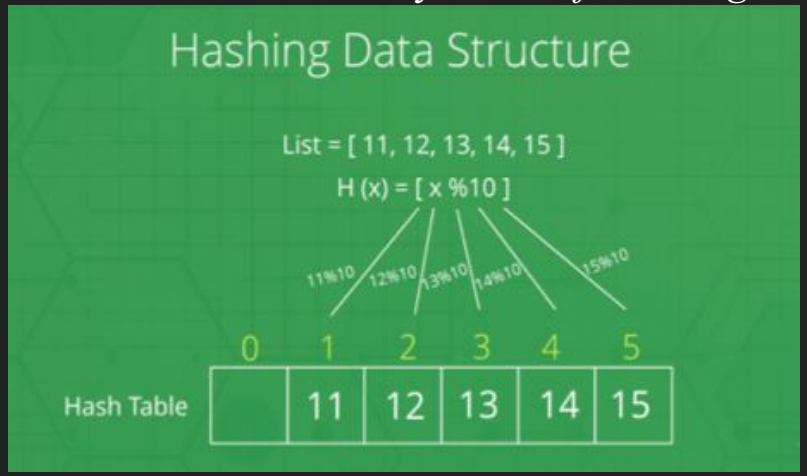
Hash Tables

Chrystal Mingo

What is a hash table?

- Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements.
- **Hashing** is the transformation of a string of characters into a usually *shorter fixed-length value or key* that represents the original string.



What is my Project?

- For my project I implemented a Hash Table which could do the following:
  - Create a hash
  - Add items to the hash table
  - Prints the number of indexes and the number of items in that location
  - Print the Hash Table
  - Print all the hash tables that fall under that one index
  - Find College (Search for Key)
  - Remove Items from Hash Table

In the end my project collected student names, and used their college as a key.

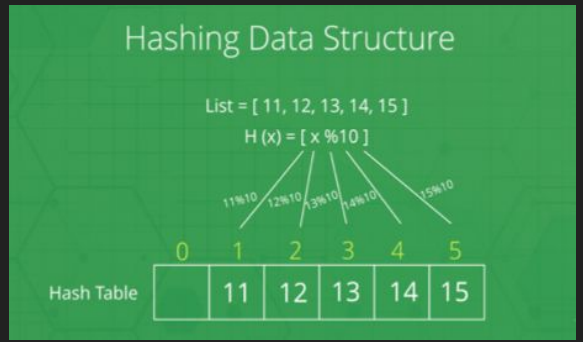
//Problem from: Paul’s Programming on Implementing Hash Tables Projects.

My implementation of Hash Tables:

```
int hash::Hash(string key){
    int hash = 0;
    int index;
    index = key.length(); //index is represented by the lenght of the string
    for (int i = 0; i< key.length(); i++){ //Adds up all the characters in the string
        hash = hash + (int)key[i];
    }
    //key[0]; would print the first letter of the string
    //now when you do (int)key[0 or any other index] you get the integer that represents that number
    index = hash % tableSize;
    //example if key total = 402 / 100(table size) = 4 r 2 and you store that remainder value as the index in the hash table
```

Located in hash.cpp

Process of Hashing: This function above is what hashes the strings. Taking the key, then its length. The for loop adds up all the values that represent the each char of the string to come up with a unique hash key (or index). If you look at the image in hashing they take the mod. Therefore, I took the hash and mod it by the size of the hash table, to find a index within range for the string.



My implementation of Hash Tables Continuation

```
void hash::AddItem(string name, string college){
    int index = Hash(name);//hold the location in the hash table that stores the int of that string
    if(HashTable[index]->name == "empty"){ //over writing
        HashTable[index]->name = name;
        HashTable[index]->college = college;

    }
    else {
        item* Ptr = HashTable[index]; //point to the begginning of that table
        item* n = new item; //Adding new item with name & college
        n->name = name;
        n->college = college;
        n->next = NULL; //make the next item point to NULL;
        while(Ptr->next != NULL){//will make the pointer transverse till the end of the list/hashTable
            Ptr = Ptr->next;
        }
        Ptr->next = n; //links the last item in that list with the new item in the hash table
    }
}
```

Located in hash.cpp

This function above adds Items to the Hash Table, which would include the person's name and college.

