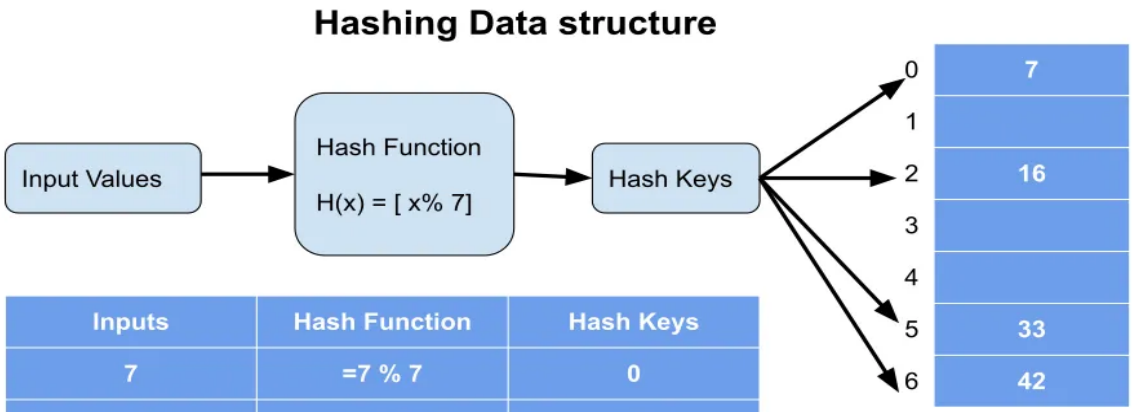
IT DS Assignment - 1

Submitted By ADITYA SINGH 2K19/EP/005

Q1. Explain hashing ?

Ans. Hashing is a technique used for storing and retrieving information as quickly as possible. It is used to perform optimal searches and is useful in implementing symbol tables.

Why Hashing ?
BSTs can support operations such as insert, delete and search in O(logn) time. In applications, if we need these operations in O(1), then hashing provides a way. Remember that the worst case complexity of hashing is still O(n), but it gives O(1) on the average.



Inputs	Hash Function	Hash Keys
7	=7 % 7	0
16	=16 % 7	2
33	=33 % 7	5
42	=42 % 7	6

Components of Hashing :-

a) **Hash Table**

It is a generalization of arrays. With an array, we store the element whose key is k at a position k of the array. That means, given a key k, we find the element whose key is k by just looking in the k th position of the array. This is called direct addressing. Hash table or hash map is a data structure that stores the keys and their associated values, and a hash table uses a hash function to map keys to their associated values.

b) Hash Function

The hash function is used to transform the key into the index. Ideally, the hash function should map each possible key to a unique slot index, but it is difficult to achieve in practice.

A good hash function should have the following characteristics:

- Minimize collision
- Be easy and quick to compute
- Distribute key values evenly in the hash table
- Use all the information provided in the key
- Have a high load factor for a given set of keys

How Hashing Gets O(1) Complexity ?

By using the load factor we make sure that each block (for example, linked list in separate chaining approach) on the average stores the maximum number of elements less than the load factor.

The access time of the table depends on the load factor which in turn depends on the hash function. This is because the hash function distributes the elements to the hash table. For this reason, we say a hash table gives O(1) complexity on average.

Implementation

```
class HashTable{
private:
  list<int> *table;
  int total_elements;

  // Hash function to calculate hash for a value:
  int getHash(int key){
    return key % total_elements;
  }

public:
  // Constructor to create a hash table with 'n' indices:
  HashTable(int n){
    total_elements = n;
    table = new list<int>[total_elements];
  }

  // Insert data in the hash table:
  void insertElement(int key){
    table[getHash(key)].push_back(key);
  }
```

```
// Remove data from the hash table:
void removeElement(int key){
  int x = getHash(key);

  list<int>::iterator i;
  for (i = table[x].begin(); i != table[x].end(); i++) {
    // Check if the iterator points to the required item:
    if (*i == key)
      break;
  }

  // If the item was found in the list, then delete it:
  if (i != table[x].end())
    table[x].erase(i);
}
```

Q2. Discuss collision resolution techniques?

