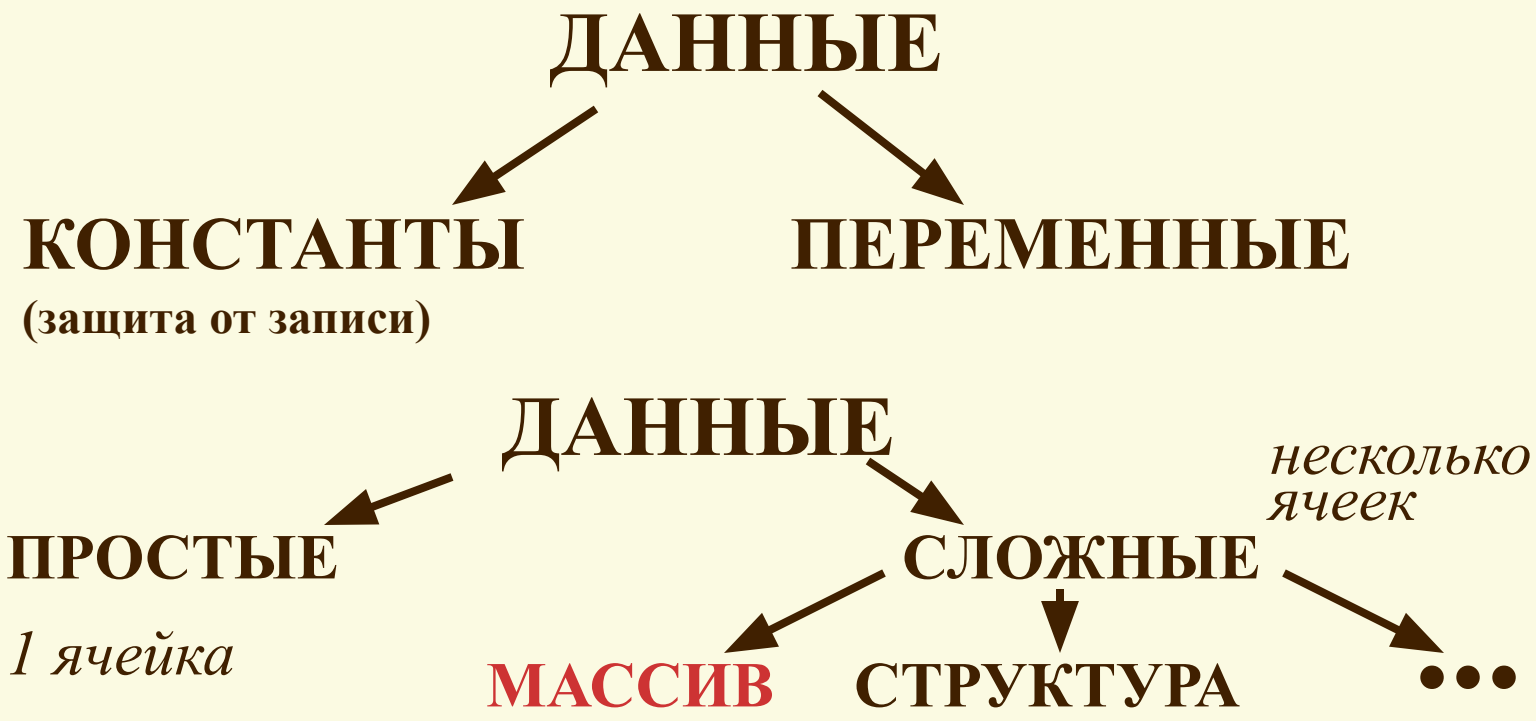
МАССИВЫ

- Определение
- Описание
- Обращение к элементам массива
- Связь массивов с указателями
- Примеры программ

КЛАССИФИКАЦИЯ ДАННЫХ ПО СТРУКТУРЕ



ОПРЕДЕЛЕНИЕ

Массив - это сложное данное, состоящее из конечного числа упорядоченных компонент, имеющих одно имя, одинаковый тип и расположенных в последовательных ячейках памяти компьютера.

Упорядоченность компонент массива: компоненты пронумерованы.

Доступ к элементу массива - по его номерам (индексам).

Размерность массива - количество индексов у его элементов.

Размер - количество значений каждого индекса.

МАССИВЫ В ПРОГРАММЕ

ОПИСАНИЕ

СИ

размеры - только константы

тип имя[размер_1]…[размер_N]

ОБРАЩЕНИЕ К ЭЛЕМЕНТУ МАССИВА

СИ

имя[индекс_1]…[индекс_N]

индекс_i - целое выражение, индекс_i = 0,1,…,N-1

В Си элементы массивов нумеруются, начиная с нуля.

МАССИВЫ В СИ-ПРОГРАММЕ

Примеры. **float a[20];**

a[0], a[1],...,a[19].

int b[3][5];

b[0][0] b[0][1] ... b[0][4]
b[1][0] b[1][1] ... b[1][4]
b[2][0] b[2][1] ... b[2][4]

Первый индекс - номер строки, второй - столбца

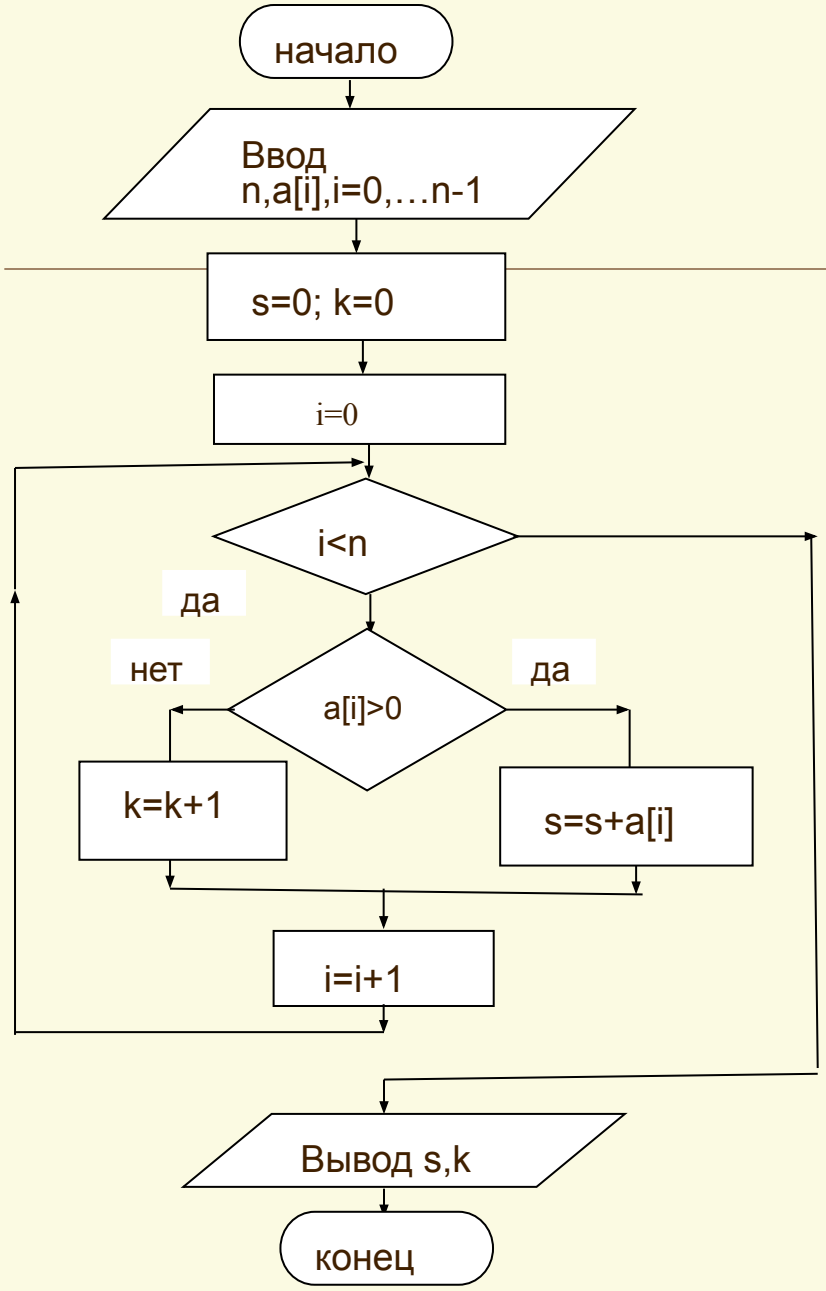
В памяти компьютера элементы массива расположены по строкам (чаще меняется последний индекс)

Примеры программ с массивами

Дан массив **a** из **n** элементов, **n≤20**. Вычислить сумму положительных и количество неположительных элементов массива.

