3. Понятие таблицы…

⌛ Как задать функцию f ?

↳Возможный способ – *табличное задание функции*

имя	адрес
AB	3012
…	…
FE	4408
…	…

- ***Таблица*** - последовательность строк ***(записей)***
- **Запись** может состоять из нескольких ***полей***
- Одно из полей должно задавать ***имя записи*** (***ключ***), остальные поля образуют ***тело записи***

⌛ Таблица – линейная структура ?

4. Важность понятия таблиц

- ☑ Организация доступа по имени для управления информацией в привычной для человека форме
- ☑ Представление данных во многих задачах из разных областей приложений (таблицы идентификаторов, номенклатура изделий, словари и т.п.)
- ☑ Абстрагирование от проблем распределения памяти при размещении данных
- ☑ Реализация концепции *ассоциативной памяти* (память с доступом к данным по содержимому в этих данных)
- ☑ Отображение на ЭВМ такого важного математического понятия как *множества*

5. Формализация…

☑ Пусть

- **K** - множество имен,
- **A** - множество значений,
- $\mathbf{A^* = A + æ}$ - расширенное множество ($æ \notin A$),
- $\mathbf{Z = K \times A^*}$ - множество записей ($z = \langle k,a \rangle \in Z$),
- $\boldsymbol{Z}$ - множество всех подмножеств **Z**.

↳Тогда таблица есть структура $\mathbf{T = \langle Z, p \rangle}$

- **p** - базисное отношение включения,
- $t_n^k \in T$ - текущий элемент (состояние) из k записей (n – размер памяти),

☑ Основные операции

- $T[k] \to A^*$ - поиск по ключу (при $\langle k,* \rangle \notin \mathbf{T} \Rightarrow a = \mathbf{æ}$ и $\delta=3$),
- $T + z \to T'$ - вставка записи (при $k=n \Rightarrow T' = T$ и $\delta=2$),
- $T - k \to T'$ - исключение записи (при $k=0 \Rightarrow T' = T$ и $\delta=1$).

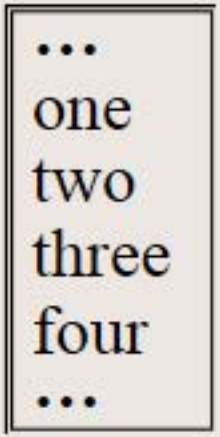
☑ Дополнительные операции

- $T(n) \to t_n^0$ - создание таблицы,
- $\alpha_0(T)$ - предикат проверки пустоты ($\alpha_0(T)=1$ при $k=0$),
- $\alpha(T)$ - проверка переполнения памяти ($\alpha(T)=1$ при $k=n$).

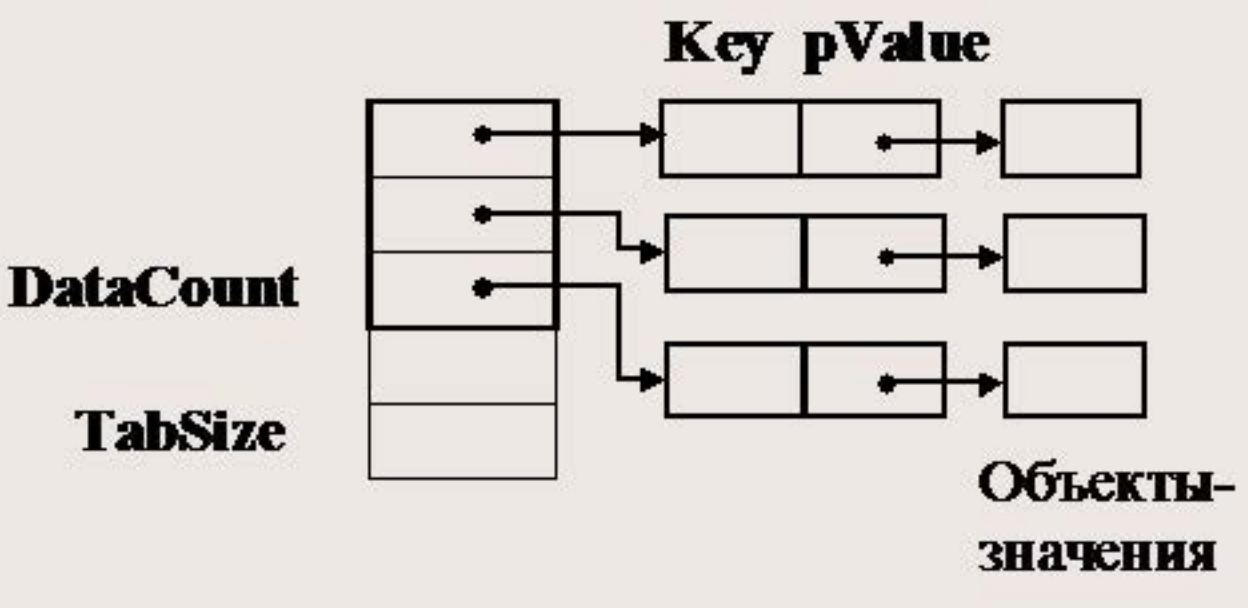
⇩ При завершении операции формируется код выполнения δ

