Лекция 2. Статические СД. Массивы

Массивы

Понятия массива и типа массива сильно различаются в сильно и слабо типизированных языках.

Классическое понятие в сильно типизированных языках (например, в языке Паскаль).

Тип массива определяется на основе двух вспомогательных типов: типа элементов массива (базового типа) и типа индекса массива.

В языке Паскаль определение типа массива выглядит:

`type T = array [I] of TO;`, где `TO` — базовый тип, а `I` —тип индекса.

`TO` может быть любым встроенным или ранее определенным типом. Тип индекса `I` должен состоять из конечного числа перечисляемых значений, т.е. быть уточненным, перечисляемым, символьным или булевским типом.

В языках линии Паскаль допускается и неявное определение уточненного типа пассива.
Например, допустимы следующие определения типа массива:

`type T = array [1..6] of integer;` ИЛИ `type T = array ['a'..'c'| of real;`.

Для выборки элемента массива используется конструкция `x[i]`, значением которой является значение i-того элемента массива.

Эта же конструкция может использоваться в левой части оператора присваивания, т.е. элементы массива могут изменяться индивидуально. Кроме того, при подобной строгой типизации массивов допустимы присваивания значений переменных типа массива.

Базовым типом типа массива м.б. любой встроенный или определенный тип, в том числе и тип массива -это будет многомерный массив или матрица:

`type T = array [1..10] of array [1..5] of real;`

можно написать:

```
type T - array [1..10],[1..5] of real;
```

а если `x` — переменная типа `T`, то для выборки скалярного элемента вместо `x[i][j]` можно написать `x[i,j]`.

В слабо типизированных языках используем язык С.

В этом языке нет средств определения типов массива, хотя имеется возможность определения "массивных переменных". Число элементов в массивной переменной определяется либо явно, либо с помощью задания списка инициализирующих значений базового типа. Напр., массивную переменную с четырьмя элементами целого типа можно определить как `int x[4]` (неинициализированный вариант) или как `int X[] = {0, 2, 8, 22}` (инициализированная массивная переменная). Доступ к элементам массивной переменной производится с помощью конструкции выбора, по виду аналогичной соответствующей конструкции в сильно типизированных языках `x[i]`, где `i` — выражение, принимающее целое значение (но, в отличие от языка Паскаль в языке С зафиксирована интерпретация операции выбора на основе более примитивных операций адресной арифметики). Однако, в реализациях языка Си в принципе невозможен контроль выхода значения индекса за пределы массива. Кр. того, по аналогичным причинам невозможно присваивание значений массивных переменных и не допускаются функции, вырабатывающие "массивные значения".

Отмеченные свойства механизма указателей существенно повлияли на особенности реализации в языках С и С++ работы с массивами Имя массива здесь интерпретируется как имя константного указателя на первый элемент массива Операция доступа к i-тому элементу массива агг хотя и обозначается как и в языках линии Паскаль `arr[i|`, имеет низкоуровневую интерпретацию `*(arr+i)`. Поэтому было логично допустить подобную запись для любой переменной `var` с указательным типом `var[i]` интерпретируется как `*(var+i)`. По этой причине понятие массива в С/С++ существенно отличается от соответствующего понятия в Паскале.

Размер массива существенен только при его определении и используется для выделения соответствующего объема памяти. При работе программы используется только имя массива как константный указатель соответствующего типа. Нет операций над "массивными переменными" целиком, в частности, невозможно присваивание. Фактически отсутствует поддержка массивов как параметров вызова функций — передаются именно значения указателей (в связи с этим, при описании формального параметра массива его размер не указывается). Функции не могут вырабатывать "массивные" значения

- **Массив** - это регулярная однородная структура со случайным доступом, т. е., все компоненты структуры ОДНОГО типа, могут выбираться произвольно и являются одинаково доступными.
- **Массив** - это СД, обладающая след, свойствами:

1. Все элементы массива имеют один и тот же тип.
2. Размерность массива определяется при его описании и в дальнейшем не меняется.
3. К каждому элементу массива имеется прямой доступ.