Состав данных

Имя	Смысл	Тип	Структура
Исходные данные			
n	число элементов массива	целый	простая переменная
a	заданный массив	вещественный	одномерный массив из 20 элементов
Выходные данные			
s	сумма положительных элементов массива	вещественный	простая переменная
k	количество неположительных элементов	целый	простая переменная
Промежуточные данные			
i	счетчик элементов массива	целый	простая переменная



Блок-схема алгоритма

```
#include <iostream.h>
void main()
{float a[20],s; int k,i,n;
cout<<"Vvedite n\n";
cin>>n;
 cout<<"Vvedite massiv
iz"<<n<<"elementov\n";
 /* Далее цикл для поэлементного ввод
массива*/
 for (i=0; i<n; i++)
  cin>>a[i];
/*Далее алгоритм по блок-схеме*/
 s=0; k=0;
 for (i=0; i<n; i++)
  if (a[i]>0)
   s=s+a[i];
  else
   k=k+1;
 cout<<" s=  “<<s;
 cout<<”  “<<”k=“k<<”\n”;
 }
```

Программа

Инициализация массивов при описании в Си

Инициализация - задание начальных значений.

Одномерные массивы

char a[6]={'A', 'B', 'C', 'D'};

0	1	2	3	4	5
A	B	C	D	н/о	н/о

если a - локальная переменная

char a[ ]={'A', 'B', 'C', 'D'};

0	1	2	3
A	B	C	D

Размер массива определяется количеством инициализирующих значений

Локальные и глобальные данные

ДАННЫЕ

ЛОКАЛЬНЫЕ: описаны в функции (в том числе в main); по умолчанию не инициализируются.

ГЛОБАЛЬНЫЕ: описаны вне функций; при описании обнуляются.

Инициализация массивов при описании в Си

Двумерные массивы

Присваивание перечисленных значений происходит по строкам (в соответствии с расположением массивов в памяти компьютера).

int m[2][3]={0,1,2,5,6,7}; int m[ ][3]={0,1,2,5,6,7};

0	1	2
5	6	7

int m[ ][3]={{0},{1,2}};

0	н/о	н/о
1	2	н/о

Инициализация массивов при описании в Си

Вывод: при объявлении массива количество его элементов должно быть задано или явным указанием константы в квадратных скобках или количеством значений при инициализации.

Исключение: массивы-аргументы функций.

Снятие ограничения: динамические массивы

Указатели в Си

Указатель - это специальное данное, которая содержит адрес другого данного.

Основные операции для работы с указателями:

* - взятие содержимого по адресу (*i - содержимое переменной с адресом i)

& - взятие адреса (&a - адрес переменной a).

Описание имеет вид:

*тип *имя_указателя;*

При описании указателя задается тип значения, на которое он указывает.

Примеры описаний: int *i, j, *pointj;

int v1, *pointv1=&v1, *p=(int*)200;

- int i=1, num=40;

ptr=&i;

ptr=#

ptr=#

val=*ptr;

val=num;

нельзя: p=&(c+2);

p=&’A’;

Указатели в Си

УКАЗАТЕЛИ

ПЕРЕМЕННЫЕ

КОНСТАНТЫ:

- адреса переменных (или именованных констант);
- имена массивов;
- явные константы (например, (int*)200);
- константа NULL (нулевой или несуществующий адрес).

Указатели в Си

ВНИМАНИЕ!

- нельзя брать содержимое от константы без приведения типа; запись *200 является некорректной в отличие от *(int*)200;
- нельзя брать адрес явной константы (например, некорректна запись &200), в Си адрес явной константы считается недоступным;
- нельзя определять адрес выражения.

Указатели в Си

Размер памяти, отводимой под указатель, зависит:

- от разрядности адресной шины;
- от модели памяти.

Указатели в Си

Операции над указателями:

*

сравнения (<, <=, >, >=, ==, !=) - с указателями такого же типа или с NULL;

присваивания - значений указателей того же типа или NULL;

арифметические операции сложения, вычитания (с константой)

инкремента и декремента

Указатели в Си

Результат арифметической операции над указателями зависит не только от значения операндов, но и от типа, с которым связан указатель.

p=p+k*, ⇔ p увеличивается на *k*sizeof (тип)

```
Пример.  int *p; long int *pp;…//MS DOS
         p++;        /*p увеличилось на 2*/
       pp++;       /*pp увеличилось на 4*/
```

Связь массивов с указателями в Си

Одномерные массивы

Имя одномерного массива является *указателем-константой*, равной адресу начала массива, т. е. адресу элемента с индексом 0 (первого элемента).

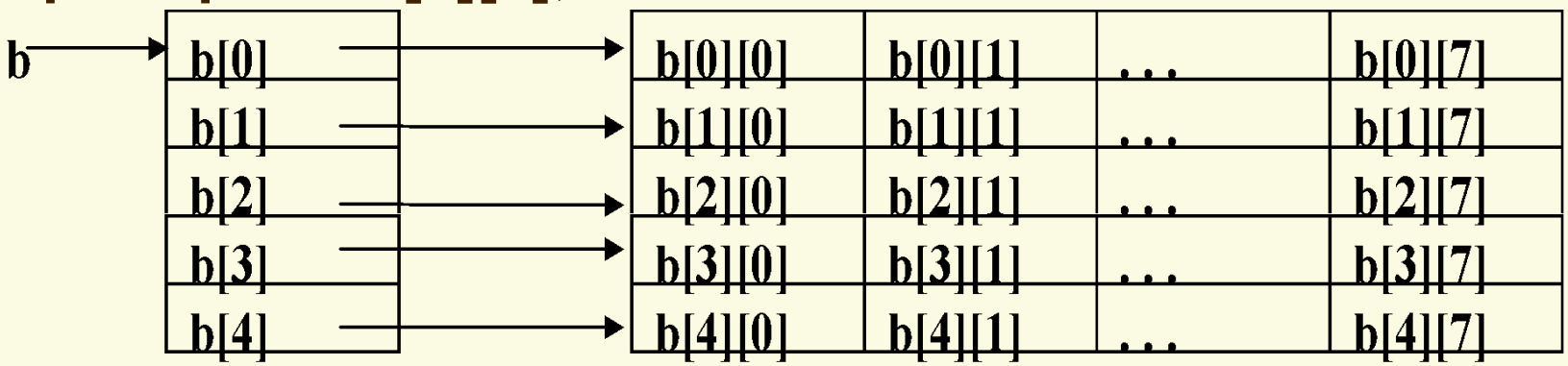
int a[10];
&a[0] эквивалентно **a**,
a[0] эквивалентно ***a**,
&a[i] эквивалентно **a+i (i=0,1,...9)**,
a[i] эквивалентно ***(a+i)**.

Связь массивов с указателями в Си

Двумерные массивы

Имя двумерного массива является *указателем-константой на начало* (элемент с индексом 0) *массива указателей-констант*, i-й элемент этого массива - указатель -константа на начало (элемент с индексом 0) i-й строки двумерного массива.

