430.217
Introduction to Data Structures

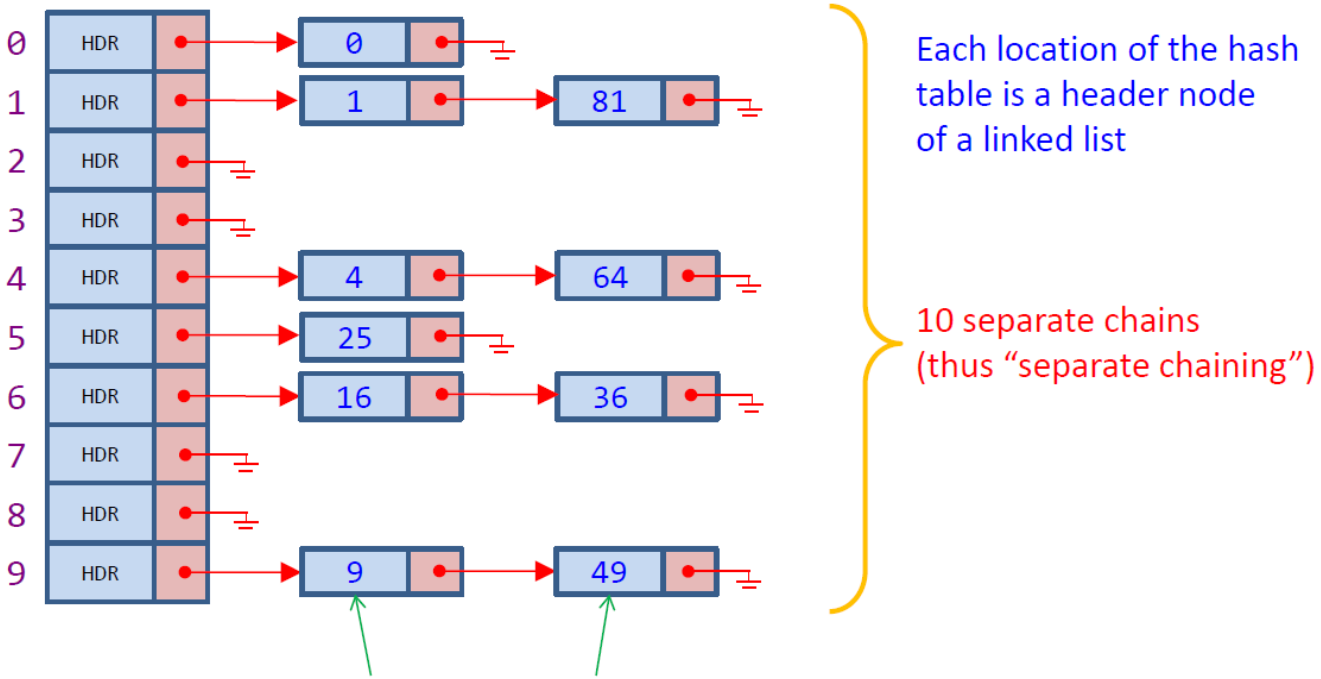
Assignment 4. Hashing

Seoul National University
Advanced Computing Laboratory

ACL ADVANCED COMPUTING LABORATORY

1. Separate chaining

- Separate chaining (Open hashing)
  - Keeps a separate linked list for each bucket
    - Example
      - $B = 10, h(x) = x \% B$



Implementation of separate chaining

- LinkedList.h의 Node class 사용할 것
  - string str
    - 사용하지 말 것
  - int data
    - Key value를 저장
- Operations
  - int Hash_function(int key)
  - void insert(int key)
  - void delete_element(int key)
  - bool find(int key)
- Member variable
  - Node** hash_table
    - Node* array를 가리키는 포인터

**header file 수정 시 주석 달것*

```
class Node
{
public:
    string str;
    int data;
    Node* next;

    Node(string str, int data, Node* ptr = NULL);
    void print();
};
#include <iostream>
#include <string>
#include "LinkedList.h"

using namespace std;

class SHash
{
public:
     SHash();
     ~SHash();
      int Hash_function(int key);
      void insert(int key);
      void delete_element(int key);
      bool find(int key);
private:
      Node** hash_table;
};
```