Т.о., каждый элемент массива имеет одно имя и свой индекс, т.е., номер элемента в массиве.
По размерности массивы м.б. одномерными (вектор), двумерными (матрица) и многомерными.

Вектор

Вектор - конечное упорядоченное множество элементов одного типа, называемых элементами вектора. Элементы вектора связаны между собой отношением непосредственного следования, т. е., они м. б. пронумерованы последовательными целыми числами (или переменными любого порядкового типа) - индексами, называемыми селекторами элементов вектора.

Описание векторов в программе должно содержать:

- Имя вектора;
- Минимальный и максимальный индексы элементов;
- Длину элемента (т. е., размер поля памяти, выделенной под элемент).

Важнейшей операцией над векторами является организация доступа к элементу вектора.

Доступ к элементу вектора

Различают логический и физический (структурный, системный) уровни доступа к элементам вектора.

- Логический уровне доступа: указывают имя и индекс
- Физический уровень доступа предполагает вычисление адреса элемента, исходя из его имени, индекса и длины.
- Память ЭВМ - это последовательность ячеек, имеющих адреса.
- Элементы вектора в памяти располагаются последовательно, по возрастанию

индексов, поэтому к каждому элементу возможен прямой доступ. Если известен адрес начала вектора, можно вычислить адрес любого элемента.

- Физической структуре ставится в соответствие дескриптор (заголовок), который содержит общие сведения о физической структуре.

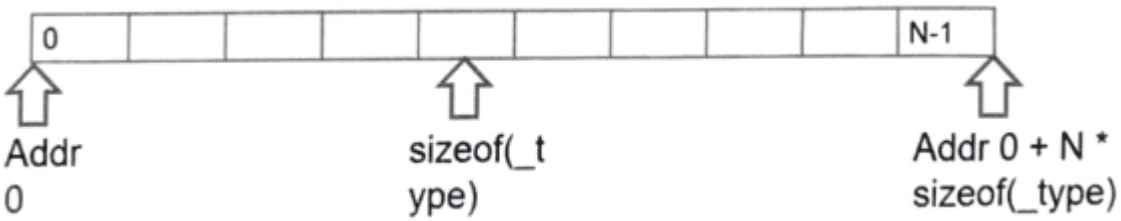
Вектор. Машинное представление.

Адресация элементов структуры.

Элементы вектора размещаются в памяти в подряд расположенных ячейках памяти.

Под элемент вектора выделяется количество байт памяти, определяемое базовым типом элемента этого вектора.

Необходимое число байтов памяти для хранения одного элемента вектора называется слотом. Размер памяти для хранения вектора определяется произведением длины слота на число элементов.



Дескриптор (заголовок) вектора

Пример: Дан вектор `Var V: array [I1..K1] of T;`

- Дескриптор вектора `V` содержит след, информацию:
- Имя - `V`
- Адрес 1-го элемента - `Addr (V[I1])`
- Индекс первого элемента - `I1` (для Си = 0)
- Индекс последнего элемента - `K1`
- Тип элемента - `T`
- Длина элемента - слот - `L`

При этом или тип, или слот относятся к избыточной информации.
Доступ к элементам вектора:

Элемент1	Элемент2	...	ЭлементK1
`Addr (V[I1])`	`Addr (V[I1])+L`		`Addr (V[I1])+L*(K1-I1)`

Т.о. доступ к J-му элементу:
`Addr (V[J1]) = Addr(V[I1]) + (J1 - I1) * L = Addr(V[I1]) -I1 * L + J1 * L = D + J1 * L`, - это есть линейная функция адресации, где `D = Addr (V[I1]) - I1 * L`

Для СИ:

Обращение к i-тому элементу вектора выполняется по адресу вектора плюс смещение к данному элементу. Смещение i-ого элемента вектора определяется по формуле

`Byte_Num = i * sizeof(type)`

Адрес `i` элемента:

`&Byte_Num = &V_name + Byte_Num`

Программисты Си часто вместо выражения вида:

`Имя[I]`

употребляют выражение вида:

`*(Имя+I)`

Двумерные массивы (матрицы)

Двумерным массивом или матрицей называется одномерный массив, компонентами которого являются вектора.

