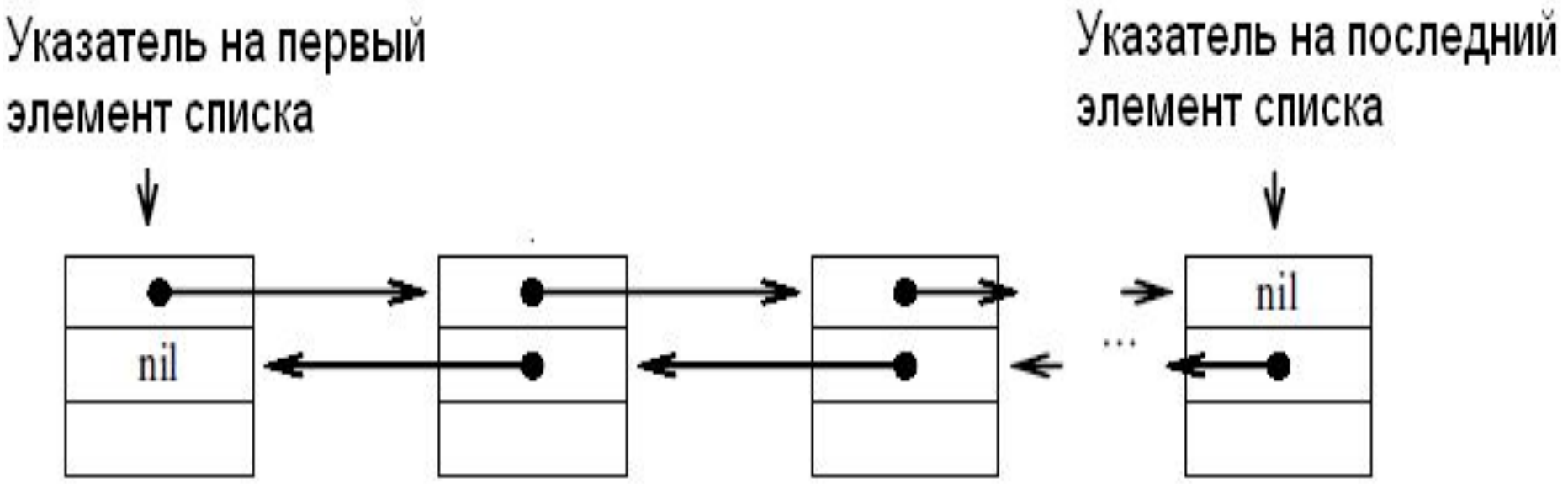
Основные операции, осуществляемые с линейным однонаправленным списком:

– вставка элемента;
– просмотр элементов;
– поиск заданного элемента;
– удаление элемента.



Линейный двунаправленный список

В этом линейном списке любой элемент имеет **два указателя**, один из которых указывает на следующий элемент в списке или является пустым указателем у последнего элемента, а второй – на предыдущий элемент в списке или является пустым указателем у первого элемента.

