

AIM:**HASH TABLE IMPLEMENTATION**

Implement a hash table data structure using different hash function and collision resolution techniques such as chaining and open addressing.

PROGRAM:

```
#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

// Node for chaining
struct Node {
    int data;
    struct Node* next;
};

struct Node* chainTable[SIZE]; // Hash table for chaining
int openAddressingTable[SIZE]; // Hash table for open addressing

// Hash function
int hashFunction(int value) {
    return value % SIZE;
}

// Chaining
void insertChaining(int value) {
    int index = hashFunction(value);
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = chainTable[index];
    chainTable[index] = newNode;
}
```

```

void displayChaining() {
    for (int i = 0; i < SIZE; i++) {
        struct Node* temp = chainTable[i];
        printf("Index %d: ", i);
        while (temp) {
            printf("%d -> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

```

// Linear Probing

```

void insertLinearProbing(int value) {
    int index = hashFunction(value);
    while (openAddressingTable[index] != 0) {
        index = (index + 1) % SIZE;
    }
    openAddressingTable[index] = value;
}

```

```

void displayLinearProbing() {
    for (int i = 0; i < SIZE; i++) {
        printf("Index %d: %d\n", i, openAddressingTable[i]);
    }
}

```

// Quadratic Probing

```

void insertQuadraticProbing(int value) {
    int index = hashFunction(value);

```

```

for (int i = 0; i < SIZE; i++) {
    int newIndex = (index + i * i) % SIZE;
    if (openAddressingTable[newIndex] == 0) {
        openAddressingTable[newIndex] = value;
        return;
    }
}

// Double Hashing
int secondHashFunction(int value) {
    return 7 - (value % 7); // Secondary hash function
}

void insertDoubleHashing(int value) {
    int index = hashFunction(value);
    int stepSize = secondHashFunction(value);
    while (openAddressingTable[index] != 0) {
        index = (index + stepSize) % SIZE;
    }
    openAddressingTable[index] = value;
}

// Main function
int main() {
    // Chaining
    printf("Chaining:\n");
    insertChaining(10);
    insertChaining(20);
    insertChaining(30);
    insertChaining(42);
}

```

```
displayChaining();

// Open Addressing
printf("\nLinear Probing:\n");
for (int i = 0; i < SIZE; i++) {
    openAddressingTable[i] = 0; // Initialize the table
}
insertLinearProbing(10);
insertLinearProbing(21);
insertLinearProbing(30);
insertLinearProbing(46);
displayLinearProbing();

// Quadratic Probing
printf("\nQuadratic Probing:\n");
for (int i = 0; i < SIZE; i++) {
    openAddressingTable[i] = 0; // Initialize the table
}
insertQuadraticProbing(12);
insertQuadraticProbing(24);
insertQuadraticProbing(34);
insertQuadraticProbing(45);
displayLinearProbing(); // Reusing display function

// Double Hashing
printf("\nDouble Hashing:\n");
for (int i = 0; i < SIZE; i++) {
    openAddressingTable[i] = 0; // Initialize the table
}
insertDoubleHashing(10);
insertDoubleHashing(22);
```

```

insertDoubleHashing(32);

insertDoubleHashing(45);

displayLinearProbing(); // Reusing display function

return 0;
}

```

OUTPUT

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Shreyash\OneDrive\Desktop\OOP_Practice> cd "c:\Users\Shreyash\OneDrive\Desktop\OOP_Practice"
PS C:\Users\Shreyash\OneDrive\Desktop\OOP_Practice> cd "c:\Users\Shreyash\OneDrive\Desktop\OOP_Practice\" ; if ($?) { g++ Practical09.cpp -o Practical09 } ; if ($?) { .\Practical09 }
Chaining:
Index 0: 30 -> 20 -> 10 -> NULL
Index 1: NULL
Index 2: 42 -> NULL
Index 3: NULL
Index 4: NULL
Index 5: NULL
Index 6: NULL
Index 7: NULL
Index 8: NULL
Index 9: NULL

Linear Probing:
Index 0: 10
Index 1: 21
Index 2: 30
Index 3: 0
Index 4: 0
Index 5: 0
Index 6: 46
Index 7: 0
Index 8: 0
Index 9: 0

Quadratic Probing:
Index 0: 0
Index 1: 0
Index 2: 12
Index 3: 0
Index 4: 24
Index 5: 34
Index 6: 45
Index 7: 0
Index 8: 0
Index 9: 0

Double Hashing:
Index 0: 10
Index 1: 0
Index 2: 22
Index 3: 0
Index 4: 0
Index 5: 32
Index 6: 0
Index 7: 0
Index 8: 0
Index 9: 45
PS C:\Users\Shreyash\OneDrive\Desktop\OOP_Practice>

```

GITHUB LINK : <https://github.com/ShreyashGajbhiye453/Data-Structure-Practical-No.-01>