Ans. The process of finding an alternate location is called collision resolution. Even though hash tables have collision problems, they are more efficient in many cases compared to all other data structures, like search trees. There are a number of collision resolution techniques, and the most popular are direct chaining and open addressing.

- Direct Chaining: An array of linked list application
  - Separate chaining
- Open Addressing: Array-based implementation
  - Linear probing (linear search)
  - Quadratic probing (nonlinear search)
  - Double hashing (use two hash functions)

Separate Chaining
Collision resolution by chaining combines linked representation with hash table. When two or more records hash to the same location, these records are constituted into a singly-linked list called a chain.

Open Addressing
In open addressing all keys are stored in the hash table itself. This approach is also known as closed hashing. This procedure is based on probing. A collision is resolved by probing.

Linear Probing	Quadratic Probing	Double hashing
Fastest among three	Easiest to implement and deploy	Makes more efficient use of memory
Uses few probes	Uses extra memory for links and it does not probe all locations in the table	Uses few probes but takes more time
A problem occurs known as primary clustering	A problem occurs known as secondary clustering	More complicated to implement
Interval between probes is fixed - often at 1.	Interval between probes increases proportional to the hash value	Interval between probes is computed by another hash function

Q3. What is a file? Explain different types of files?

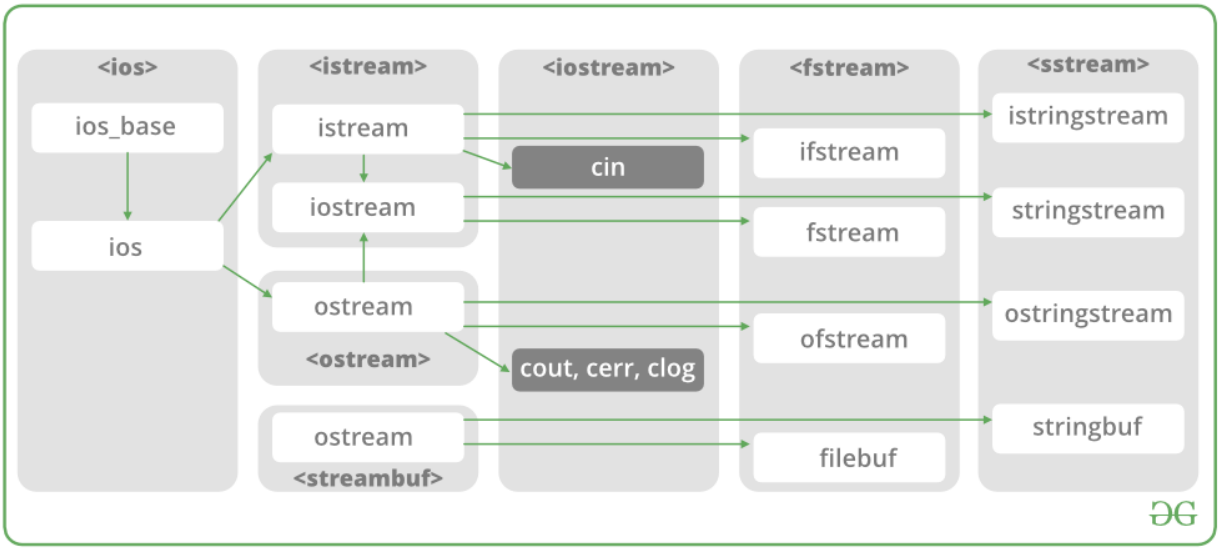
Ans. File is a collection of records related to each other. The file size is limited by the size of memory and storage medium. It is a computer resource for recording data in a computer storage device. Just as words can be written to paper, so can data be written to a computer file.
Files can be edited and transferred through the Internet on that particular computer system. A file may be designed to store an Image, a written message, a video, a computer program, or any wide variety of other kinds of data. Certain files can store multiple data types at once.