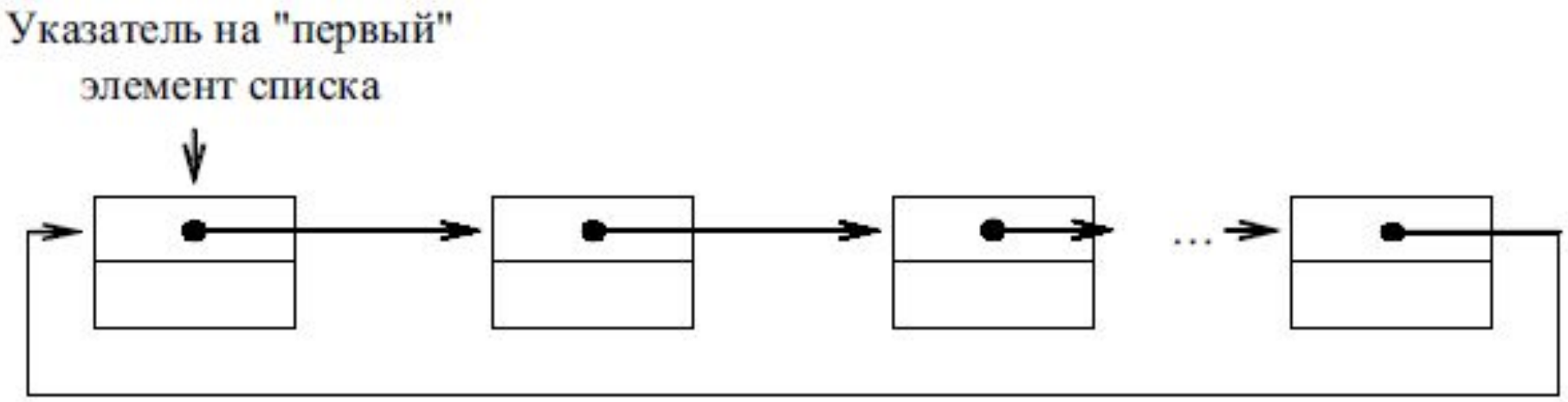


Циклический список

Основное отличие **циклического списка** состоит в том, что в этом списке нет элементов, содержащих пустые указатели, и, следовательно, нельзя выделить крайние элементы. Таким образом, все элементы являются «средними, промежуточными».

Циклический однонаправленный список похож на линейный однонаправленный список, но его последний элемент содержит указатель, связывающий его с первым элементом.

Для полного обхода такого списка достаточно иметь указатель на произвольный элемент, а не обязательно на первый, как в линейном однонаправленном списке.



Циклический список

Основные операции, осуществляемые **с циклическим** однонаправленным списком:

– вставка элемента;
– просмотр;
– поиск;
– удаление элемента.

При **вставке** в качестве входных параметров передаются данные для заполнения создаваемого элемента, указатель на начало списка и указатель на текущий элемент в списке, после которого осуществляется вставка.

Операция поиска элемента в списке заключается в последовательном просмотре всех элементов списка до тех пор, пока текущий элемент не будет содержать заданное значение или пока не достигнут «первый» элемент списка.

Операция удаления элемента циклического однонаправленного списка осуществляет удаление элемента, следующего за тем, на который установлен указатель текущего элемента.

Циклический двунаправленный список

В этом циклическом списке любой элемент имеет **два указателя**, один из которых указывает на следующий элемент в списке, а второй указывает на предыдущий элемент.

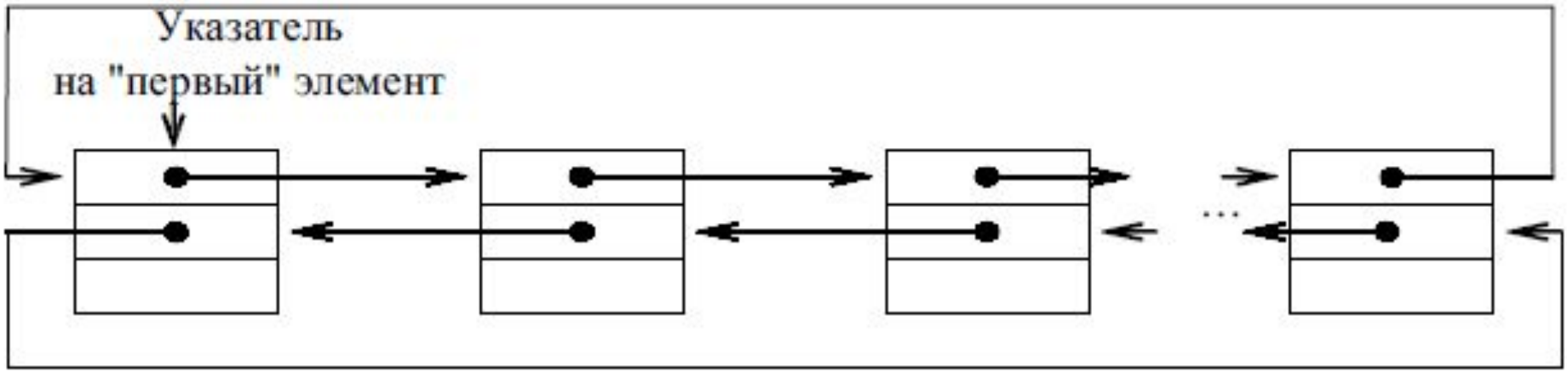
Основные операции, осуществляемые с **циклическим** двунаправленным списком:

– вставка элемента;

– просмотр;

– поиск;

– удаление элемента.



