
Fundamentals of
Data Structures using C

# Hash Functions

**B.Bhuvaneswaran, AP (SG) / CSE**

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

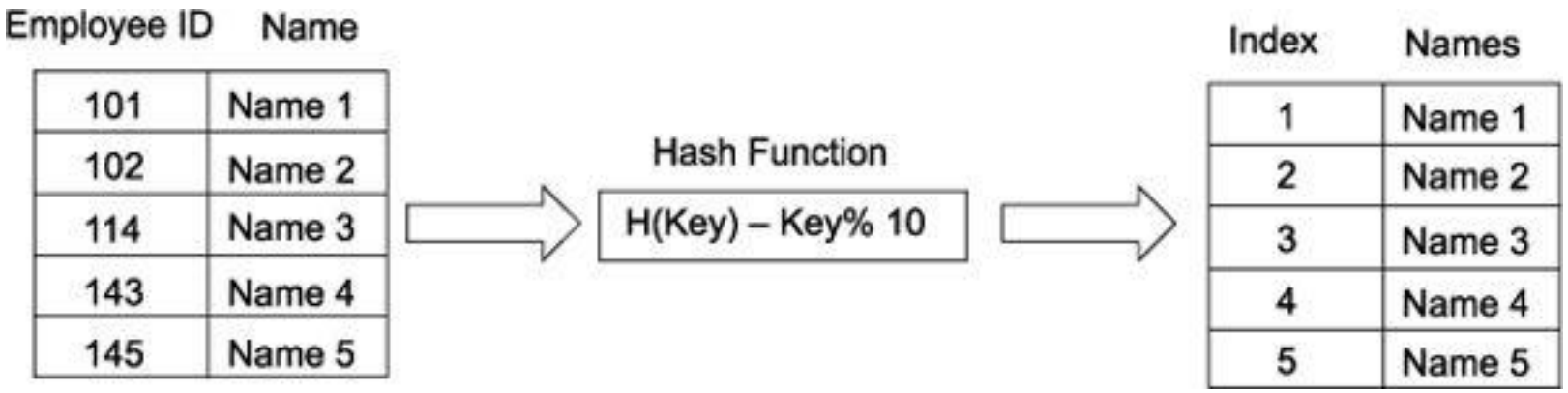
# Introduction

- Hashing finds the location of an element in a data structure without making any comparisons.
- In contrast to the other comparison-based searching techniques, like linear and binary search, hashing uses a mathematical function to determine the location of an element.
- This mathematical function called hash function accepts a value, known as key, as input and generates an output known as hash key.
- The hash function generates hash keys and stores elements corresponding to each hash key in the hash table.
- The keys that hash function accepts as input could be a digit associated with the input elements.

# Introduction

- Let us consider a simple example of a file containing information for five employees of a company.
- Each record in that file contains the name and a three-digit numeric Employee ID of the employee.
- In this case, the hash function will implement a hash table of five slots using Employee IDs as the keys.
- That means, the hash function will take Employee IDs as input and generate the hash keys, as shown in Fig.

# Generating Hash Keys



# Introduction

- In the hash table generated in the above example, the hash function is Employee ID%10.
- Thus, for Employee ID 101, hash key will be calculated as 1. Therefore, Name1 will be stored at position 1 in the hash table.
- For Employee ID 102, hash key will be 2, hence Name2 will be stored at position 2 in the hash table.
- Similarly, Name3, Name4, and Name5 will be stored at position 4, 3, and 5 respectively, as shown in Fig.
- Later, whenever an employee record is searched using the Employee ID, the hash function will indicate the exact position of that record in the hash table.

# Good Hash Function

- Minimize collisions
- Be easy and quick to compute
- Distribute key values evenly in the hash table
- Use all the information provided in the key

# Methods of Hashing Functions

- Mid square method
- Modulo division or division remainder method
- Folding method
- Pseudo random number generator method
- Digit or character extraction method
- Radix transformation method

# Mid Square Method

- In this method, the key is squared and the middle part of the result based on the number of digits required for addressing is taken as the hash value.
- This method works well if the keys do not contain a lot of leading and trailing zeros.

  H(X) = return middle digits of $X^2$
- For example:
- Map the key 2453 into a hash table of size 1000, there, X = 2453.

  $X^2$ = 6017209
- Extract middle value 172 as the hash value.

# Modulo Division or Division Remainder Method

- This method computes the hash value from the key using the modulo (%) operator.
- Here, the table size that is power of 2 like 32, 64 and 1024 should be avoided as it leads to more collisions.
- It is always better to select table size not close to the power of 2.

  H(Key) = return Key % TableSize
- For example:
- Map the key 4 into a hash table of size 4.

  H(4) = 4 % 4 = 0

# Folding Method

- This method involves splitting keys into two or more parts each of which has the same length as the required address with the possible exception of the last part and then adding the parts to form the hash address.
- There are two types of folding method. They are:
  - Fold shifting method
  - Fold boundary method

# Fold Shifting Method

- Key is broken into several parts of same length of the required address and then added to get the hash value. Final carry is ignored.
- For example:
- Map the key 123203241 to a range between 0 to 999.

  Let X = 1 2 3 2 0 3 2 4 1

  Position X into 123, 203, 241, then add these three values.

  123 + 203 + 241 = 567 to get the hash value.

# Fold Boundary Method

- Key is broken into several parts and reverse the digits in the outermost partitions and then add the partition to form the hash value.
- For example:

  X = 123203241

  Partition = 123, 203, 241

  Reverse the boundary partition = 321, 203, 142

  Add the partition = 321 + 203 + 142

  Hash value = 666.

# Pseudo Random Number Generator Method

- This method generates random number given a seed as parameter and the resulting random number then scaled into the possible address range using modulo division.
- It must ensure that it always generates the same random value for a given key.
- The random number produced can be transformed to produce a valid hash value.

$$X_{i+1} = A\ X_i \ \%\ \text{TableSize}$$

# Digit or Character Extraction Method

- This method extracts the selected digits from the key and used it as the address or it can be reversed to give the hash value.
- For example:
- Map the key 123203241 to a range between 0 to 9999.

  Select the digits from the positions: 2, 4, 5, and 8.

  Therefore the hash value = 2204.
- Similarly, the hash value can also be obtained by considering the above digit positions and reversing it, which yields the hash value as 4022.

# Radix Transformation Method

- In this method a key is transformed into another number base to obtain the hash value.
- For example:
- Map the key $(8465)_{10}$ in the range 0 to 999 using base 15.

  $(8465)_{10} = (2795)_{15}$
- The key 8465 is placed in the hash address 2795.

# Queries?

# Thank You!