Separate_chaining.h 예제

Member functions

- int Hash_function(int key)

```
int Hash_function(int key);
```

  - Key 를 hash table size로 나눈 나머지 값을 반환

- void insert(int key)

```
void insert(int key);
```

  - Hash function에 의해 결정된 hash table의 위치에 key를 삽입
  - 해당 위치에 element가 존재한다면, 기존 element 뒤에 삽입

  (Linked list의 pushback과 기능 동일)

- void delete_element(int key)

```
void delete_element(int key);
```

  - Hash table에서 key값을 가지는 element를 삭제

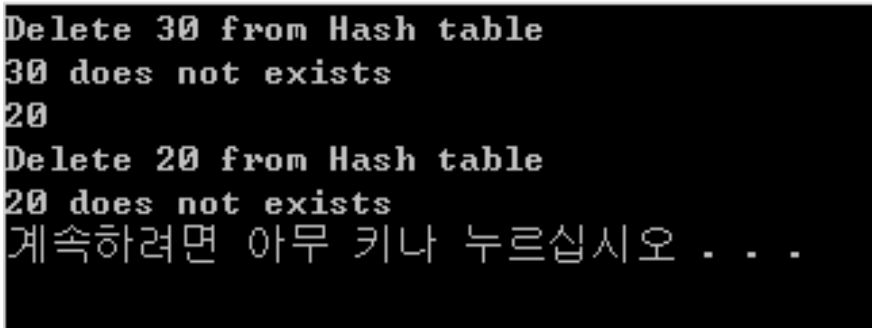
- bool find(int key)

```
bool find(int key);
```

  - Hash table에서 key 값을 가지는 element의 유무 판별, True/False 반환

Separate chaining - Practice main()

```
int main()
{
    SHash shash_test;

    shash_test.insert(10);
    shash_test.insert(11);
    shash_test.insert(20);
    shash_test.insert(30);

    shash_test.delete_element(30);
    shash_test.delete_element(30);

    shash_test.insert(12);
    shash_test.insert(22);

    shash_test.find(20);
    shash_test.delete_element(20);
    shash_test.find(20);

    return 0;
}
```



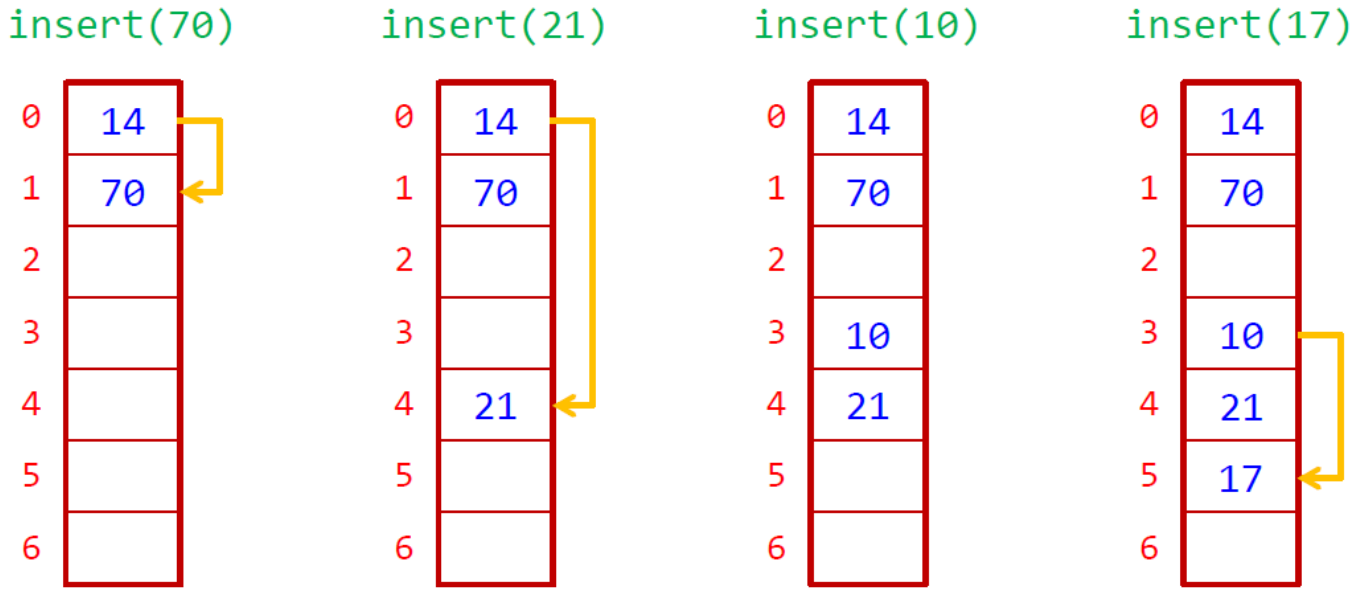
Separate_chaining 실행 예
(Table size 10으로 설정)

