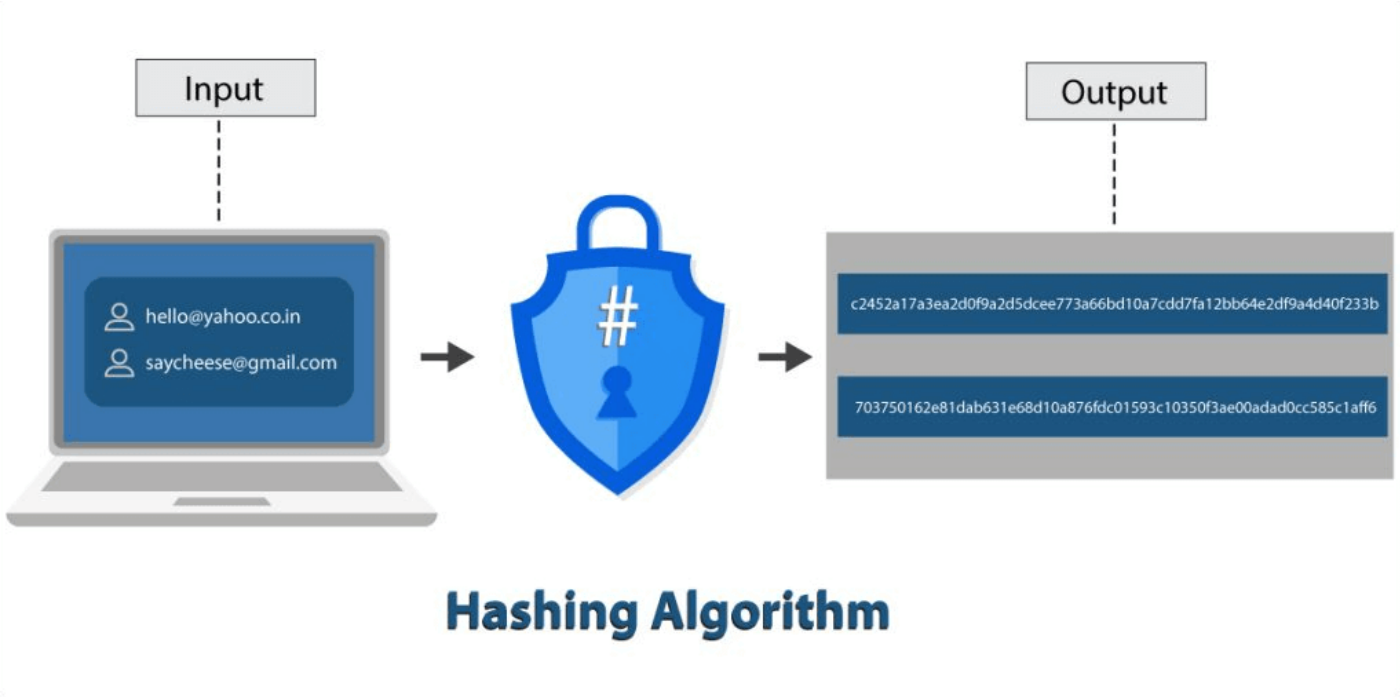
Thuật toán Hashing

Hashing là một thuật toán giúp phân biệt giữa các khối dữ liệu với nhau. Ứng dụng nổi bật và quan trọng nhất của thuật toán này đó chính là cho phép tìm kiếm và tra cứu một bản ghi dữ liệu đã cho trước và mã hóa bản ghi đó một cách nhanh chóng.

Thuật toán này cho phép phát hiện và sửa lỗi khi dữ liệu bị làm nhiễu bởi các quá trình ngẫu nhiên.



Ngoài ra, thuật toán này cũng có thể sử dụng cho phép tạo ra các giá trị dữ liệu không trùng lặp (Unique) và ứng dụng trong các bộ định tuyến để lưu trữ địa chỉ IP.

Ví dụ đơn giản về thuật toán Hash trong C++:

Key	**Index (using a hash function)**	**Data**
12	12%10 = 2	23
10	10%10 = 0	34
6	6% 10 = 6	54
23	23 % 10 = 3	76
54	54 %10 = 4	75
82	81 %10 = 1	87

Vị trí trong bảng hash sẽ là:

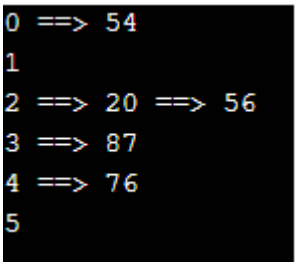
0	1	2	3	4	5	6	7	8	9
34	87	23	76	75		54			

```
#include <iostream>

#include <list>

using namespace std;
class HashMapTable {
  // size of the hash table
  inttable_size;
  // Pointer to an array containing the keys
  list < int > * table;
  public:
    // creating constructor of the above class containing all the methods
    HashMapTable(int key);
  // hash function to compute the index using table_size and key
  inthashFunction(int key) {
    return (key % table_size);
  }
  // inserting the key in the hash table
  void insertElement(int key);
  // deleting the key in the hash table
  void deleteElement(int key);
  // displaying the full hash table
  void displayHashTable();
};
//creating the hash table with the given table size
HashMapTable::HashMapTable(intts) {
  this -> table_size = ts;
  table = new list < int > [table_size];
}
// insert function to push the keys in hash table
void HashMapTable::insertElement(int key) {
  int index = hashFunction(key);
  table[index].push_back(key);
}
// delete function to delete the element from the hash table
void HashMapTable::deleteElement(int key) {
```

```
  int index = hashFunction(key);
  // finding the key at the computed index
  list < int > ::iterator i;
  for (i = table[index].begin(); i != table[index].end(); i++) {
    if ( * i == key)
      break;
  }
  // removing the key from hash table if found
  if (i != table[index].end())
    table[index].erase(i);
}
// display function to showcase the whole hash table
void HashMapTable::displayHashTable() {
  for (inti = 0; i < table_size; i++) {
    cout << i;
    // traversing at the recent/ current index
    for (auto j: table[i])
      cout << " ==> " << j;
    cout << endl;
  }
}
// Main function
intmain() {
  // array of all the keys to be inserted in hash table
  intarr[] = {
    20,
    34,
    56,
    54,
    76,
    87
  };
  int n = sizeof(arr) / sizeof(arr[0]);
  // table_size of hash table as 6
  HashMapTableht(6);
  for (inti = 0; i < n; i++)
    ht.insertElement(arr[i]);
  // deleting element 34 from the hash table
  ht.deleteElement(34);
  // displaying the final data of hash table
```

```
    ht.displayHashTable();
    return 0;
}
```

Output:



Revision #1
Created 6 May 2023 04:07:28 by Laptrinh.vn
Updated 27 June 2023 09:20:43 by Laptrinh.vn