PICT, PUNE

DSAL

shrikrushna Zirape
21286

Assignment- 05

problem Statement:-
Impliment function of the dictionary using hashing and handle collision using chaining with/without replacement.

Learning Objective:-
- impliment hashtable to impliment dictionary
- use chaining to reduce complexity.

Learning Outcome:-
- impliment a dictionary using hash table
- use different hash function & analyze the result.

Theory:-
hashing is a technique or process of mapping keys, value into a hash table using hashfunction the efficiency of mapping depends on that of fine hash function used.

Seprate chaining is the idea to make each cell of the hash table point to a linked list of record that same hash function value.

Advantage of chaining.

- Simple to Impliment.
- Hash Table is dynamic.

Disadvantage of chaining.

1) Wastage of storage space as some indices may never used.
2) Search Complexity will be O(n) for longer inputs.

Pseudo Code :-

```
Class Node
{ string keyword,
    meaning
  Node()
  { keyword = "";
    meaning = "";
    next = Null;
  }
  Node(string k, string m)
  { keyword = k
    meaning = m
    next = Null;
  }
  } friend class HashTable;
}

class HashTable
{ static const int size = 23;
  Node *ht[size];
  int len=0;
```

```
Algorithm hasfunction. (string key)
int k=0
for (int i=0 ; i < key.length() ; i++) {
k += key[i] }
return k%.size.

Algorithm insert (string key,
string mean).

Node * C = Null;
Search(key, &c)
if(C != Null ) {
  print (" Element already exist ")
  return
}
len++
int(index = hasfunction(key))
if( ht[index] == Null) do
{ Node * newNode = new node(key, mean)
  ht[index] = new node.;
  return
}

else
{
Node * temp = ht[index];
while( temp -> next != Null)
  {
  temp = temp -> next;
  }
```

```
        temp -> next = new Node (key, mean);
        }
    }
```

Algorithm Search (key, Node ** current)

```
{
int comp=0
index = hashFunction (key);
Node * temp = ht [index];
while ( temp != NULL) do
  {
    if (temp -> keyword) == key
      {
        current = temp;
        print (" Element found");
        return
      }
    comp++ ;
    temp = temp -> next ;
  }
}
```

Algorithm delete Node (string key)

```
{
  Node * c = Null;
  search (key, &c);
  if (c == Null) do {
  print (" Element not found ")
   return }
```

```
Node* temp= ht[index]
if (ht[index]-> keyword ==key)do
  {  ht[index] =temp -> next;
     delete(temp);
     return;
  }
while( temp -> next -> keyword != key)
  {
     temp = temp -> next;
  }
Node* n =temp -> next;
temp -> next = temp-> next -> next;
delete(n)
return
}

Algorithm display()
  {
    for( i =0 to i = size-1) do
    {
      if (ht[i] != NULL) do
      {
        Node* temp= ht[i];
        print ("Entries with index i");
        while (temp != NULL)
          { print (temp-> keyword
              and temp-> next);
            temp= temp-> next;
          } }
```

PICT, PUNE

```
eles { print("No entry for " i);
}}}
```

Test Cases:-

input.	Res output	Result
i) insert →		
dog:animal	9 cat animal	
cat:animal	10 mango fruit	Pass.
rose: flower	11 dog animal	
mango:fruit.		
delete →		
rose	"deleted"	
mouse.	"not found"	

Conclusion:-

Thus we have implimented dictionary using Hashtable in C++ ~~python~~ successfully.