Hash Table

Aim :

To implement hash table using C++.

Algorithm :

1. Start.
2. Create a class self-pointer and an integer variable to hold data.
3. In the main function, create an array of class of size defined through pre-processor or by user input.
4. Create a function that holds the hash key and returns the position where the element should be inserted or can be found.
5. Using a menu driven program, perform insertion, deletion, searching and printing on the hash table.
6. For every function called, the function that holds the hash function is invoked to find the position to be operated. The rest of the operation is continued as of a linear linked list.
7. End.

Program :

```
#include <iostream>
using namespace std;

#define max 10

class table
{
public:
    int data;
    table *next;
};

int hash_key(int x) // hashing key -> returns the position.
{
    return x % max; // hash function
}

void insert(table *hash[], int element)
{
    int i = hash_key(element);
    table *temp = hash[i];
    table *new_element = new table();
    new_element->data = element;
    new_element->next = NULL;
    if (hash[i] == NULL)
    {
        hash[i] = new_element;
```

```
        return;
    }
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = new_element;
}

bool search(table *hash[], int x)
{
    int i = hash_key(x);
    table *temp = hash[i];
    while (temp != NULL)
    {
        if (temp->data == x)
        {
            return true;
        }
        else
        {
            temp = temp->next;
        }
    }
    return false;
}

void del(table *hash[], int element)
{
    int i = hash_key(element);
    table *temp = hash[i];
    table *prev = hash[i];
    if (hash[i]->data == element)
    {
        if (hash[i]->next == NULL)
        {
            hash[i] = NULL;
        }
        else
        {
            hash[i] = hash[i]->next;
        }
        return;
    }
    while (temp->data != element && temp->next != NULL)
    {
        prev = temp;
        temp = temp->next;
    }
    if (temp->data == element)
    {
        prev->next = temp->next;
        delete temp;
        return;
```

```
    }
}

void display(table *hash[])
{
    for (int i = 0; i < max; i++)
    {
        table *temp = hash[i];
        if (temp == NULL)
        {
            cout << "0" << endl;
        }
        else
        {
            while (temp != NULL)
            {
                cout << temp->data;
                if (temp->next != NULL)
                {
                    cout << " -> ";
                }
                temp = temp->next;
            }
            cout << endl;
        }
    }
}

int main()
{
    table *hash[max] = {NULL};
    int choice;
    cout << "1. Insert" << endl;
    cout << "2. Search" << endl;
    cout << "3. Delete" << endl;
    cout << "4. Display" << endl;
    cout << "5. Exit" << endl;
    while (true)
    {
        cout << "Enter your choice : ";
        cin >> choice;
        if (choice == 1)
        {
            int element;
            cout << "Enter the element to be inserted : ";
            cin >> element;
            insert(hash, element);
        }
        else if (choice == 2)
        {
            int element;
            cout << "Enter the element to be searched : ";
            cin >> element;
            if (search(hash, element))
```

```
            {
                cout << "Element Found !" << endl;
            }
            else
            {
                cout << "Element not Found !" << endl;
            }
        }
        else if (choice == 3)
        {
            int element;
            cout << "Enter the element to be deleted  : ";
            cin >> element;
            if (search(hash, element))
            {
                del(hash, element);
            }
            else
            {
                cout << "Element not found !" << endl;
            }
        }
        else if (choice == 4)
        {
            display(hash);
        }
    }
}
```

Output :

```
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice : 1
Enter the element to be inserted : 1
Enter your choice : 1
Enter the element to be inserted : 11
Enter your choice : 1
Enter the element to be inserted : 21
Enter your choice : 1
Enter the element to be inserted : 5
Enter your choice : 1
Enter the element to be inserted : 25
Enter your choice : 1
Enter the element to be inserted : 45
Enter your choice : 1
Enter the element to be inserted : 66
Enter your choice : 1
Enter the element to be inserted : 6
Enter your choice : 1
Enter the element to be inserted : 63
Enter your choice : 1
Enter the element to be inserted : 3
Enter your choice : 1
Enter the element to be inserted : 78
Enter your choice : 1
Enter the element to be inserted : 89
Enter your choice : 4
0
1 -> 11 -> 21
0
63 -> 3
0
5 -> 25 -> 45
66 -> 6
0
78
89
Enter your choice : 2
Enter the element to be searched : 100
Element not Found !
Enter your choice : 2
Enter the element to be searched : 45
Element Found !
Enter your choice : 2
Enter the element to be searched : 0
Element not Found !
Enter your choice : 3
Enter the element to be deleted  : 25
Enter your choice : 3
Enter the element to be deleted  : 63
Enter your choice : 3
Enter the element to be deleted  : 27
Element not found !
Enter your choice : 4
0
1 -> 11 -> 21
0
3
0
5 -> 45
66 -> 6
0
78
89
Enter your choice : 3
Enter the element to be deleted  : 89
Enter your choice : 4
0
1 -> 11 -> 21
0
3
0
5 -> 45
66 -> 6
0
78
0
```