Пример: int b[5][8];



Связь массивов с указателями в Си

Двумерные массивы

b[i][j] ⇔ *(b[i]+j) ⇔ *(*(b+i)+j);
&b[i][j] ⇔ b[i]+j ⇔ *(b+i)+j

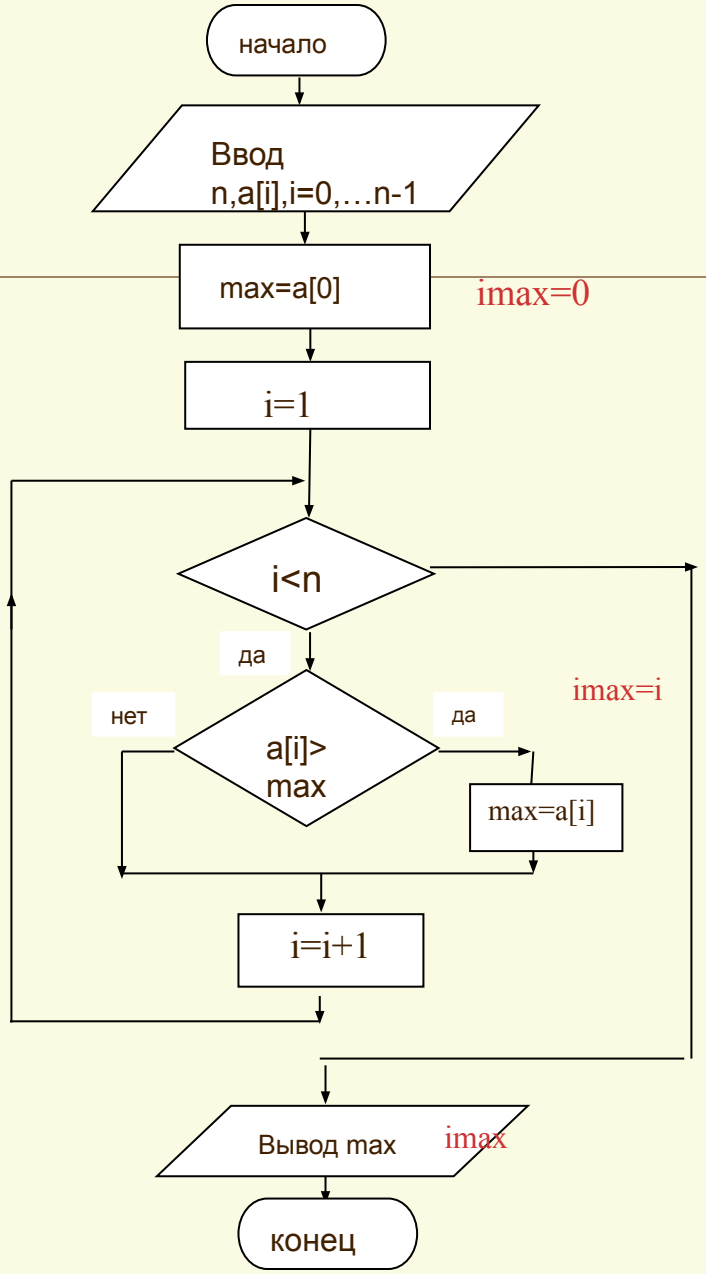
Для любого из трех обозначений элемента двумерного массива программа в кодах получается практически одинаковой по производительности, хотя при использовании арифметики указателей вместо квадратных скобок несколько более короткой.
Хороший стиль программирования предполагает употребление в пределах одной программы одного (из трех) обозначений.

Примеры программ с массивами

Дан массив **a** из **n** элементов, **n≤20**.Найти максимальное значение элементов массива.

Состав данных

Имя	Смысл	Тип	Структура
Исходные данные			
n	число элементов массива	целый	простая переменная
a	заданный массив	вещественный	одномерный массив из 20 элементов
Выходные данные			
max	Максимальное значение элементов массива	вещественный	простая переменная
Промежуточные данные			
i	счетчик элементов массива	целый	простая переменная



Блок-схема алгоритма

```
#include <iostream.h>
void main()
{float a[20],max; int i,n;
cout<<"Vvedite n\n";
cin>>n;
 cout<<"Vvedite massiv iz"<<n<<"elementov\n";
 /* Далее цикл для поэлементного ввод
массива*/
 for (i=0; i<n; i++)
  cin>>a[i];
/*Далее алгоритм по блок-схеме*/
max=a[0]; imax=0;
 for (i=1; i<n; i++)
  if (a[i]>max)
   {max=a[i]; imax=i;
   }
 cout<<" max=  “<<max<<”\n”;
 cout<<" imax=  “<<imax<<”\n”;


 }
```

Программа

Сортировки: введение

Для ускорения поиска информации её необходимо отсортировать

- Файл размером N – некоторая последовательность из N элементов r(1), r(2),…,r(N), каждый из которых называется записью.
- С каждой записью r(i) связывается некоторый ключ k(i).

Сортировки: терминология

N – количество элементов массива

Проход – последовательный просмотр всех элементов массива в прямом или обратном направлении

C – число необходимых сравнений элементов

M - обмен (перестановка) элементов – запись значения i-того элемента массива в k-тый, а k-того – в i-тый с промежуточным сохранением одного из значений в специальной переменной

Эффективность сортировок:

- время, затрачиваемое на программирование;
- время, затрачиваемое на собственно сортировку;
- необходимый объем памяти.

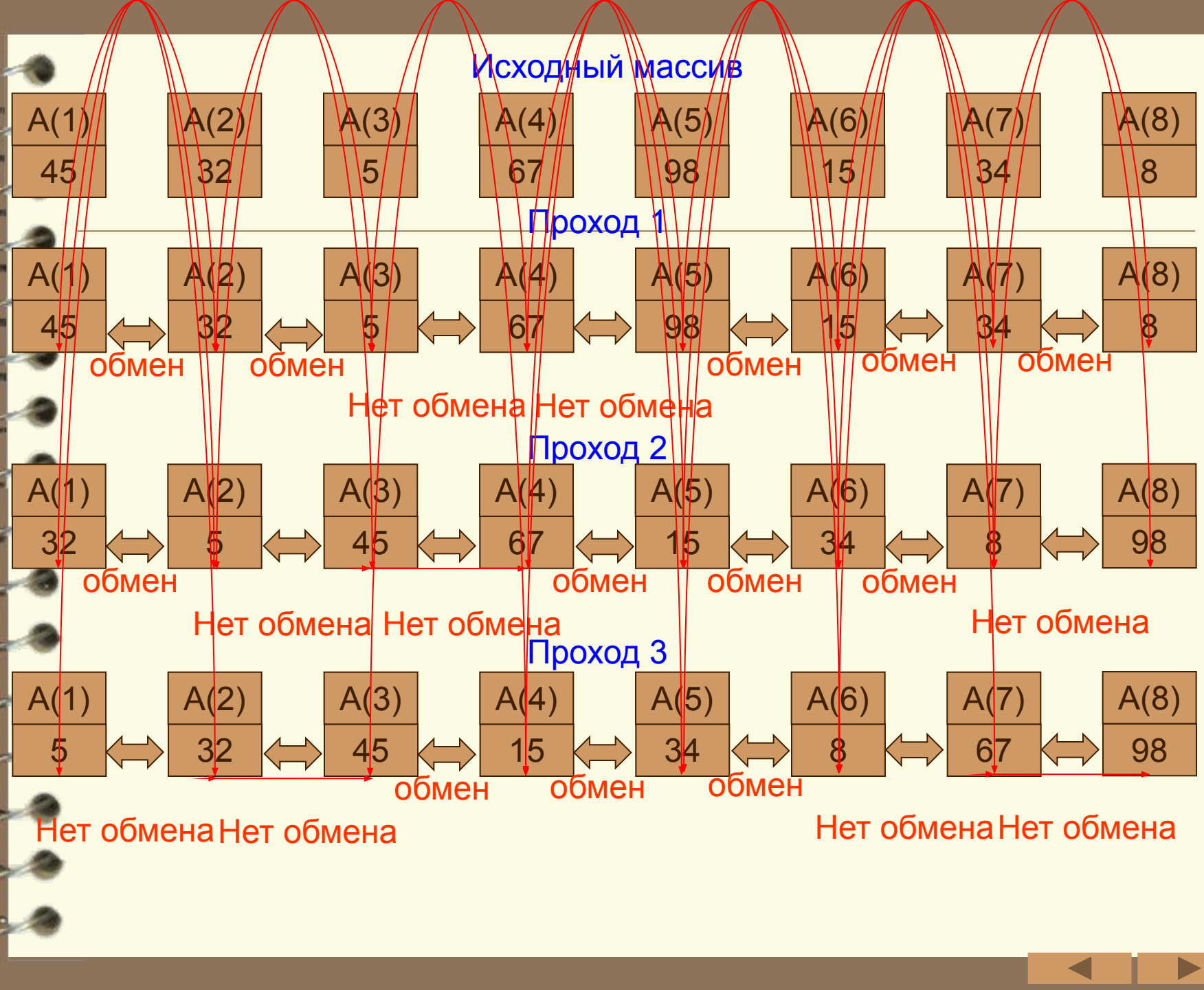
Усовершенствованные методы сортировок:
$C \div N*\log N$

