Структуры и алгоритмы обработки данных (СиАОД)

Курс ведут:

Лобанов Александр Анатольевич

Шмелева Дарья Викторовна

Рекомендуемая литература

- Материал курса опирается на знания студентов, полученные ранее в курсах:
- «Процедурное программирование»,
- «Математическая логика и теория алгоритмов»,
- «Дискретная математика».

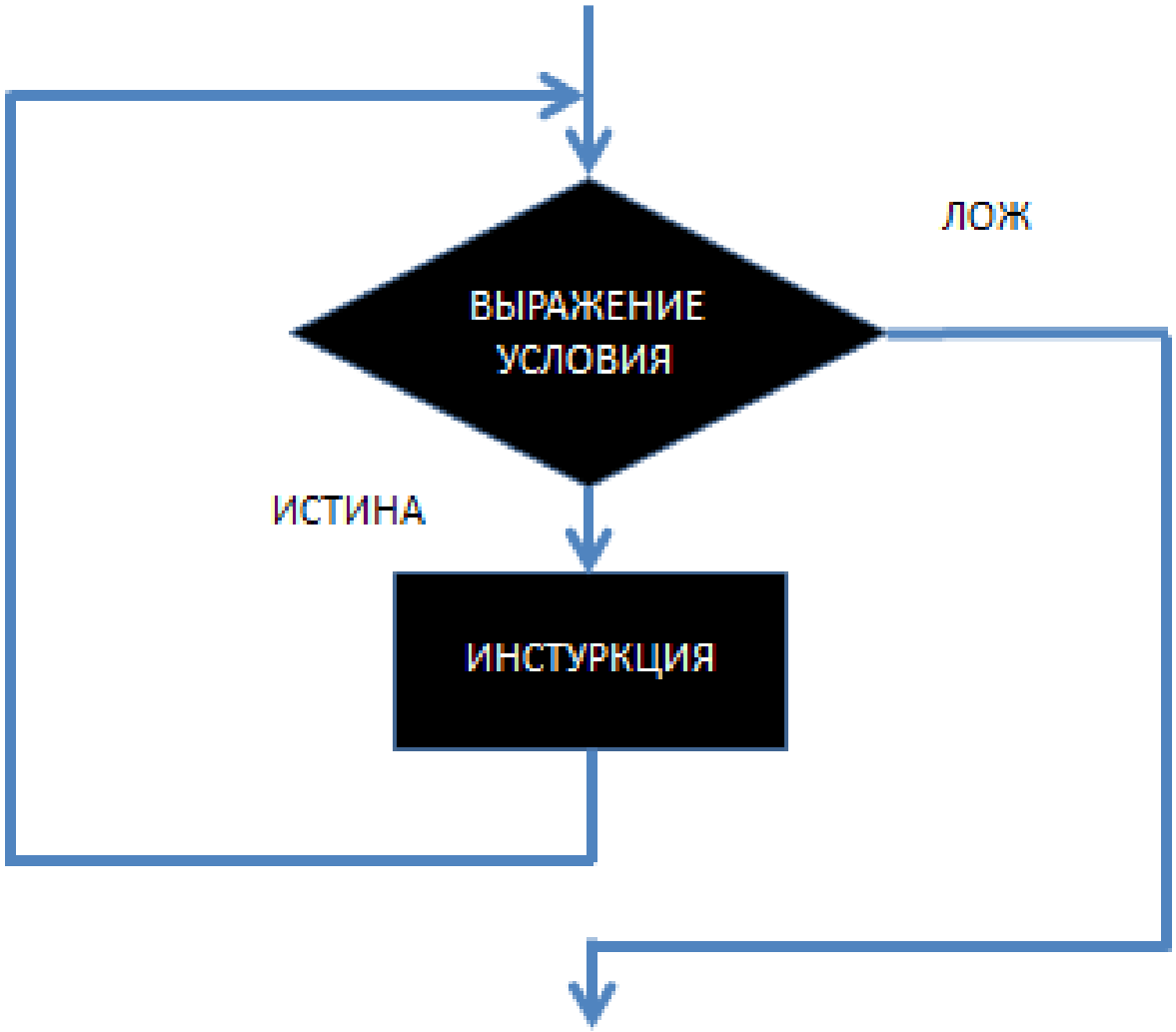
Рекомендуемая литература

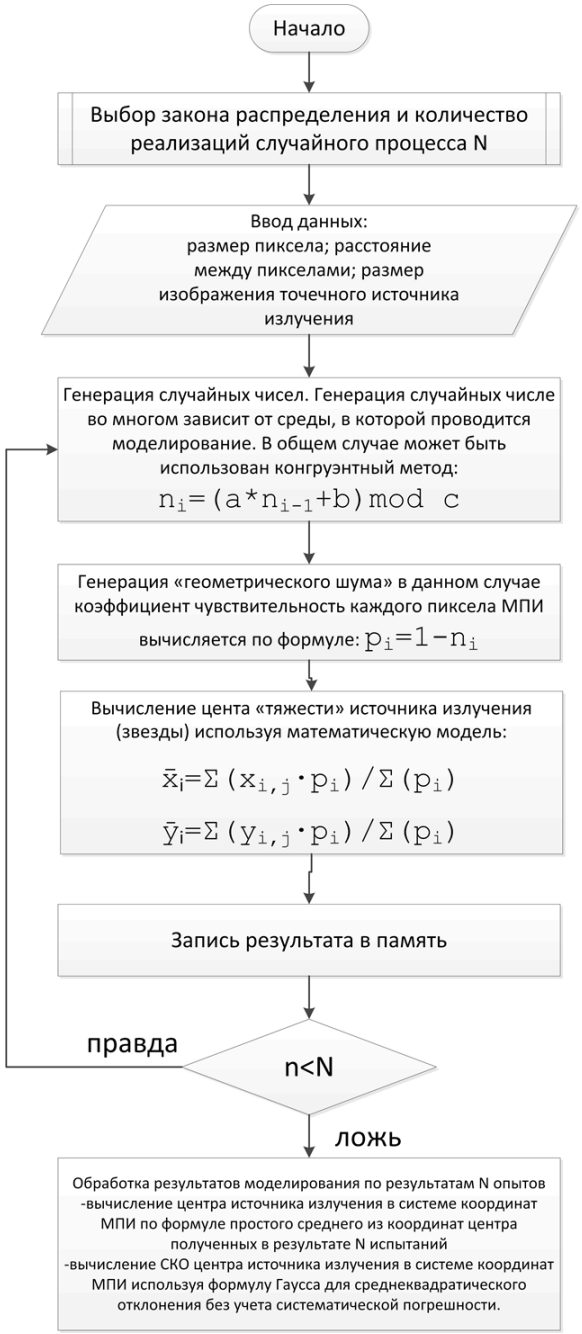
1. Ключарев А. А., Матьяш В. А., Щекин С. В. Структуры и алгоритмы обработки данных: Учеб. пособие/СПбГУАП. СПб., 2003. 172 с.
2. Пышкин Е.В. Структуры данных и алгоритмы: реализация на C/C++. - СПб.: ФТК СПБГПУ, 2009.-200 с.