Доступ к любому из элементов матрицы осуществляется по двум индексам: номеру строки и номеру столбца.

Двумерный массив хорошо иллюстрирует различие между логическим и физическим представлением данных.

Логически матрица представляется прямоугольным массивом, каждый элемент которого характеризуется двумя индексами: строки и столбца.

Физически матрицы можно хранить двумя способами: отображением по строкам и отображением по столбцам (способ расположения в памяти зависит от языка программирования). При различном отображении в памяти адрес одного и того же элемента матрицы будет различен. Из определения матрицы следует, что матрица реализуется структурой вектора векторов:

`Mt[I1..K1,I2..K2]` или `Mt[I1..K1][I2..K2]`;

Для того чтобы развернуть этот вектор в "линию", нужна линейная функция адресации.

Дан двумерный массив: `Var Mt: array[1..n, 1..m] of T;`

При отображении по строкам получается след, функция адресации:

```
Addr (Mt[I.JJ) = Addr (Mt[1,1]) + L* (m * (I-1) + (J-1)),
```

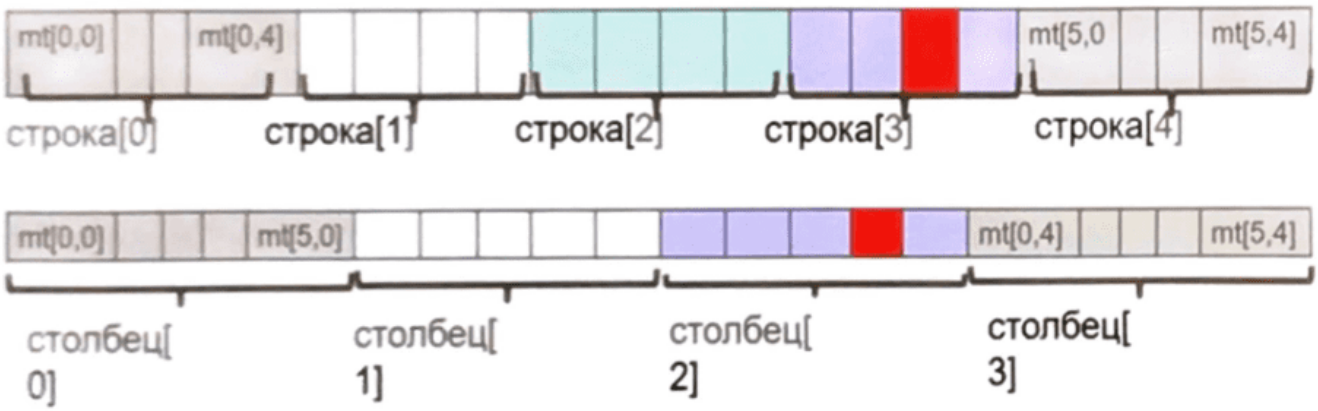
при отображении по столбцам функция адресации:

```
Addr (Mt[I,J]) = Addr (Mt[1,1]) + L* (n * (J-1) + (I-1))
```

Дпя С: Дан двумерный массив: `_type mt [n][m]`;

Функция адресации при отображении по строкам:
`&mt[i][j] = &mt[0][0] + L * (m * i + j),`

Функция адресации при отображении по столбцам:
`&mt[i][j] = &mt[0][0] + L * (п * j + i)`



для массива: `mt(5*4)` при `&mt[0][0] = 200` и `L = 4`
адрес элемента `mt[3,2]` (n = 5, m = 4, i = 3, j = 2) будет:

- при расположении по строкам:
  200 + 4 * (4 * 3 + 2) = 256
- при расположении по столбцам:
  200 + 4 * (5 * 2 + 3) = 252

Многомерный массив

Аналогично матрице многомерный массив:

```
M[L1..K1,L2..K2, ..... Ln..Kn];
```

реализуется как вектор массивов на 1 меньшей размерности.