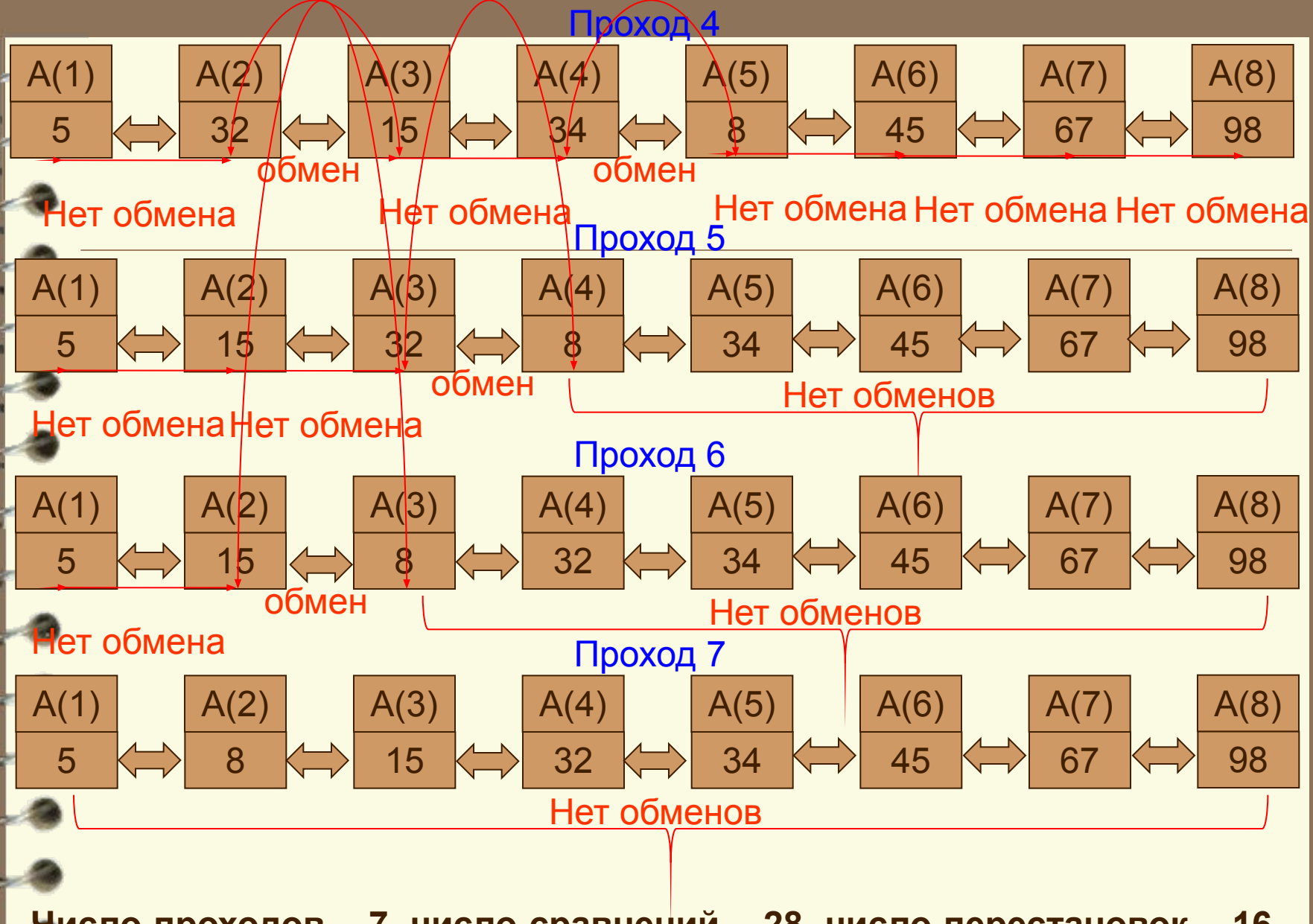
Простые методы: $C \div N^2$

Перестановки элементов – строго на «том же месте»,
т.е. вспомогательный массив не используется

Сортировка прямого обмена

- Алгоритм: на каждом проходе сравниваются элементы A(i) и A(i+1) для i в интервале от 1 до N-1, если A(i) больше A(i+1), то происходит обмен.
- Число проходов N-1
- Число сравнений C=n*(n-1)/2
- Число обменов (среднее) M=C

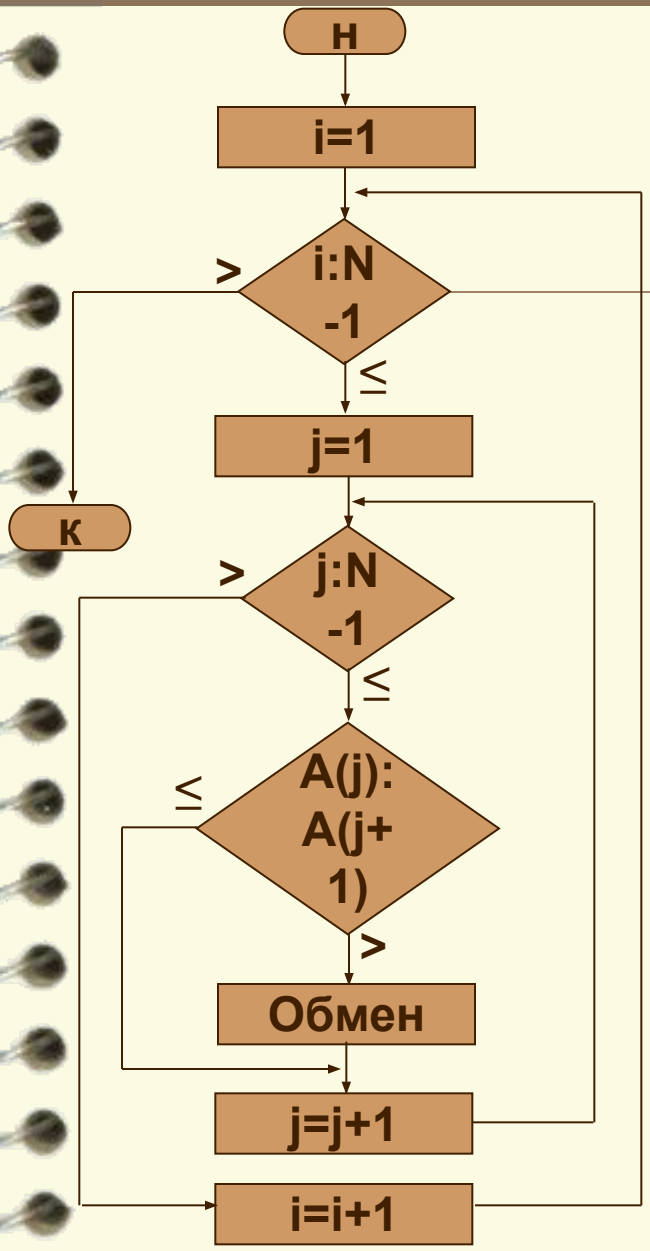

Исходный массив
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
45 32 5 67 98 15 34 8
Проход 1
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
45 32 5 67 98 15 34 8
обмен обмен Нет обмена Нет обмена обмен обмен обмен
Проход 2
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
32 5 45 67 15 34 8 98
обмен Нет обмена Нет обмена обмен обмен обмен Нет обмена
Проход 3
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
5 32 45 15 34 8 67 98
Нет обмена Нет обмена обмен обмен обмен Нет обмена Нет обмена



Число проходов – 7, число сравнений – 28, число перестановок – 16. Если рассматривать массив как вертикальный, то легкие элементы постепенно «всплывают» наверх – «пузырек».