Алгоритм

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

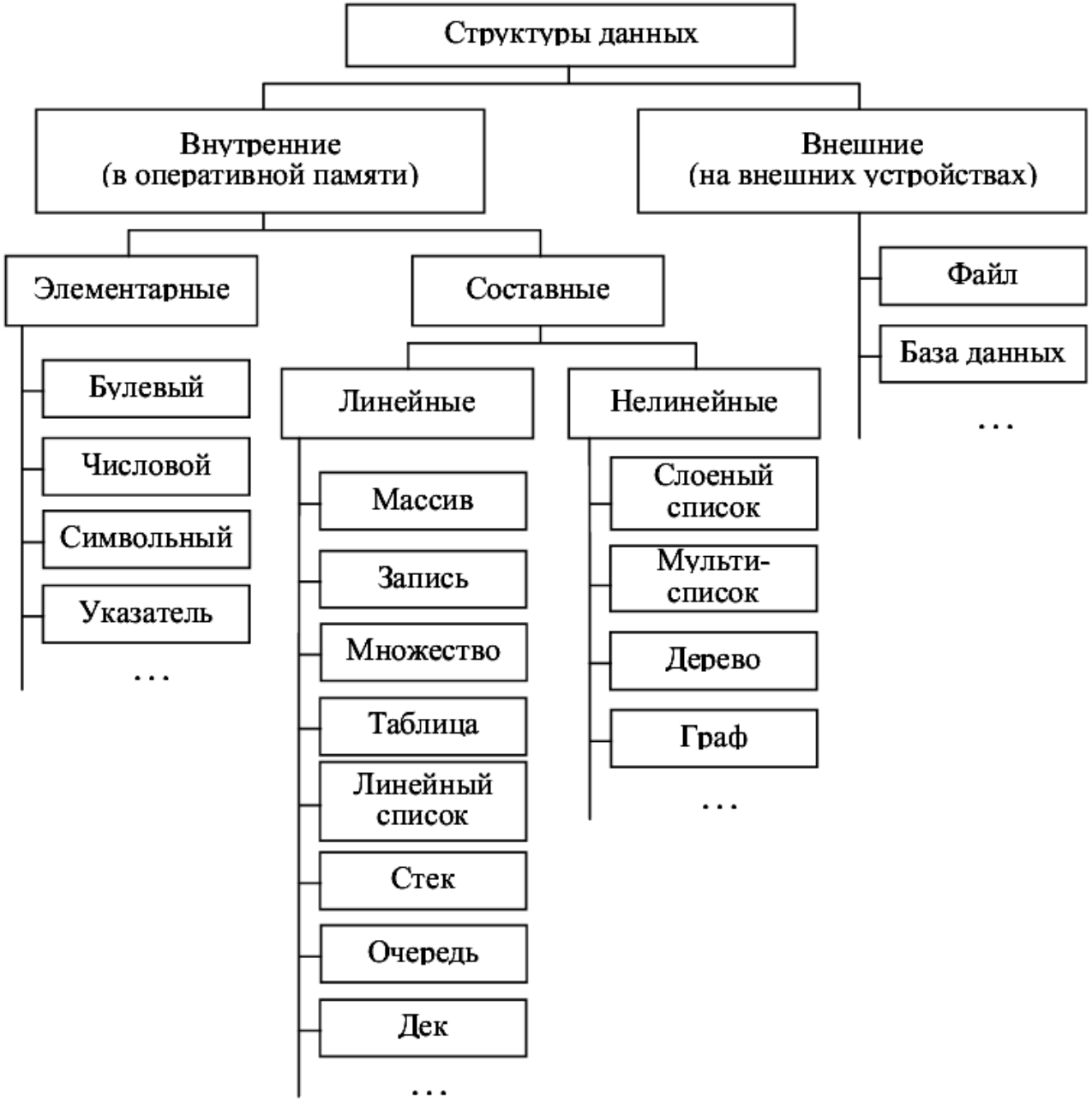
Алгоритм цикла while




Начало
Выбор закона распределения и количество реализаций случайного процесса N
Ввод данных:
размер пиксела; расстояние между пикселами; размер изображения точечного источника излучения
Генерация случайных чисел. Генерация случайных числе во многом зависит от среды, в которой проводится моделирование. В общем случае может быть использован конгруэнтный метод:
$n_i=(a*n_{i-1}+b) \bmod c$
Генерация «геометрического шума» в данном случае коэффициент чувствительность каждого пиксела МПИ вычисляется по формуле: $p_i=1-n_i$
Вычисление цента «тяжести» источника излучения (звезды) используя математическую модель:
$\bar{x}_i=\Sigma(x_{i,j}\cdot p_i)/\Sigma(p_i)$
$\bar{y}_i=\Sigma(y_{i,j}\cdot p_i)/\Sigma(p_i)$
Запись результата в память
правда
n<N
ложь
Обработка результатов моделирования по результатам N опытов
-вычисление центра источника излучения в системе координат МПИ по формуле простого среднего из координат центра полученных в результате N испытаний
-вычисление СКО центра источника излучения в системе координат МПИ используя формулу Гаусса для среднеквадратического отклонения без учета систематической погрешности.

