Tabell (eng. dictionary)

213	Andersson	720220
103	Svensson	731023
72	Andersson	780718
202	Larsson	760503
17	Karlsson	740430
170	Nilsson	770809
201	Pettersson	700226

Nyckel

```
HashTable table = new HashTable(17);
InfoType info = new InfoType();

info.name = "Persson";
info.date = "790606";
table.hashInsert(123, info);

i = table.hashSearch(202);
```


Tabell

Applikation som använder tabell:
- Ordlista
- Personregister
- Symboltabell

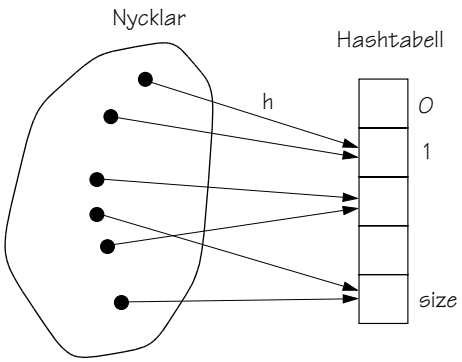
Tabell ADT

T.table(128); T.insert(Key, Info)
T.delete(Key) T.update(Key, Info)
T.enumerate()

Representation:
- Sorterad lista O(n)
- Binärt träd (AVL) O(log n)
- Hashtabell O(1)

Hashtabell

Nyckel och information lagras i en array. Positionen i arrayen bestäms av en hash-funktion.



Enkel hashfunktion:
- Gör om nyckeln till ett heltal n
- Index = n % size

T.ex. om nyckeln redan är ett heltal:

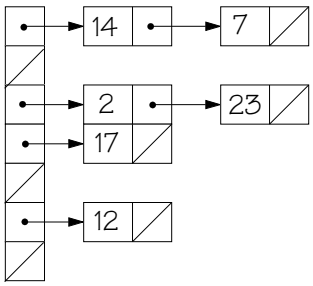
```
int h(int n) {
  return n % size;
}
```


Kollisionshantering

Hur hantera kollisioner?

- Öppen adressering med:
  - Linjär sondering
  - Dubbelhashning
  - Kvadratisk sondering
- Separata kedjor (eng. separate chaining)
- "Hinkar" (eng. buckets)

Hashning med separat kedjning



Fyllnadsgrad och kluster

Fyllnadsgrad (eng. load factoring):

- Andelen av arrayen som är fylld.

Kluster (eng. clustering):

- Sekvens av arraypositioner som är upptagna.
  - Linjär sondering leder till (primära) kluster

Teoretiska resultat

Antal sonderingar vid olika fyllnadsgrad:

Fyllnadsgrad (%)						
	10	25	50	75	90	99
Lyckad sökning						
chaining	1.05	1.12	1.25	1.37	1.45	1.49
linear prob.	1.06	1.17	1.50	2.50	5.50	50.5
double hash	1.05	1.15	1.39	1.85	2.56	4.65
Misslyckad sökning						
chaining	0.1	0.25	0.5	0.75	0.9	0.99
linear prob	1.12	1.39	2.50	8.5	50.5	5000
double hash	1.11	1.33	2.0	4.0	10.0	100

Exempel

```
class InfoType {
  // For exemple
  String name;
  String date;
}

class TableEntry {
  int        key;
  InfoType   info;
}
```

Exempel (forts.)

```
class HashTable {

  public final static int emptyKey = 0;
  int      M;
  int      count;
  TableEntry[] T;

  HashTable(int tableSize) {
    M =     tableSize;
    count = 0;
    T =     new TableEntry[M];

    for (int i = 0; i < M; i++ ) {
      T[i] = new TableEntry();
      T[i].key = emptyKey;
    }
  }

  void hashInsert(int K, InfoType I) {
    // Se OH 10
  }

  int hashSearch(int K) {
    // Se OH 11
  }

  int h(int K) {
    return K % M;
  }

  int p(int K) {
    int r = (K / M) % M;
    return (r == 0) ? 1 : r;
  }

} //end class HashTable
```


Exempel (forts.)

```
void hashInsert(KeyType K, InfoType I) {

  int   i;
  int   probeDecrement;

  i = h(K);
  probeDecrement = p(K);

  while (T[i].key != emptyKey) {
    i -= probeDecrement;
    if (i < 0) {
      i += M;
    }
  }

  T[i].key = K;
  T[i].info = I;
  count++;
}
```


Exempel (forts.)

```
int hashSearch(KeyType K) {

  int      i;
  int      probeDecrement;
  KeyType  probeKey;


  // initializations
  i = h(K);
  probeDecrement = p(K);
  probeKey = T[i].key;    |

  // search loop
  while((K != probeKey)&&(probeKey != emptyKey)) {
    i -= probeDecrement;
    if (i < 0) {
      i += M;
    }
    probeKey = T[i].key;
  }

  // determine success or failure
  if (probeKey == emptyKey) {
    return -1;
  } else {
    return  i;
  }
}
```


Hur välja hashfunktion

Beror på:

- Typ av nycklar
- Nycklarnas distribution
- Arraystorlek
- Funktionens enkelhet

Vanliga metoder:

- Divisionsmetoden
- Folding
- Middle squaring
- Truncation
- …

Olika representationer

	construct	full?	find/update
Sorterad array	O(n)	O(1)	O(log n)
AVL	O(1)	O(1)	O(log n)
Hashtabell	O(n)	O(1)	O(1)

	insert	delete	enum
Sorterad array	O(n)	O(n)	O(n)
AVL	O(log n)	O(log n)	O(n)
Hashtabell	O(1)	O(1)	O(n log n)