Блок-схема

Ввод и вывод массива не показаны



Внешний цикл реализует N-1 проход, i – номер прохода

Внутренний цикл реализует N-1 сравнение элементов внутри прохода

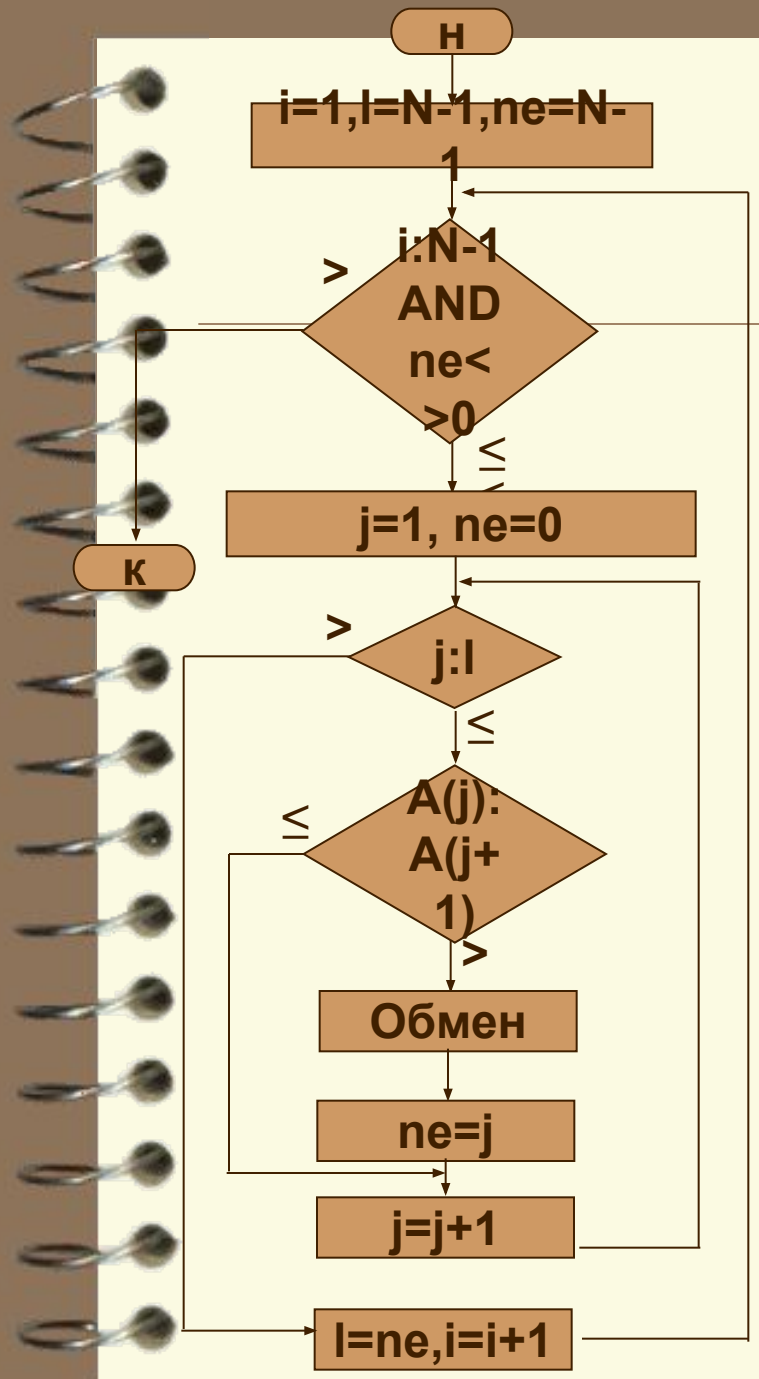
Для обеспечения устойчивости сортировки знак «равно» должен быть именно здесь

Напрашиваются улучшения:

- запоминать индекс последнего обмена в проходе и следующий проход прерывать на данном индексе;
- если в текущем проходе не было обменов, сортировку можно заканчивать

Т.е., можно ввести специальную переменную numb_exc, в которой можно запоминать левый индекс последнего обмена. Если numb_ecx=0, то сортировку можно заканчивать

Блок-схема

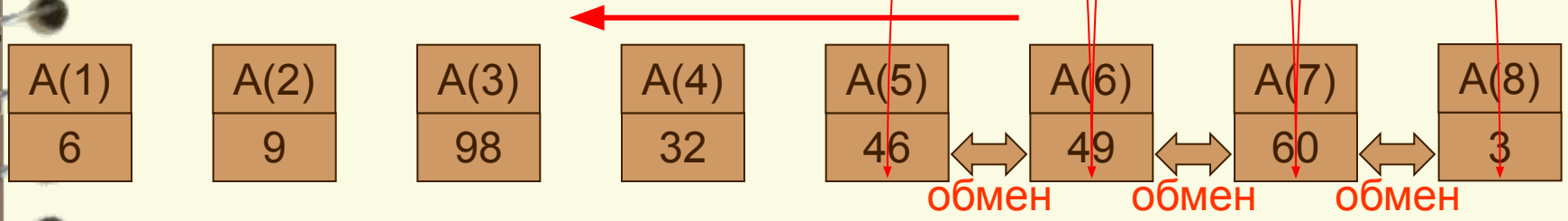


ne (numb_exc) – если в предыдущем проходе не было обменов, ne=0 и сортировка заканчивается