Структура данных

- множество элементов данных и множество связей между ними. Такое определение охватывает все возможные подходы к структуризации данных, но в каждой конкретной задаче используются те или иные его аспекты. Поэтому вводится дополнительная классификация структур данных, которая соответствует различным аспектам их рассмотрения. Прежде чем приступать к изучению конкретных структур данных приведем их общую классификацию по нескольким признакам.


Структуры данных
Внутренние
(в оперативной памяти)
Внешние
(на внешних устройствах)
Элементарные
Составные
Файл
База данных
...
Булевый
Числовой
Символьный
Указатель
...
Линейные
Нелинейные
Массив
Запись
Множество
Таблица
Линейный
список
Стек
Очередь
Дек
...
Слоеный
список
Мульти-
список
Дерево
Граф
...

Изменчивость структуры данных

- изменение числа элементов и/или связей между составными частями структуры.
- При этом мы не говорим об изменении самих данных!
- Различают статические и динамические структуры данных.

Классификация структуры данных

- В зависимости от характера взаимного расположения элементов в памяти структуры данных можно разделить на структуры со смежным (или последовательным) распределением элементов в памяти и структуры со связным распределением элементов в памяти.
- По признаку изменчивости количества элементов и связей между ними различают структуры статические (векторы, массивы, записи) полустатические (стеки, очереди, деки, строки) и динамические (списки, деревья, графы).

- Над любыми структурами данных могут выполняться четыре общие операции: создание, уничтожение, доступ, обновление.

Структуры данных по степени изменчивости

Статические

1. Вектор
2. Массив
3. Множество
4. Запись
5. Таблица

Полустатические

1. Стек
2. Очередь
3. Дек
4. Строка

Динамические

1. Линейные связные списки
2. Разветвленные связные списки
3. Графы
4. Деревья

Массивы

- Логически массив представляет собой структуру данных, которая характеризуется фиксированным числом элементов одного типа, каждый из которых имеет уникальный набор значений индексов. Количество индексов у каждого элемента массива определяет его размерность. Обращение к элементу массива осуществляется по имени массива и значениям индексов для данного элемента.

Запись

- Логически запись - конечное упорядоченное множество полей, характеризующихся различным типом данных (в языке С записи называются структурами).

ПОЛУСТАТИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

1. Имеют переменное число элементов;
2. Изменение длины структуры происходит, как правило, в определённых пределах, не превышая какого – то максимального числа элементов;
3. Логически полустатические структуры представляют собой последовательность данных, связанных отношениями линейного списка.

Стеки (stack-стопка)

- Логически стек представляет собой последовательность элементов с переменной длиной; включение и исключение элементов из последовательности происходит при этом только с одной стороны, называемой вершиной стека. Другие названия стека – магазин (оружейный).
- Организован по принципу LIFO (Last In - First Out).