Многомерный массив логический массив можно преобразовать в физическую одномерную структуру путем процесса линеаризации: линейной последовательностью элементов массива.

Надо только иметь в виду, что при представлении массива по «строкам» быстрее всего меняется последний индекс, а при представлении по «столбцам» - первый индекс.

- Общая формула для вычисления произвольного элемента
- n - мерного массива `M (J1..J2, ..... Jn)`,
- если известен адрес его первого элемента: `M [I1,I2, ,In]` и длина элемента `L`, будет выглядеть так:
  - `Addr (M[J1,J2 Jn])` =
  - $Addr(M[I1, I2, ...In]) - L \cdot \sum_{m=1}^{n} ImDm + L \cdot \sum_{m=1}^{n}$
  - постоянная часть для одного массива (п.ч.)
  - $Addr(M[I1, I2, ...In]) - L \cdot \sum_{m=1}^{n} Im \cdot Dm + L \cdot \sum_{m=1}^{n} Jm \cdot Dm$
  - можно вычислить заранее

Коэффициенты `D1....Dm` можно вычислить по рекуррентным формулам:

1. при отображении по строкам:
   $Dm = (K_{m+1} - I_{m+1} + 1) * D_{m+1}$, где `m = n - 1,...,1` и `Dn = 1`
2. при отображении по столбцам:
   $Dm = (K_{m+1} - I_{m-1} + 1) * D_{m-1}$, где $m = 2.....n$ и $D_1 = 1$,

где I_m - начальный индекс, а K_m - конечный индекс m-ной размерности массива.

Пример:

Составить дескриптор матрицы из целых чисел `Mt [2..3,-1..1]` при отображении по строкам.

Размерность массива `n = 2`, длина элемента = `4`;

```
L1 = 2,K1 = 3,I2 = -1, K2 = 1;
m = n - 1 = 2 - 1 = 1,Dn = D2 = 1;
D1 = (K2 -I2 + 1) * D2 = (1 -(-1) + 1)*1 =3;
L * Dm = L * (I1 * D1 + I2 * D2) = 4 * (2 * 3 + (-1) * 1) = 20
```

Для ускорения вычислений адресных выражений эффективно еще на этапе трансляции вычислить и занести в дескриптор индексные множители L * Dm. Для нашего примера индексные множители будут такими:

`L * D1 = 4 * 3 = 12; L * D2 = 4 * 1 = 4`

Таким образом, мы получили дескриптор с точностью до начального адреса массива:

Имя	Адрес первого Элемента `Addr(M[2,-1])`	Постоянная часть: `addr(M[2,-1]) - 20`	Размерность	I1	K1	I2	K2	L * D1
Mt	4	4-20=-16	2	2	3	-1	1	12

Значит, адрес, напр., элемента `Mt[3,0]` при начальном адресе = 4, `J1` = 3 и `J2` = 0 следующий:

4-20	+	3* 12 + 0*4 =20
Постоянная часть		Индексные множители

Т.к. количество индексов равно размерности массива, то при этом методе получения адресных выражений количество умножений пропорционально размерности массива.

Рассчитать дома:

- Дано: матрица A (6,4), длина L = 6,
- Addr(A[1,1]) = 100;
- Определить элемент матрицы A[3,4]
  1. при расположении по строкам;
  2. при расположении по столбцам.
- Составить дескриптор матрицы
- Осуществить проверку расчета адресов