l (limit) – равняется индексу последнего обмена в предыдущем проходе

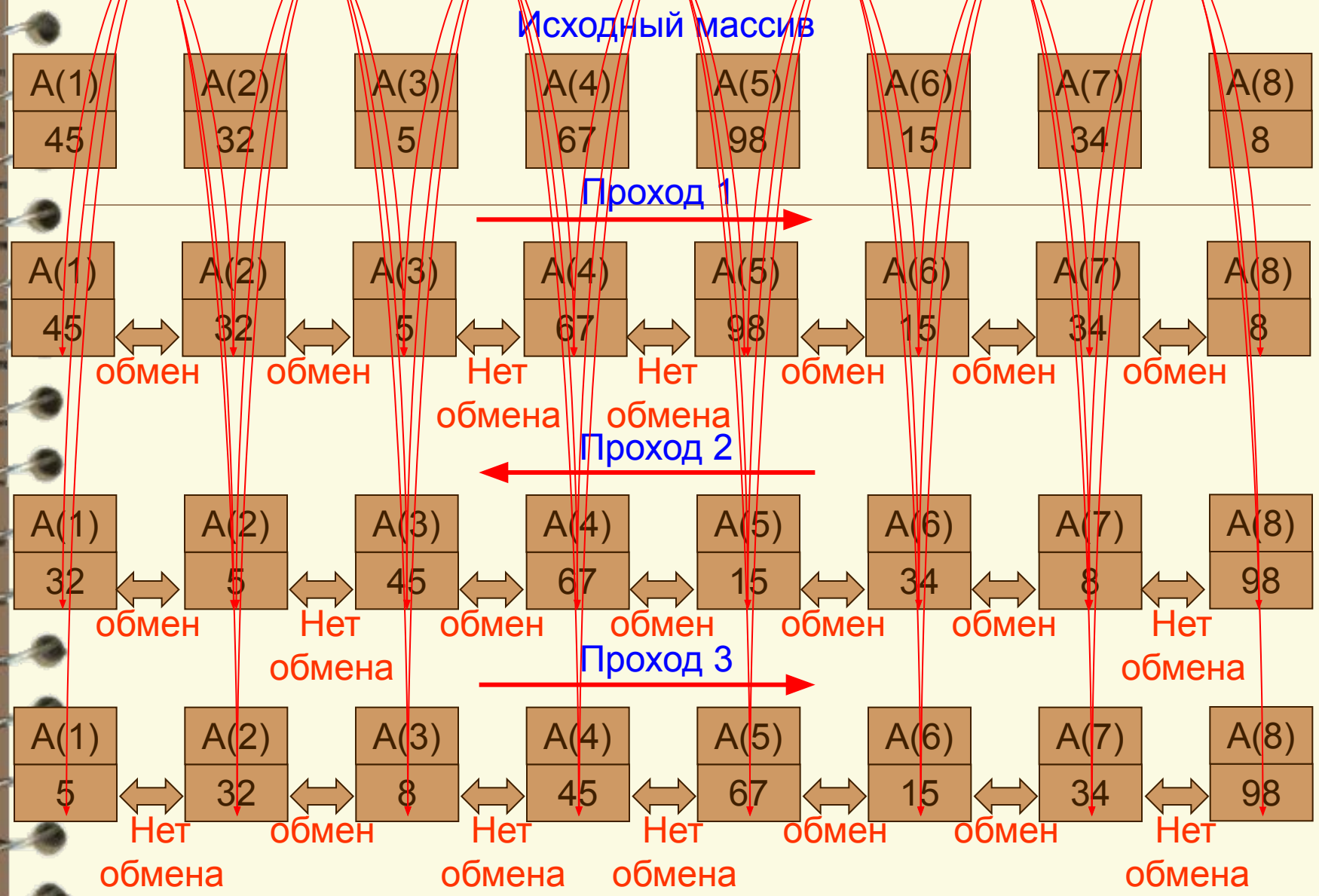
Асимметрия метода:

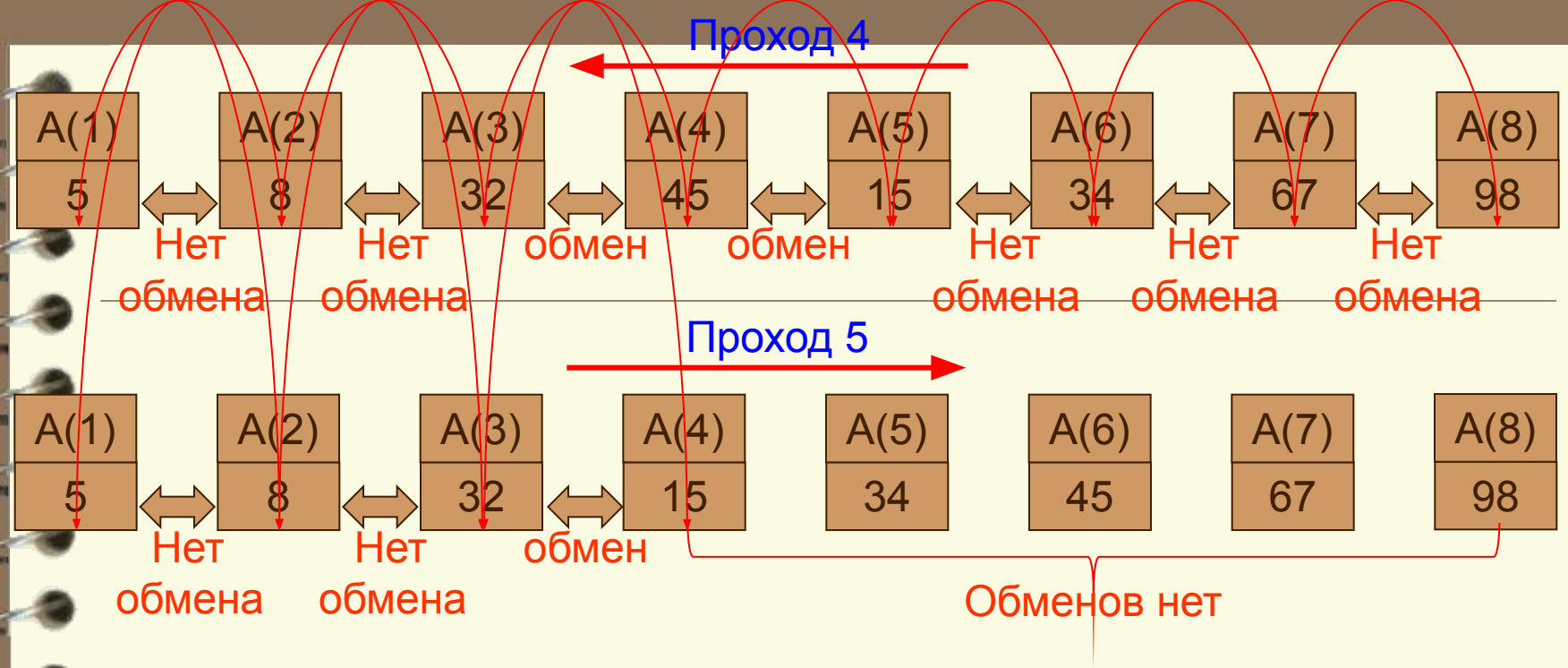
-«легкий» элемент в конце массива в случае просмотра слева направо будет просачиваться на свое место на один шаг за каждый проход, а в случае просмотра справа налево – станет на свое место за один проход



И т.д., вплоть до A(1)

Улучшение: на каждом проходе чередовать направление просмотра – «шейкерная» сортировка


Исходный массив
A(1) 45
A(2) 32
A(3) 5
A(4) 67
A(5) 98
A(6) 15
A(7) 34
A(8) 8
Проход 1
обмен
обмен
Нет обмена
Нет обмена
обмен
обмен
обмен
Проход 2
A(1) 32
A(2) 5
A(3) 45
A(4) 67
A(5) 15
A(6) 34
A(7) 8
A(8) 98
обмен
Нет обмена
обмен
обмен
обмен
обмен
Нет обмена
Проход 3
A(1) 5
A(2) 32
A(3) 8
A(4) 45
A(5) 67
A(6) 15
A(7) 34
A(8) 98
Нет обмена
обмен
Нет обмена
Нет обмена
обмен
обмен
Нет обмена



Данную сортировку целесообразно применять, когда известно, что массив уже почти упорядочен.

Большого эффекта все улучшения дать не могут, т. к. не влияют на число перестановок.

Сортировка прямым выбором

- Идея: массив делится на две части – левую, уже отсортированную и правую, исходную; на каждом проходе в исходном массиве ищется минимальный элемент и меняется местами с первым в неотсортированной части, который переходит в отсортированную часть.
- Число проходов N-1
- Число сравнений C=N(N-1)/2
- Число обменов