My implementation of Hash Tables:

```
int hash::NumberOfItemsInIndex(int index){
    int count = 0; //initialize the counter at 0
    if(HashTable[index]->name == "empty"){ //if we haven't put anything in then return the default
        return count;
    }
    else{
        count ++; //incrementing count
        item *Ptr = HashTable[index]; //point to beginning of the list
        while(Ptr->next != NULL){ //As long as the next pointer is not empty
            count ++;//increment as long as something is attached
            Ptr = Ptr->next;//continue to transverse to the next
        }
    }
```

Located in hash.cpp

Informs you of how much Items are in one Index of the Hash Table.

My implementation of Hash Tables:

```
void hash::PrintTable(){//Will Print out all the information in the HashTable
    int number; //hold the num of elements in each buckets
    for(int i =0; i < tableSize; i++){
        number = NumberOfItemsInIndex(i);
        cout <<"--------------------\n";
        cout<<"index = " << i << endl;
        cout<<HashTable[i]->name<<endl;
        cout<<HashTable[i]->college<<endl;
        cout<<"# of Items = "<<number<<endl;
        cout <<"--------------------\n";
    }
}
```

```
void hash::PrintItemsInIndex(int index){
    item* Ptr = HashTable[index]; //points to the first item in the bucket/hash table
    if(Ptr->name == "empty"){
        cout<<"index = "<< index << " is empty"<< endl;
    }
    else{
        cout <<"index = "<<index<< "contains the following items"<<endl;
        while(Ptr != NULL){
            cout <<"--------------------\n";
            cout<<Ptr->name<< endl;
            cout<<Ptr->college<<endl;
            cout <<"--------------------\n";
            Ptr = Ptr->next;
        }
    }
```

The first function prints all the elements in the hash table.

The second function prints out all the elements linked to a specific index of the hash table.

My implementation of Hash Tables:

```
void hash::FindCollege(string name){
    int index = Hash(name);
    bool foundName = false; //Name not found
    string college;
    item* Ptr = HashTable[index]; //pointing to the first item in the bucket
    while (Ptr!= NULL){
        if(Ptr->name == name){
            foundName = true;
            college = Ptr->college;
        }
        Ptr = Ptr->next;
    }
    if(foundName == true){
        cout<<"College = "<<college<<endl;
    }
    else{
        cout<< name <<" not found in the hash table"<<endl;
    }

}
```

This function, FindCollege searches for the key, and displays all the items that fall under that key.

Why did I choose this project? Was it Worth it?

I decided to implement the Hash Tables to learn about how hashing works, how to create a hash, display the table with information, add information, just learn overall. It was definitely worth it, because it was quite similar to linked list and served as a refresher. Also it was cool to see what different application can be done to the program, at first it was informing one of a person's name and favorite drink, then I changed it to see students names and their colleges. I also feel like I learned so much from implementing, and overall feel like a stronger programmer in c++.

Why did you use specifically use this data structure for the problem? Would a different data structure work better?

I specifically used the Hash table because it could allow me to store and search for items efficiently. I also wanted learn about how to implement hash tables, and how it works to hash the key. I could’ve used another form of hashing by using maps. I implemented a Hash Table, but I think graphs could works as well to showcase friends, but connect them as students to the same college.

What other task would work well suited for hash tables?

- Contact List
- Cryptography
- Message Digest
- Hash values or codes
- Blockchain
- Tables
- Password Verification

What was my favorite part and challenges?

My favorite part of this project, is honestly the fact that I learned so much about hashing, feel like a better programmer, just over all was worth the sleepless night. My comments help me understand better how the code works, doing it by myself I was able to stop and take my time to analyze what each function was doing. I was going to do ransom notes and take the easy way out, but implementing this from scratch I learned so much more about databases and the structuring over all. I actually understand, which makes me happy, and feel accomplished.

Citation:

https://www.geeksforgeeks.org/hashing-data-structure/

https://www.geeksforgeeks.org/applications-of-hashing/