Разреженные матрицы

Разреженная матрица – двумерный массив, большинство элементов которого равны между собой, так что хранить в памяти достаточно лишь небольшое число значений, отличных от основного (***фонового***) значения остальных элементов.

Различают **два типа разреженных матриц**:

1) матрицы, в которых **местоположения** элементов со значениями, отличными от фонового, **могут быть математически описаны**;

2) матрицы **со случайным расположением** элементов.

Матрицы с математическим описанием местоположения элементов

Элементы, значения которых являются **фоновыми**, называют **нулевыми**, а элементы, значения которых **отличны от фонового**, называют **ненулевыми**. Но необходимо помнить, что фоновое значение не всегда равно нулю.

Ненулевые значения хранятся, как правило, в одномерном массиве (векторе), а связь между местоположением в разреженной матрице и в новом, одномерном массиве, описывается математически с помощью формулы, преобразующей индексы матрицы в индексы вектора.

Разреженные матрицы с математическим описанием местоположения элементов

Для работы с разреженной матрицей разрабатываются функции:

1) **для преобразования индексов** матрицы в индекс вектора;

2) **для получения значения элемента** матрицы из ее упакованного представления по двум индексам (строка, столбец);

3) **для записи значения элемента** матрицы в ее упакованное представление по двум индексам.

Пример. Пусть имеется двумерная разреженная матрица, в которой все ненулевые элементы расположены в шахматном порядке, начиная со второго элемента. Для такой матрицы формула вычисления индекса элемента в линейном представлении будет следующей:

$$I = ((x - 1)\cdot Xm + y)/2,$$

где I – индекс в линейном представлении; x, y – индексы соответственно строки и столбца в двумерном представлении; Xm – количество элементов в строке исходной матрицы.

Разреженные матрицы со случайным расположением элементов

К данному типу относятся матрицы, у которых местоположение элементов со значениями, отличными от фонового, не может быть математически описано, т. е. в их расположении нет какой-либо закономерности.

Пример

0	0	6	0	9	0	0
2	0	0	7	8	0	4
10	0	0	0	0	0	0
0	0	12	0	0	0	0
0	0	0	3	0	0	5

Последовательное представление матрицы со случайным расположением элементов

Один из основных способов хранения подобных разреженных матриц заключается в **запоминании ненулевых элементов в одномерном массиве записей** с идентификацией каждого элемента массива индексами строки и столбца матрицы (т.е. с полями Row, Column, Value).

Такой способ хранения называется **последовательным представлением разреженных матриц.**

Доступ к элементу матрицы **A** с индексами i и j выполняется выборкой индекса i из поля Row, индекса j из поля Column и значения элемента из поля Value.

При таком способе представления **элементы матрицы обязательно запоминаются в порядке возрастания номеров строк для ускорения поиска** (таблица изображена на следующем слайде).

Недостатки последовательного представления разреженных матриц

Включение и исключение новых элементов матрицы вызывает необходимость перемещения большого числа существующих элементов.

Последовательное представление матрицы со случайным расположением элементов

Пример. Пусть имеется матрица A размерности 5×7, в которой из 35 элементов только 10 отличны от нуля:

0	0	6	0	9	0	0
2	0	0	7	8	0	4
10	0	0	0	0	0	0
0	0	12	0	0	0	0
0	0	0	3	0	0	5

Row	*Colum*	*Value*
1	3	6
1	5	9
2	1	2
2	4	7
2	5	8
2	7	4
3	1	10
4	3	12
5	4	3
5	7	5

Метод связанных структур

Метод связанных структур переводит статическую структуру матрицы в **динамическую**. На рисунке представлена возможная структура, реализованная в виде циклических списков.

Циклический список представляет отдельную строку или столбец. **Список столбца** может содержать общие элементы с одним или более **списком строки**. Все списки строк и столбцов имеют головные элементы. Головной элемент каждого списка строки содержит ноль в поле Column; столбца – имеет ноль в поле Row. Строка или столбец, содержащие только нулевые элементы, представлены головными вершинами, у которых поле Left или Up указывает само на себя.

Row	*Colum*	*Value*
1	3	6
1	5	9
2	1	2
2	4	7
2	5	8
2	7	4
3	1	10
4	3	12
5	4	3
5	7	5