Исходный массив

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
45	32	5	67	98	15	34	8

Проход 1

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
45	32	5	67	98	15	34	8

обмен

Проход 2

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	32	45	67	98	15	34	8

обмен

Проход 3

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	45	67	98	15	34	32

обмен

Проход 4

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	15	67	98	45	34	32

обмен

Проход 5

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	15	32	98	45	34	67

обмен

Проход 6

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	15	32	34	45	98	67

! Нет обмена

Проход 7

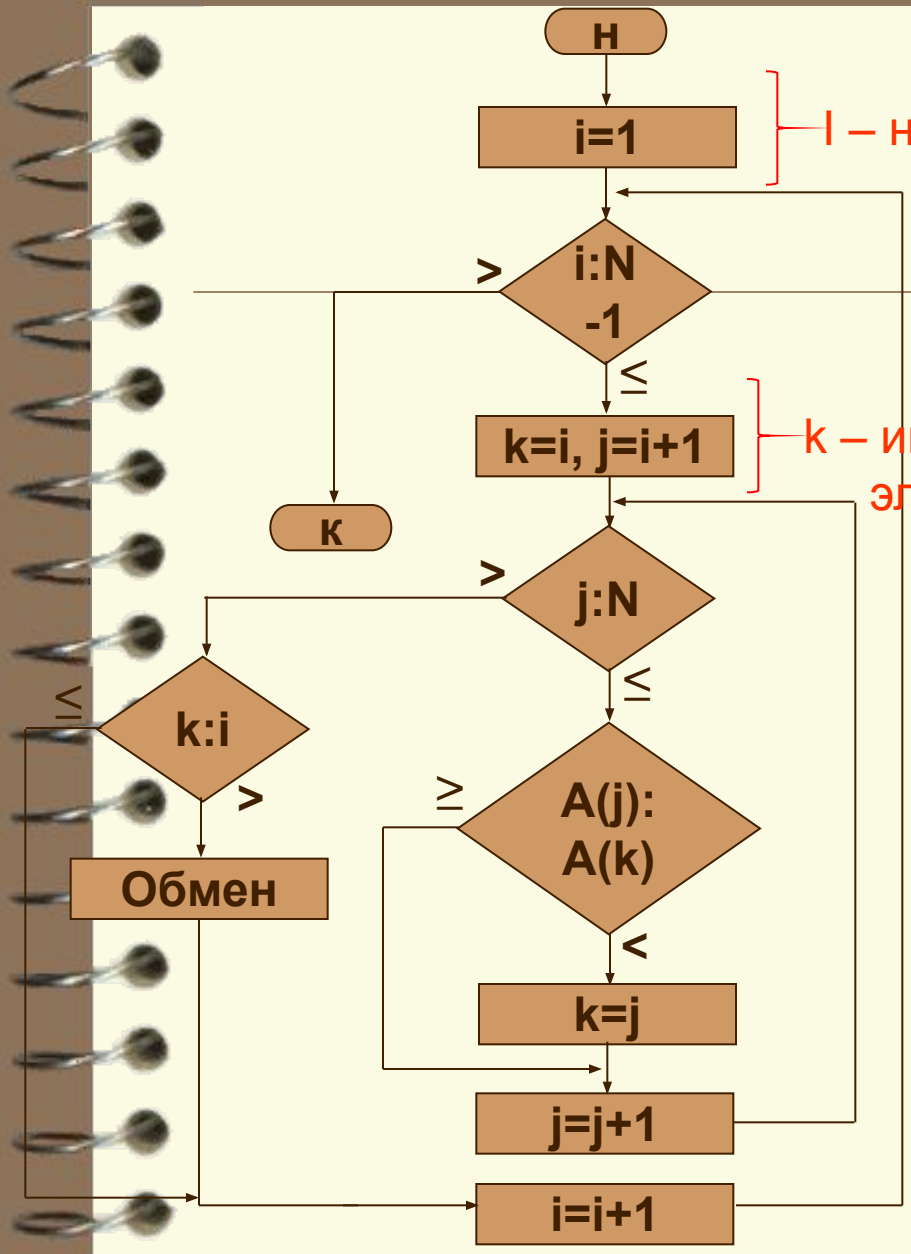
A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	15	32	34	45	98	67

обмен

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	8	15	32	34	45	67	98

Число проходов – 7, число сравнений – 28, число перестановок - 6

Блок схема



I – номер прохода

k – индекс минимального элемента в проходе

Устойчивость сортировки

Если k>i, значит был найден минимальный элемент

Сортировка прямого включения

- Идея: массив делится на две части – левую, уже отсортированную и правую, исходную; на каждом проходе, начиная с i=2 и увеличивая i каждый раз на 1, из исходной последовательности извлекается i-й элемент и вставляется на нужное место в отсортированную часть массива
- Число проходов N-1
- Число сравнений $C=(N^2+N-2)/4$
- Число обменов $M=(N^2+9N-10)/4$

Исходный массив

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
45	32	5	67	98	15	34	8

Проход 1

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
45	32	5	67	98	15	34	8

обмен

Проход 2

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
32	45	5	67	98	15	34	8

обмен обмен

Проход 3

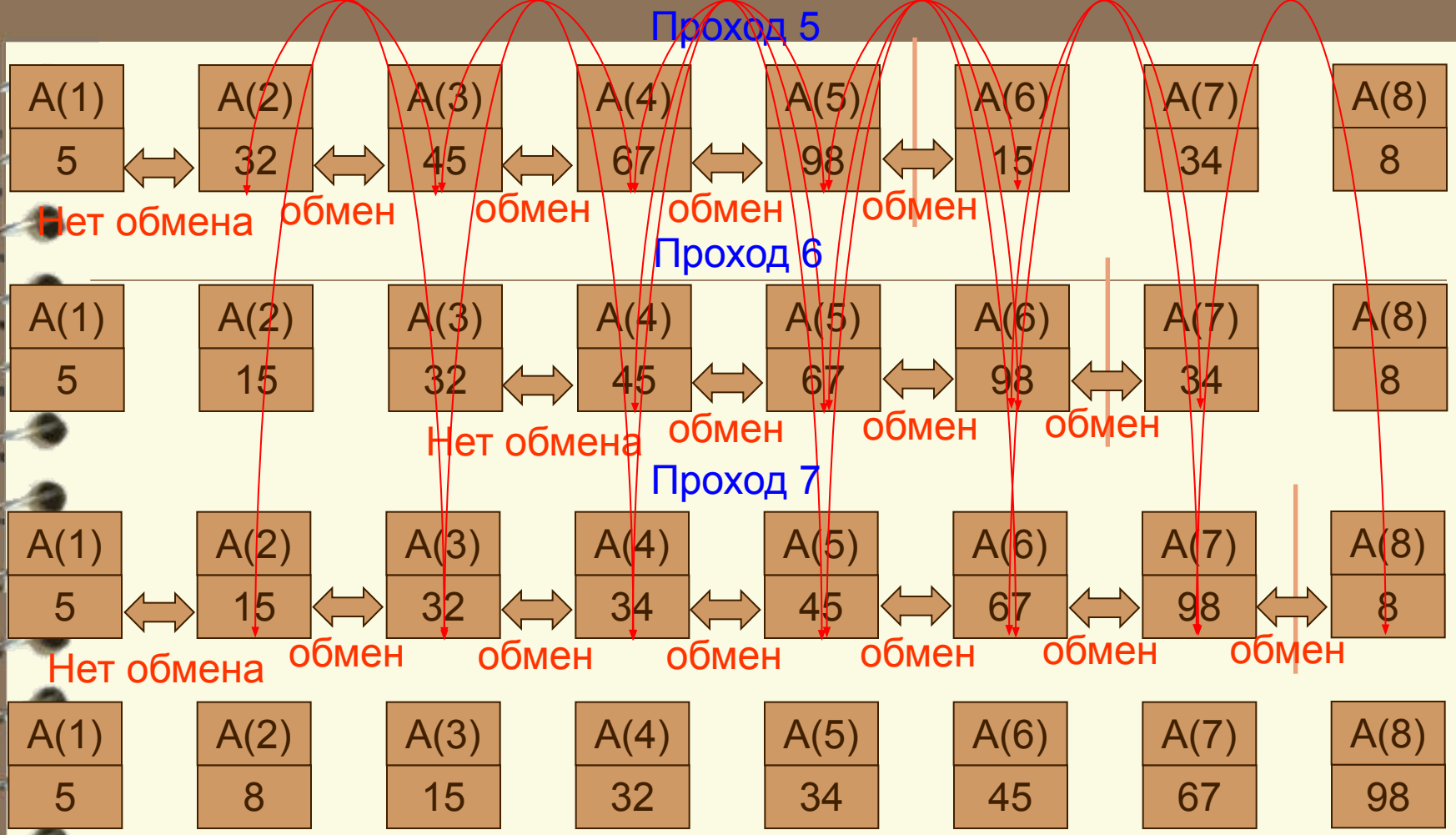
A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	32	45	67	98	15	34	8