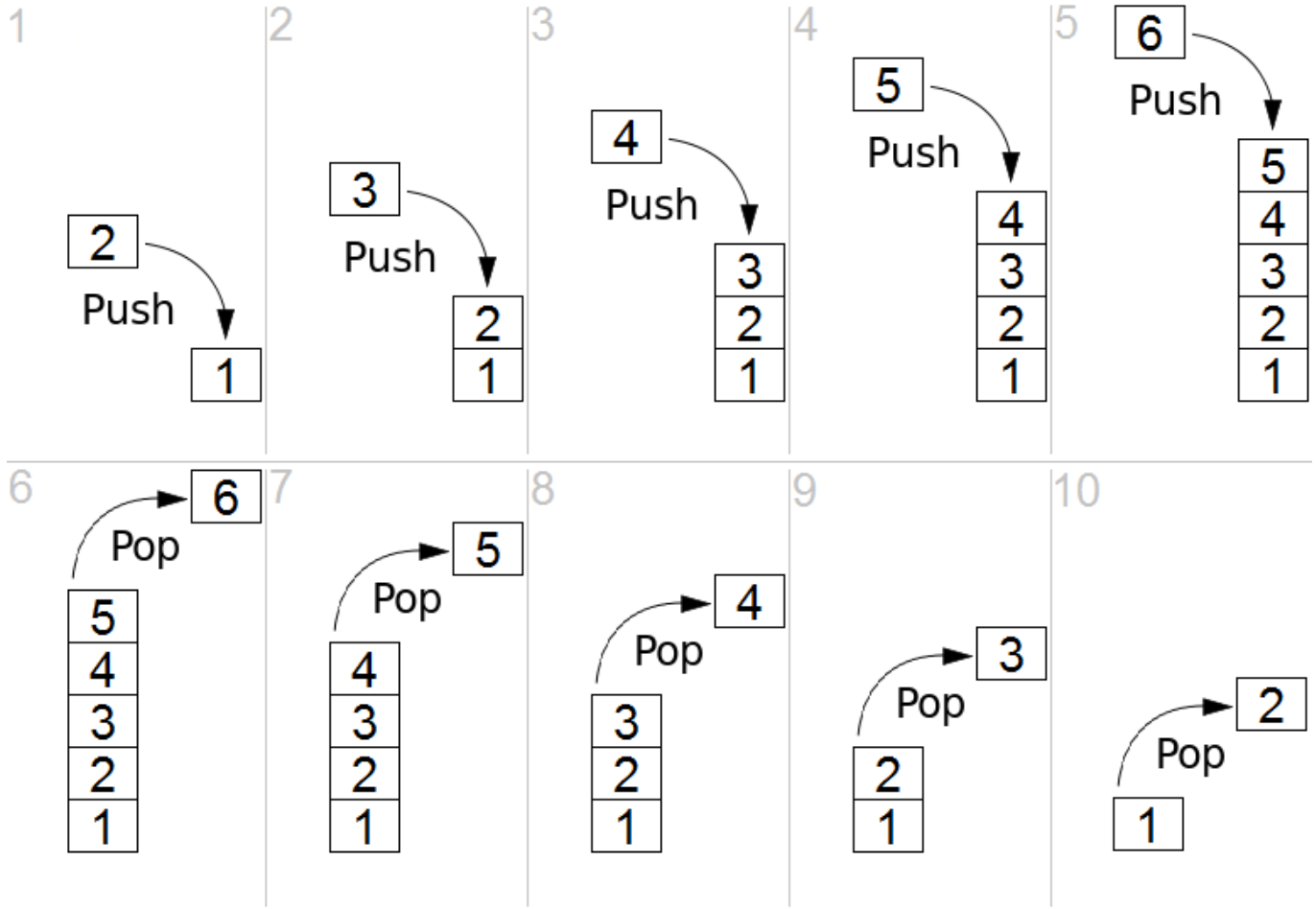
Организация стека

Операции со стеком

- Добавление элемента (иначе проталкивание, *push*);
- Удаление (исключение) элемента (*pop*);
- Чтение головного элемента (*peek*);
- Определение текущего числа элементов в стеке;
- Очистка стека.

- При проталкивании (push) добавляется новый элемент, указывающий на элемент, бывший до этого головой. Новый элемент теперь становится головным.
- При удалении элемента (pop) убирается первый, а головным становится тот, на который был указатель у этого объекта (следующий элемент). При этом значение убранного элемента возвращается.