$=1$ – таблица пуста,
$=2$ – таблица заполнена,
$=3$ – записи в таблице нет,
$=4$ – запись в таблице уже есть

Идея похода (пример) – начальное накопление списка слов при изучении иностранного языка



- ☑ **Поиск** – линейный (последовательный) просмотр
- ☑ **Вставка** – добавление в коней списка
- ☑ **Удаление** – исключение (стирание) с уплотнением (ex., путем переписыванием последней записи в удаляемую строку)


Key pValue
DataCount
TabSize
Объекты-
значения


TDataCom
TTable
TArrayTable
TScanTable
Абстрактный базовый класс – спецификации методов таблицы
Абстрактный класс – управление структурой хранения
Класс, обеспечивающий реализацию просматриваемых таблиц

2. Реализация – класс TArrayTable

- ☑ Управление памятью (выделение/освобождение)
- ☑ Расширение к-ва записей, к которым возможен доступ

  Методы GetKey и GetDatValue применимы к первой (FIRST), текущей (CURRENT) или последней (LAST) записи таблицы; желаемый вариант доступа задается через параметр метода)
- ☑ Введение операции прямого доступа к записям
  - SetCurrentPos – установить текущую позицию на запись с заданным номером
  - GetCurrentPos – получить номер текущей записи
- ☑ Реализация методов навигации (!)

3. Оценка эффективности…

- ☑ Эффективность алгоритма – время решения задачи (*временная сложность алгоритма* T)
- ☑ Время выполнения алгоритма на ЭВМ зависит от многих факторов – возможный способ состоит в измерении временной сложности в инструкциях (шагах), которые нужно выполнить для достижения результата
- ☑ Выражение для оценки временной сложности обычно зависит от размера (количества) входных данных N

Пример. Сложность алгоритма перемножения матриц

$$T = 2N^3-N^2,$$

где N есть порядок перемножаемых матриц

3. Оценка эффективности…

☑ Детальное построение выражения для временной сложности алгоритма часто является затруднительным, во многих случаях достаточно знания *верхних граничных оценок*

Определение 3.2. Функция f является верхней границей для функции g

$$g(N)=O(f(N)),$$

если

$$\exists n_0,\ c>0,\ \forall\ n>n_0 \Rightarrow g(N) \le f(N)$$

☑ Если $g(N)=O(f(N))$, то говорят, что $g(N)$ имеет *порядок* (*степень* или *скорость*) *роста* $f(N)$.

↳При построении верхних оценок определяющим является элемент наиболее высокого порядка формульного выражения сложности алгоритма

Типовые зависимости граничных оценок

- O(1) – константная оценка,
- O(logN) – логарифмическая оценка,
- O(N) – линейная оценка,
- O(N logN) – линейно- логарифмическая оценка,
- $O(N^a)$ – полиномиальная оценка (a - константа),
- $O(a^N)$ – экспоненциальная оценка (a - константа).

$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
0,0	1	0,0	1	1	2
1,0	2	2,0	4	8	4
2,3	5	11,6	25	125	32
3,3	10	33,2	100	1000	1024
3,9	15	58,6	225	3375	32768
4,3	20	86,4	400	8000	1048576
4,9	30	147,2	900	27000	1073741824
5,3	40	212,9	1600	64000	1099511627776
5,6	50	282,2	2500	125000	1125899906842620

Сложность операций просматриваемых таблиц

Поиск

- $T_{min} = 1$
- $T_{max} = N$
- $T_{ср} = N/2$ (поиск имеющихся в таблице записей)
  $= p(N/2)+(1-p)N$ (p-вероятность наличия записи)

↳ При построении оценок использовалось предположение, что вероятность использования всех ключей является одинаковой

Сложность операций просматриваемых таблиц

Вставка

- $T_{ср} = p(N/2)+(1-p)(N+1)$

  $= N+1$ (вставка без дублирования)

Удаление

- $T_{ср} = p(N/2+1)+(1-p)N$

- Просматриваемые таблицы эффективны при незначительном количестве записей
- Эффективное использование просматриваемых таблиц предполагает:
  - поиск и удаление только имеющихся в таблице записей
  - вставка только новых записей (для исключения проверки дублирования)

- Таблицы являются важным и широко распространенным в практике типом структур данных
- Просматриваемые таблицы являются одним из самых простых способов организации таблиц; данный подход является эффективным при небольшом наборе имеющихся записей
- Временная сложность как показатель эффективности алгоритма; методика анализа сложности (O-нотация, виды зависимостей, минимальная и максимальная сложность, сложность в среднем)

☑ Аксиомы

- $T=T(n) \Rightarrow t = t_n^0$ и $\alpha_0(t)=1$ – созданная таблица является пустой
- $T'=T+z \Rightarrow \alpha_0(t')=0$ – таблица после операции вставки не пуста
- $T'=T-k \Rightarrow \alpha(t') =0$ – таблица после исключения записи не полна
- $(t+<k,a>)[k]\neq æ$ – после вставки записи в таблице поиск этой же записи должен быть успешным
- $(t-k)[k]=æ$ – после удалении записи в таблице данная запись должна отсутствовать
- $k1\neq k2 \Rightarrow (t \pm k2)[k1]=t[k1]\pm k2$ – результат поиска не зависит от операций вставки и удаления записей с другими ключами