! Нет обмена

Проход 4

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
5	32	45	67	98	15	34	8

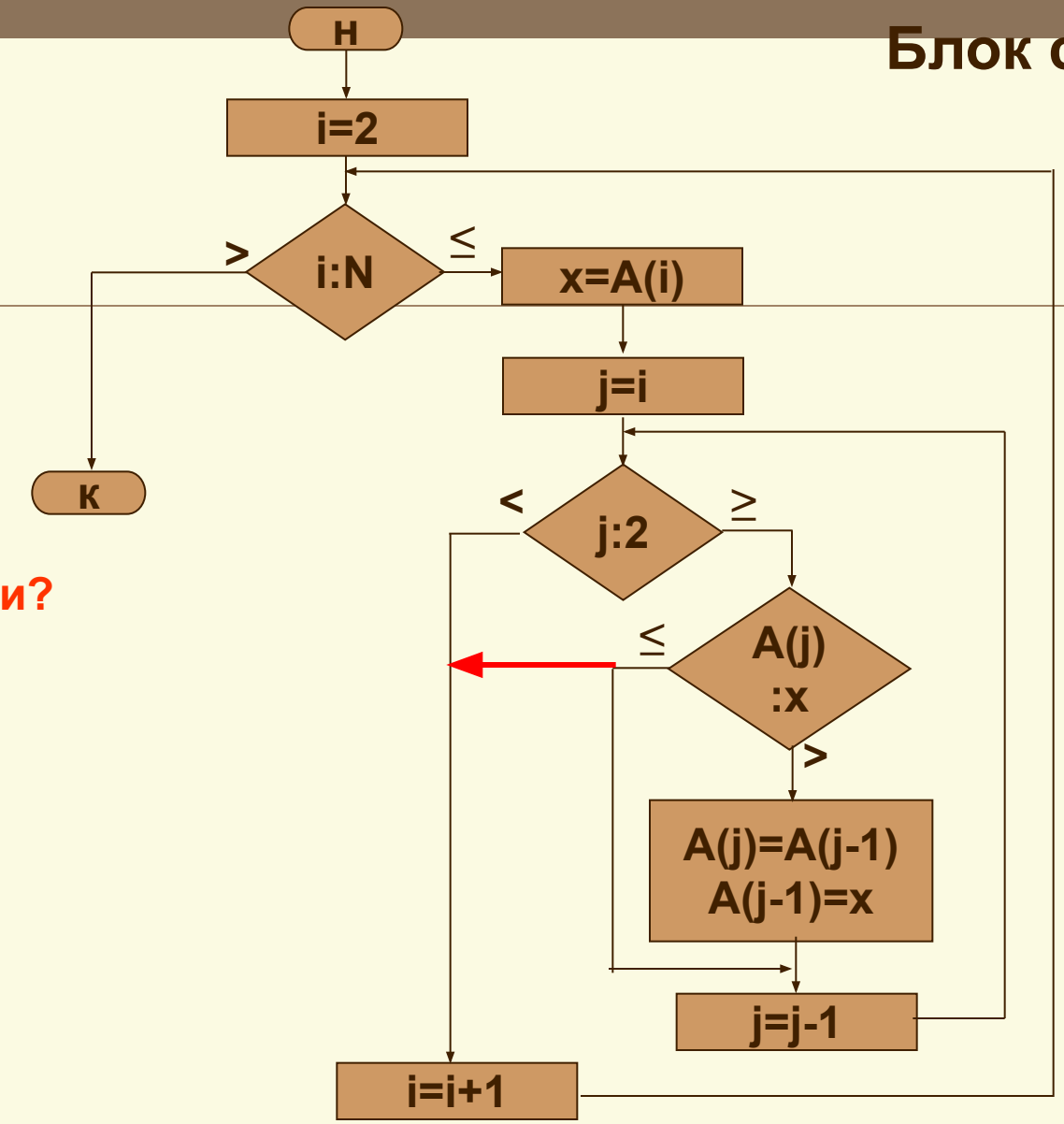
! Нет обмена



Число проходов – 7, число сравнений – 21, число перестановок - 16

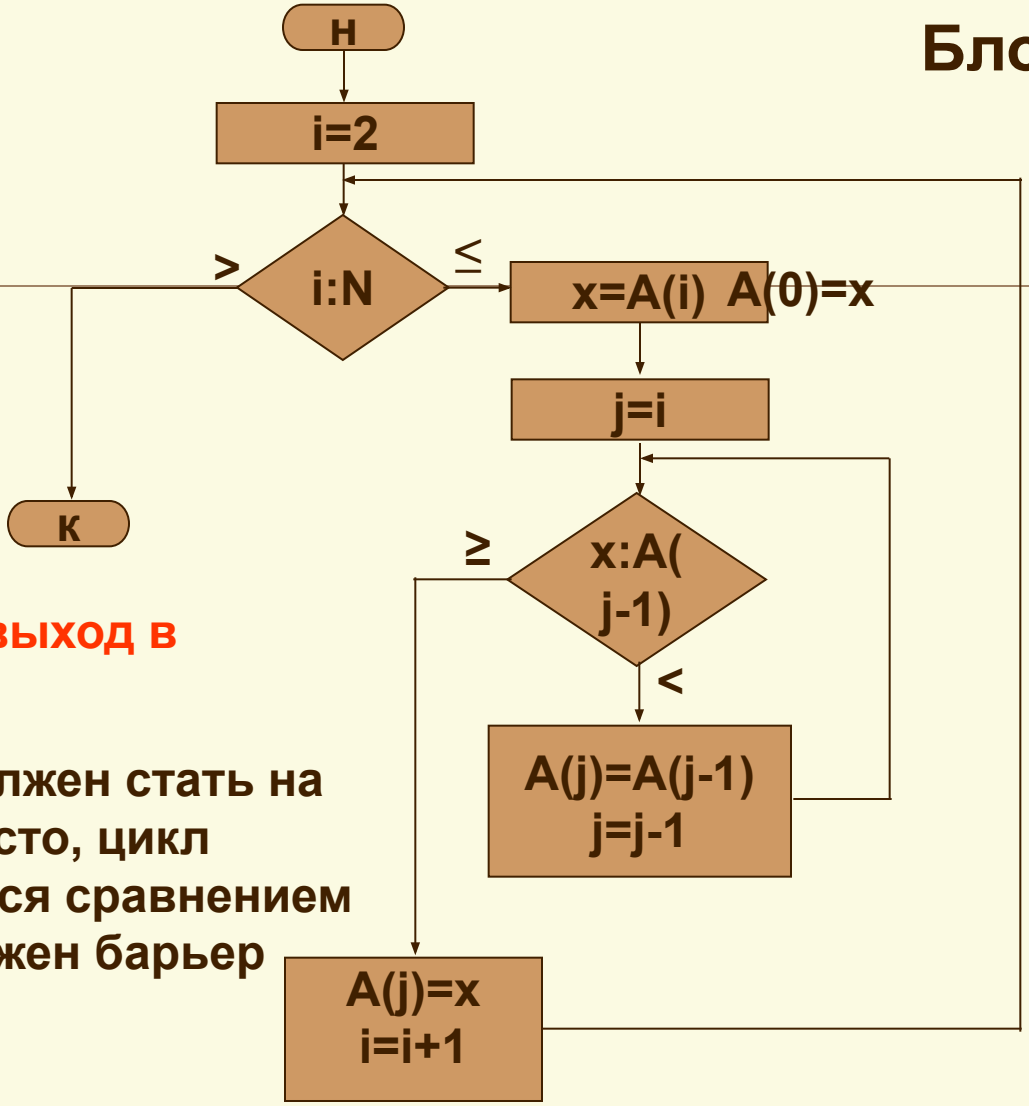
Метод плох из-за сдвижек целых групп элементов

Блок схема



Недостатки?

Блок схема № 2



Возможен выход в ошибку !

Когда x должен стать на первое место, цикл завершается сравнением x:A(0) – нужен барьер A(0)=x