2. Quadratic probing

- Quadratic probing
  - Follows a probing sequence

$$h_i(x) = [h(x) + f(i)] \,\%\, B \text{ , where } f(i) = i^2$$

- Example
  - $B = 7, h(x) = x \,\%\, B, h_i(x) = (h(x) + i^2) \,\%\, B$



Implementation of quadratic probing

- Operations
  - int Hash_function(int key)
  - void insert(int key)
  - void delete_element(int key)
  - int find(int key)

- Member variable
  - int* hash_table
    - Hash table을 가리키는 포인터

```
class QHash
{
public:
    QHash();
    ~QHash();
     int Hash_function(int key);
     void insert(int key);
     void delete_element(int key);
     int find(int key);

private:
    int* hash_table;
};
```

Member functions

- int Hash_function(int key)

```
int Hash_function(int key);
```

  - Hash value를 반환
    - collision 없는 경우 Key 를 hash table size로 나눈 나머지 값을 반환
    - Collision 발생한 경우 아래 식을 통하여 새로운 hash 값 반환

$$h_i(x) = [h(x) + i^2] \% B$$

- void insert(int key)

```
void insert(int key);
```

  - Hash function에 의해 결정된 hash table의 위치에 key를 삽입
- void delete_element(int key)

```
void delete_element(int key);
```

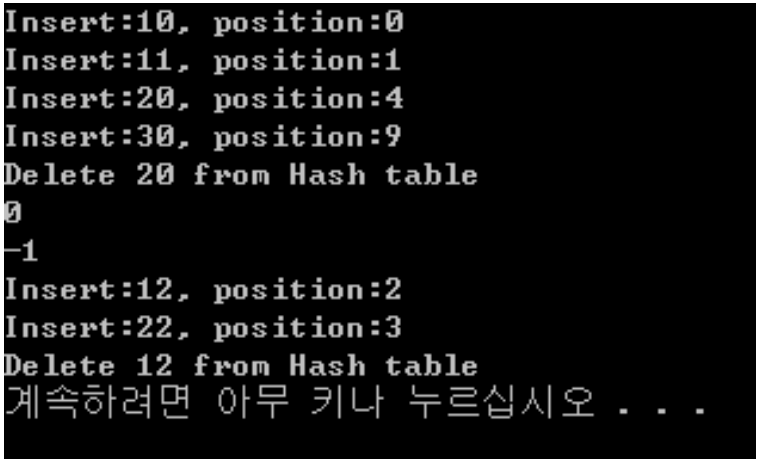
  - Hash table에서 key값을 가지는 element를 삭제 (NULL로 대체)
- int find(int key)

```
int find(int key);
```

  - Hash table에서 key 값을 가지는 element의 유무 판별
    - Element가 위치한 array의 index 반환
    - 값이 없으면 -1 반환

Quadratic probing - Practice main()

```
int main()
{
    QHash qhash_test;
    qhash_test.insert(10);
    qhash_test.insert(11);
    qhash_test.insert(20);
    qhash_test.insert(30);

    qhash_test.delete_element(20);
    cout<<qhash_test.find(10)<<endl;
    cout<<qhash_test.find(20)<<endl;

    qhash_test.insert(12);
    qhash_test.insert(22);
    qhash_test.delete_element(12);

    return 0;
}
```



Quadratic_probing 실행 예
(Table size 10으로 설정)

제출

- 문의 사항은 Q&A 이메일을 적극적으로 활용할 것 (snu.qna.ds@gmail.com)
- 코드 주석은 필수 - 부분점수
- 제출
  - ETL에서 assignment 4 에 제출할 것
  - Due: 11/22(일) 23:59
  - 위 assignment를 구현한 Visual studio 프로젝트 폴더자체를 압축하여 제출
  - 압축파일 형식(학번.zip): 2015-12345.zip