

**자료 구조 과제 6**

**과목명 자료구조**

**담당교수 김희철교수님**

**제출일 20210606**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

* **문제 1-1 번**

사전을 관리하는 프로그램을 작성하시오.

요구조건: 사전은 해싱으로 구현한다.

open addressing을 사용하고 충돌해결방법으로 이중해싱(double hashing)을 사용

명령어 종류

(1) 단어삽입 단어 한글뜻 // 단어와 뜻을 사전에 추가

// 단어가 사전에 있을 경우 error1 출력

(2) 단어검색 단어 // 사전에서 단어를 검색하여 뜻을 출력

// 단어가 사전에 없을 경우 error2 출력

(3) 단어뜻수정 단어 수정된한글뜻 // 사전에서 단어의 뜻을 수정

// 단어가 사전에 없을 경우 error2 출력

(4) 종료 // 종료

입력 예 출력결과

단어삽입 time 시간

단어삽입 bee 벌

단어삽입 zoo 동물원

단어삽입 school 학기

단어삽입 trade 무역

단어삽입 farm 농장

단어삽입 zoo 동물원 error1

단어뜻수정 schol 학교 error2

단어뜻수정 school 학교

단어검색 school 학교

종료

* **알고리즘 및 자료구조**

**Class DoubleHashing**

* Init 생성자 1. self.M 개수 2. self.hash\_table 해시 키 리스트 3. self.data key관련 data 저장 4. self.N 항목수
* Getkey 함수: 키의 [0]알파벳의 아스키코드값을 받음
* Hash\_func 함수: 키를 해시의 총 크기로 나눠 나머지를 구함
* getAddress 함수: getkey함수에서 키값을 받아와서 size를 나눠 나머지를 구하는 함수
* put 함수: 단어 삽입함수 충돌 시 이중해시 방법으로 7 – (self.getkey(key) % 7)사용
* get 함수: 단어 검색 함수는 key값을 입력받으면 해당되는 key의 data를 반환함
* correct함수: 단어 뜻 수정 함수는 입력받은 키값이 일치할 경우 입력받은 data로 수정함
* print\_table: 사전의 모든 key값과 data값 출력(사전 잘 작동 되는지 확인용)
* **느낀 점**

Open addressing 자료구조를 사전에 단어를 입력, 삽입, 수정, 검색함수를 구현해 봄으로써 충돌 시 이중 해싱의 작동원리를 이해할 수 있었습니다.

* **프로그램 코드**
* # 1.제목:한국외대 자료구조 과제 6(6-1-1)  
  # 2.날짜:20210605  
  # 사전 해싱  
  class DoubleHashing:  
   def \_\_init\_\_(self, size):  
   self.M = size  
   self.hash\_table = [None for x in range(size+1)] # hashtable  
   self.data = [None for x in range(size+1)] # key관련 data저장  
   self.N = 0 # 항목수  
    
   def getkey(self, key): # 키를 [0]알파벳의 아스키코드값으로 받음  
   self.key = ord(key[0])  
   return self.key  
    
   def hash\_func(self, key):  
   return key % self.M  
    
   def getAddress(self, key): # 해시값을 구함 인덱스 번호  
   myKey = self.getkey(key)  
   hash\_address = self.hash\_func(myKey)  
   return hash\_address  
    
   def put(self, key, data): # 단어 삽입 함수  
   init\_num = self.getAddress(key) # 초기 위치  
   i = init\_num # 초기 인덱스  
   second\_i = 7 - (self.getkey(key) % 7) # hash 2  
   j = 0  
   while True:  
   if self.hash\_table[i] == None: # 삽입 위치 찾기  
   self.hash\_table[i] = key # key를 해쉬 테이블에 저장  
   self.data[i] = data # key값이 데이터를 동일한 인덱스에 저장  
   self.N += 1  
   return True  
   elif self.hash\_table[i] != None: # 해시값에 data가 있는경우  
   if self.hash\_table[i] == key: # 입력받은 키와 기존의 키가 같으면  
   return print("error1") # error1출력  
   if self.hash\_table[i] == key:  
   self.data[i] = data # data 갱신  
   return True  
   j += 1  
   i = (init\_num + j\*second\_i) % self.M # 충돌시 이중해쉬 하는 부분  
   if self.N > self.M: # 테이블이 꽉찰 경우  
   break  
   return print("error1")  
    
   def get(self, key): # 검색  
   init\_num = self.getAddress(key)  
   i = init\_num  
   second\_i = 7 - (self.getkey(key) % 7)  
   j = 0  
   while self.hash\_table[i] != None:  
   if self.hash\_table[i] == key: # 사전에서 단어를 검색하여  
   return self.data[i] # 뜻을 출력  
   j += 1  
   i = (init\_num + j\*second\_i) % self.M  
   return print("error2") # 단어가 사전에 없을 경우 error2 출력  
    
   def correct(self, key, data): # 당어 뜻 수정 함수  
   init\_num = self.getAddress(key)  
   i = init\_num  
   second\_i = 7 - (self.getkey(key) % 7)  
   j = 0  
   while self.hash\_table[i] != None:  
   if self.hash\_table[i] == key: # key가 일치할경우   
   self.data[i] = data # 입력받은 data로 수정  
   return  
   j += 1  
   i = (init\_num + j \* second\_i) % self.M  
   return print("error2") # 단어가 사전에 없을 경우 error2 출력  
    
   def print\_table(self): # 사전 출력 ( 쓰이진 않음)  
   for i in range(self.M):  
   if self.hash\_table[i] == None:  
   pass  
   else:  
   print(str(self.hash\_table[i]) + str(self.data[i]))  
    
    
    
  h = DoubleHashing(13)  
  while True:  
   command = input().split()  
   #print(command)  
   if command[0] == '단어삽입':  
   h.put(command[1], command[2])  
   elif command[0] == '단어뜻수정':  
   h.correct(command[1], command[2])  
   elif command[0] == '단어검색':  
   print(h.get(command[1]))  
   elif command[0] == 'p': # 사전 확인용  
   h.print\_table()  
   elif command[0] == '종료': # 종료  
   break