By using computer programs, a person can open, read, change, save, and close a computer file.

```
// Create a text string, which is used to output the text file
string myText;

// Read from the text file
ifstream MyReadFile("filename.txt");

// Use a while loop together with the getline() function to read the file line by line
while (getline (MyReadFile, myText)) {
  // Output the text from the file
  cout << myText;
}

// Close the file
MyReadFile.close();
```

Files are typically organized in a file system, which tracks file locations on the disk and enables user access.

Files in C++ - The fstream library allows us to work with files.
To create a file, use either the ofstream or fstream class, and specify the name of the file.
To write to the file, use the insertion operator (<<).

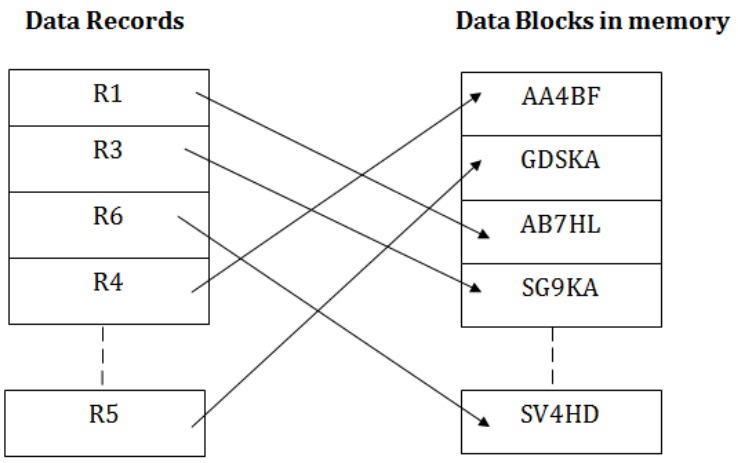


Types of Files

a) Ordinary Files or Simple File: any type of Application for example notepad, paint, C Program, Songs et, created by a user. It can store the information which contains text, database, any image or any other type of information.
b) Directory files: The Files that are Stored into a Particular Directory or Folder. For ex: a Folder Name Songs which Contains Many Songs So that all the Files of Songs are known as Directory Files.
c) Special Files: The Special Files are those which are not created by the user. Or The Files that are necessary to run a System. Means all the Files of an Operating System or Windows.
d) FIFO Files: The First in First Out Files are used by the System for Executing the Processes into Some Order. Means To Say the Files that Come first, will be Executed First and the System Maintains an Order or Sequence Order.

Q4. How to organize a file in data structure?

Ans. File organization ensures that records are available for processing. It is used to determine an efficient file organization for each base relation.



For example, if we want to retrieve employee records in alphabetical order of name. Sorting the file by employee name is a good file organization. However, if we want to retrieve all employees whose marks are in a certain range, a file ordered by employee name would not be a good file organization.

There are three types of organizing the file:

1. Sequential access file organization

Storing and sorting in contiguous blocks within files on tape or disk is called sequential access file organization. In sequential access file organization, all records are stored in a sequential order. The records are arranged in the ascending or descending order of a key field.

2. Direct access file organization

Direct access file is also known as random access or relative file organization.In a direct access file, all records are stored in a direct access storage device (DASD), such as a hard disk. The records are randomly placed throughout the file.
The records do not need to be in sequence because they are updated directly and rewritten back in the same location.

3. Indexed sequential access file organization

Indexed sequential access file combines both sequential file and direct access file organization. In an indexed sequential access file, records are stored randomly on a direct access device such as a magnetic disk by a primary key.
This file has multiple keys. These keys can be alphanumeric in which the records are ordered and are called primary keys.