Операции вталкивания и выталкивания данных



Реализация стека на C

```
struct stack
{
    char *data;
    struct stack *next;
};
```

Размер стека определим в виде константы:

```
#define MAXN 1000
```

```
typedef struct
{
    int  sp;
    int val [MAXN];
}  stack;
```

Можем создавать стеки, просто написав, например,

```
stack a,  b;
```

Операции со стеком на С

```
void push( STACK *ps, int x ) // Добавление в стек нового элемента
{
    if ( ps->size == STACKSIZE ) // Не переполнен ли стек?
    {
        fputs( "Error: stack overflow\n", stderr );
        abort();
    }
    else
    {
        ps->items[ps->size++] = x;
    }
}

int pop( STACK *ps ) // Удаление из стека
{
    if ( ps->size == 0 ) // Не опустел ли стек?
    {
        fputs( "Error: stack underflow\n", stderr );
        abort();
    }
    else
    {
        return ps->items[--ps->size];
    }
}
```

Область применения стека

- обход структур данных, например, дерево или граф;
- отслеживание точек возврата из подпрограмм;
- сохранение незаконченной работы при переключении на другую задачу;

- арифметические сопроцессоры и т.п.

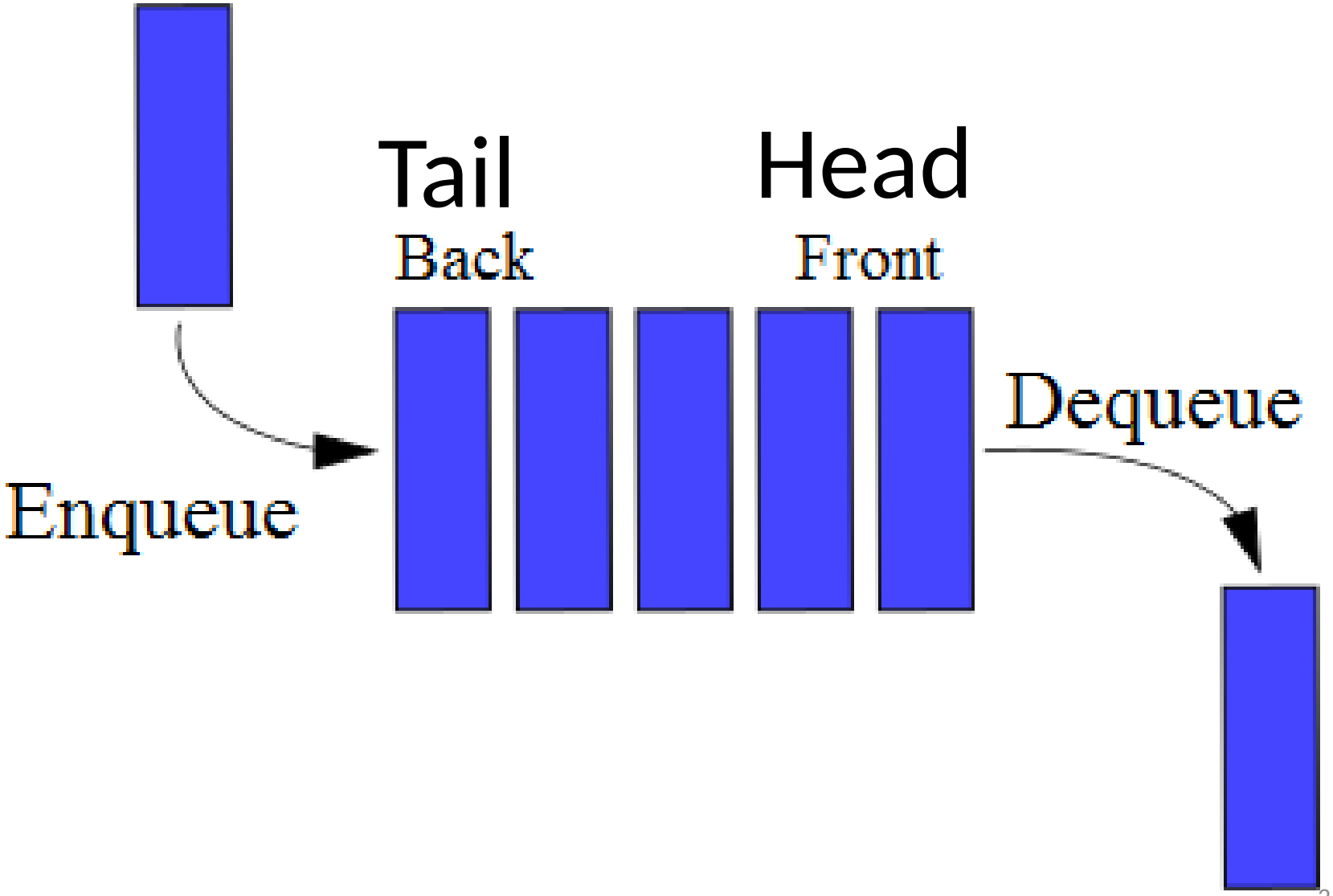
Очередь

- Структура данных с последовательностью доступа к элементам «первый пришёл — первый вышел» (FIFO, First In — First Out).