Вектора Айлиффа

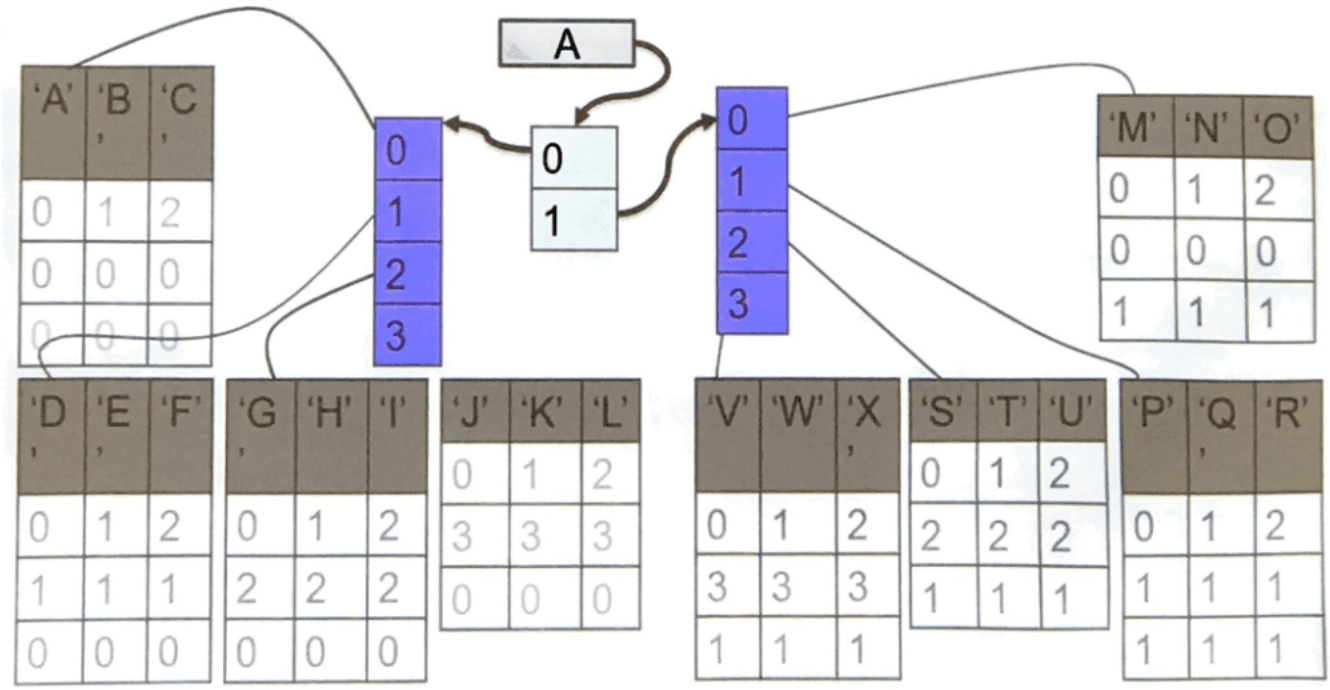
Для ускорения вычислений можно исключить умножение, используя так наз. метод

векторов Айлиффа.

Вектор Айлиффа - вспомогательный дескриптор. Каждый элемент в-ра Айлиффа содержит указатель на дескриптор следующего, более низкого уровня, а дескр-р самого низкого уровня указывает на группы элементов массива. Основной дескриптор массива (нулевой деск-р) хранит указатель вектора Айлиффа первого уровня. Каждый из указателей адресует обл-ть памяти, соответствующую нулевому значению индекса. После прибавления к указателю значения индекса получаем адрес соот-ющего компонента вектора Айлиффа или, в случае вектора нижнего уровня, - адрес требуемого элемента массива. Т. о., к элементу массива `M[J1 ,J2, ,Jn]` можно обратиться, пройдя по
цепи от основного дескр-ра через все компоненты вектора Айлиффа, после чего и будет получен адрес требуемого элемента:

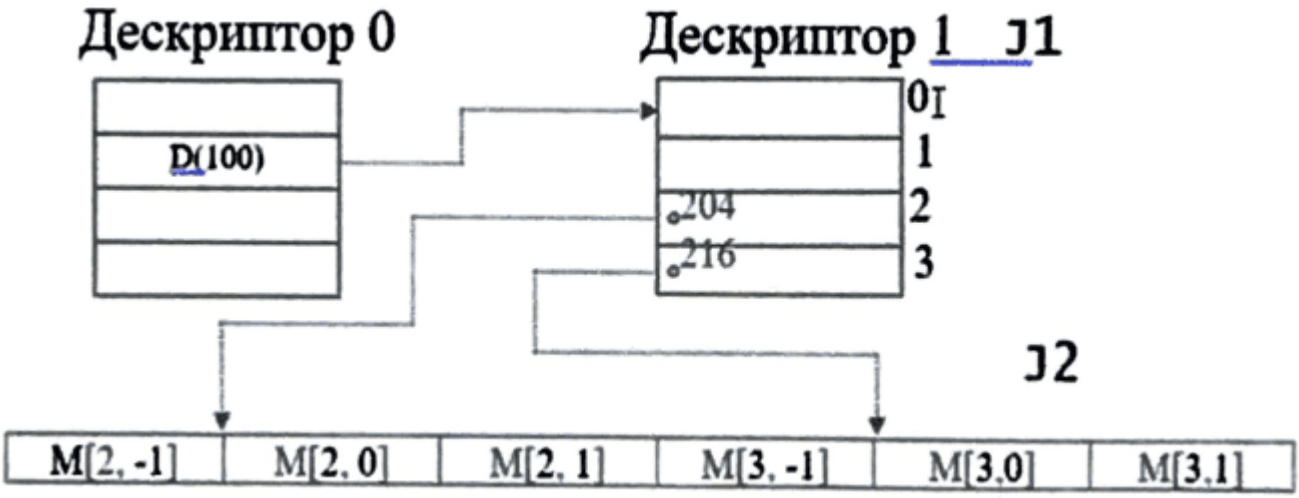
`Addr(M[J1,J2 ... Jn]) = (...(...(D + Л) + J2) +...) + Jn`,
где `D` - значение указателя вектора Айлиффа первого уровня, хранящееся в основном дескрипторе.

Адресация элементов с помощью векторов



Пример: Структура вектора Айлиффа для матрицы
M[2.. 3, -1.. 1] с длиной элемента L=4

Напр., дескриптор 1 имеет адрес 100, адрес начала массива - 200, тогда вектора Айлиффа будут выглядеть так:



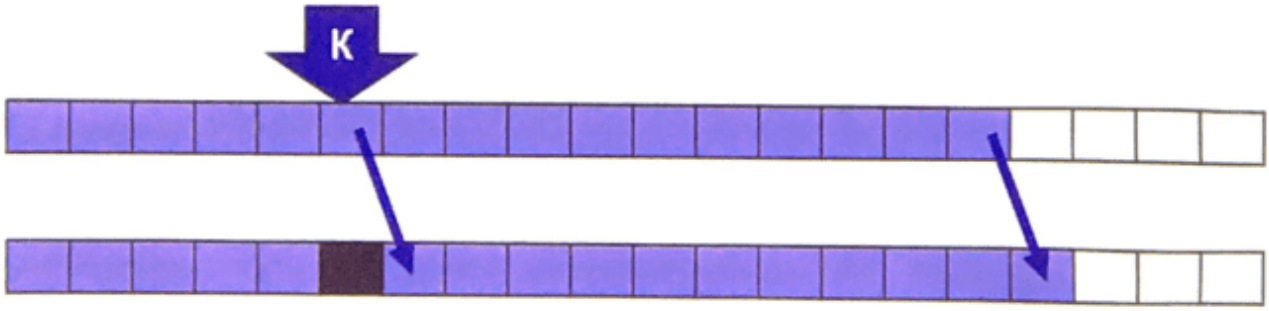
Пример: Для трехмерного массива `M(4..5,-1..1,0..1) вектора Айлиффа будут:

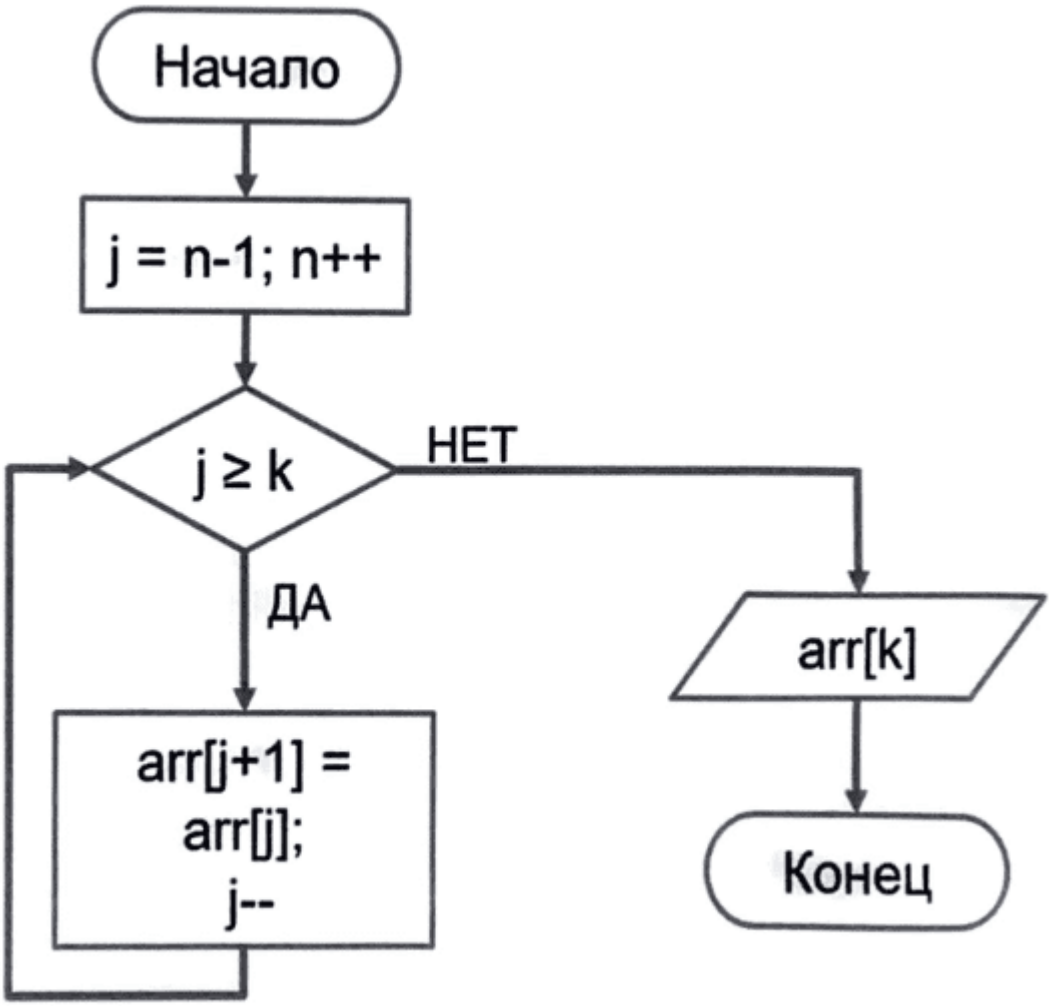
Операции над созданными структурами данных

- **Обход структуры**: доступ к каждому элементу структуры («посещение» элемента);
- **Поиск**: нахождение расположения элемента с данным значением (ключом);
- **Вставка**: включение нового элемента в структуру;
- **Удаление**: исключение элемента из структуры.

Алгоритм вставки элемента в массив на позицию к

1. Проверить, есть ли `k`-я позиция в описанных индексах массива и не полон ли массив (тогда вставлять нельзя!)
2. Увеличить количество элементов массива на `1`
3. Переместить на одну позицию вправо все элементы массива, начиная с конца до позиции вставки.
4. Вставить в освободившуюся позицию элемент.





Алгоритм удаления элемента на позиции k из массива

1. Проверить, есть ли `k`-я позиция в описанных индексах массива и не последний ли он.
2. Начиная с `k + 1` элемента переставить каждый последующий элемент на место предыдущего и
3. уменьшить количество элементов массива на `1`