Организация очереди

- Добавление элемента (принято обозначать словом ***enqueue*** — поставить в очередь) возможно лишь в конец очереди;
- Выборка — только из начала очереди (что принято называть словом ***dequeue*** — убрать из очереди), при этом выбранный элемент из очереди удаляется.

Организация очереди



Реализация стека на С

Очередь реализуем в виде структуры:

```
typedef struct
{
   int qh, qt;
   int val[MAXN];
} queue;
```

Для создания очереди пишем queue a, b;

Операции с очередью

- Для очереди существует две основные операции:
- добавить элемент в хвост (tail) очереди (enq);
- извлечь элемент из головы (head) очереди (deq).

```
void enq(queue  *q,   int   x)
{
    q->val[(q->qt++)%MAXN]   =  x;
}


int deq(queue  *q)
{
    return q->val[(q->qh++)%.MAXN] ;
}
```

Представление очереди

- Физически очередь представляется в памяти машины аналогично стеку, т.е. в виде вектора, размер которого определяет максимальный размер очереди.
- Дескриптор очереди в дополнение к обычным для дескриптора вектора параметрам должен содержать два указателя:

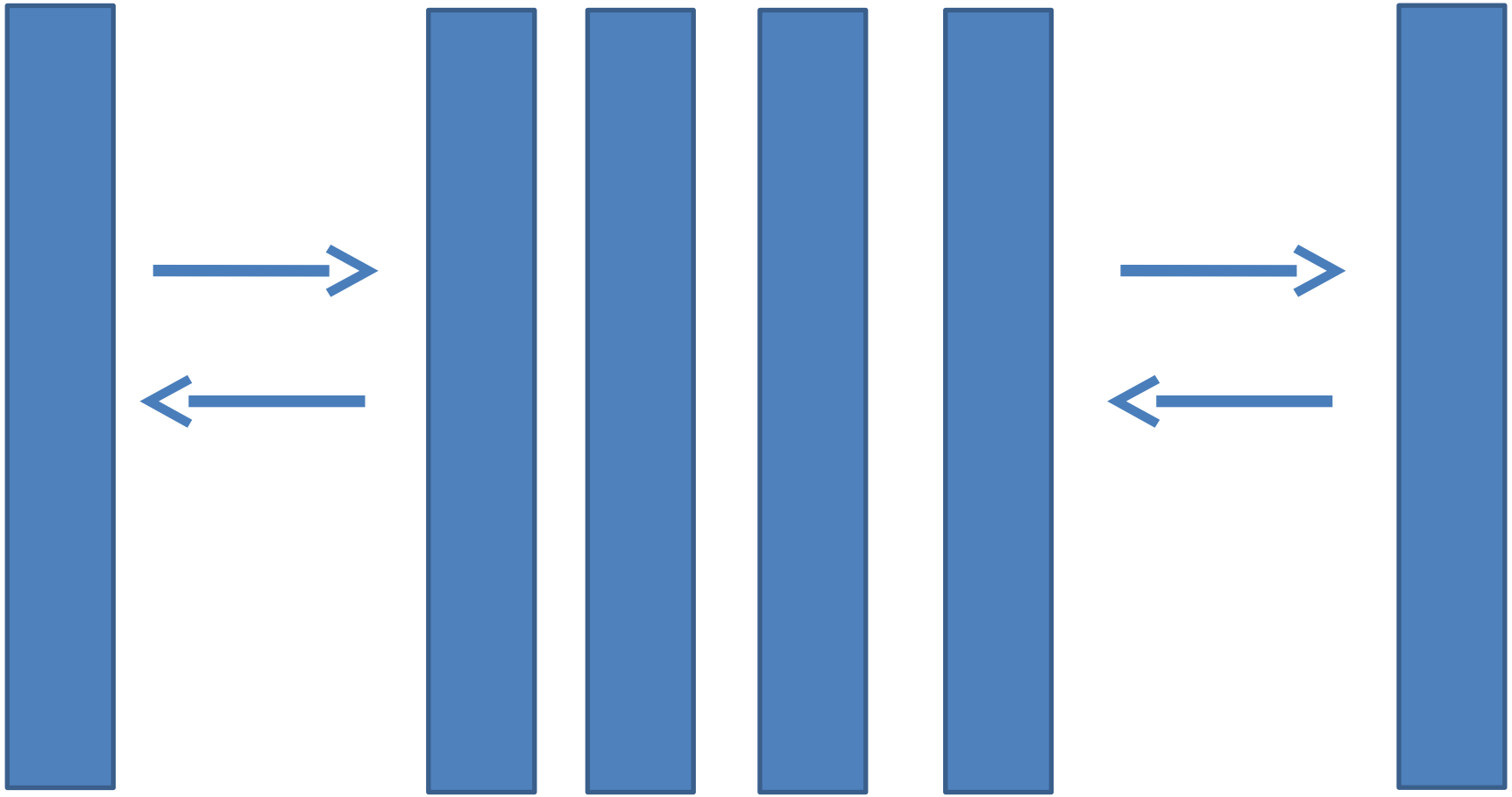
  на начало очереди (на первый элемент в очереди) и

  на конец очереди (первый свободный элемент в очереди).
- При включении элемента в очередь элемент записывается по адресу, определяемому указателем на конец, после чего этот указатель модифицируется (увеличивается на размер слота).
- При исключении элемента из очереди читается элемент, адресуемый указателем на начало, после чего этот указатель также модифицируется (уменьшается на размер слота).

Двухсвязная очередь (Дек)

- Дек (deque) -это сокращенная фраза «double-ended-queue»;
- структура данных, в которой элементы можно добавлять и удалять как в начало, так и в конец, то есть принципу является одновременно FIFO и LIFO.

Организация Дека



Операции с деком

- PushBack — добавление в конец очереди.
- PushFront — добавление в начало очереди.
- PopBack — выборка с конца очереди.
- PopFront — выборка с начала очереди.
- Проверка наличия элементов.
- Очистка.

Реализация дека на С

- Для дека можно использовать ту же структуру ,что и для очереди, но она будет содержать 4-е функции (см. предыдущий слайд);

```
typedef struct
{
  int dh,  dt;
  int val[MAXN];
} deque;
```

C++ содержит библиотеку STL (Standart template library, стандартная библиотека шаблонов).
В библиотеку STL входят шаблоны для реализации стеков, деков, очередей для произвольных типов данных.
Например, чтобы создать стек целых чисел,
необходимо использовать тип stack<int>,
для создания стека строк – stack<string>.

Описание шаблонов stack, queue, deque содержится в одноименных заголовочных файлах, то есть для использования этих структур необходимо подключить один из трех заголовочных файлов:

```
#include<stack>
#include<queue>
#include<deque>
```

После этого можно создавать объекты на основе данных шаблонов:

```
stack<int> S;      // Стек объектов типа int
queue<double> Q;   // Очередь объектов типа double
deque<string> D;   // Дек объектов типа string
```

После этого с объявленными объектами можно выполнять операции:

```
S.push(3);      // Положить в стек S число 3
Q.size();       // Узнать размер очереди Q
D.pop_front();  // Удалить элемент из начала дека D
```