

Data structure: Assignment 1

Seung-Hoon Na

October 1, 2018

1 Assignment 1

1.1 Binary search

주어진 정렬된 입력 파일이 있다고 가정하자. 단, 파일내의 숫자는 공백으로 구분, file내에 숫자들은 multiline으로 구성될 수 있으며, 한 라인에는 임의의 갯수의 숫자가 순서대로 나열될 수 있다. 숫자는 정수라고 가정한다.

입력파일의 예는 다음과 같다 (input.txt).

```
10 11 12 16 18 23 29
33 48 54 57 68 77 84 98
```

위의 입력 파일을 읽어들이어 사용자로부터 숫자를 입력받아 해당 숫자가 몇 번째 위치에 있는지를 출력하는 **이진탐색 (binary search)** 프로그램을 재귀형과 반복형 함수로 각각 작성하시오. (binarysearch.recurisve.cpp와 binarysearch.iterative.cpp 작성 제출)

각 코드에는 아래 BinarySearch함수와 사용자 입력을 받아 이진탐색을 수행하는 main()함수가 포함되어야 한다.

```
int BinarySearch(int A[], int start, int end, int key);
```

- 위의 샘플 input.txt 파일외에 다른 입력 파일에도 동작될 수 있도록 작성되어야 함.

1.2 k -ary search

문제 1.1를 일반화하여 주어진 데이터를 k 의 파트로 나누고 이들 중 하나로 계속 탐색해나가는 k -ary search ($k \geq 2$) 프로그램을 재귀형 함수로 작성하시오 (karysearch.cpp 작성 제출)

마찬가지로, karysearch.cpp 에는 아래 KarySearch함수와 사용자 입력을 받아 이진탐색을 수행하는 main()함수가 포함되어야 한다.

```
int KarySearch(int A[], int k, int start, int end, int key);
```

1.3 이중 연결 리스트 구현

1.3.1 이중 연결 리스트 구현: 정수형 버전

이중 연결리스트 (doubly-linked list)를 위한 노드 구조는 다음과 같다.

```
typedef struct nodeRecord
{
    int Data;
    struct nodeRecord *Prev, *Next;
} node;
```

이렇게 확장된 노드 구조가 반영된 이중 연결리스트를 위한 인터페이스 파일 (DoublyListP.h)의 일부는 다음과 같다.

```
typedef struct nodeRecord
{
    int Data;
    struct nodeRecord *Prev, *Next;
} node;

typedef node* Nptr;
class listClass
{
public:
    listClass();
    listClass(const listClass &L);
    ~listClass();
    void Insert(int Position, int Item);
    void Delete(int Position);
    void Retrieve(int Position, int & item);
    bool isEmpty ();
    int Length();
private:
    int Count;
    Nptr Head;
}
```

위의 인터페이스 파일을 완성하고, 이를 구현한 DoublyListP.cpp를 작성하시오. 이들 리스트 함수를 다양하게 호출하는 Test코드 DoublyListTest.cpp 를 작성하시오.

1.3.2 이중 연결 리스트 구현: template버전

위의 이중연결리스트는 int 정수형 데이터에 대해서 동작되는 연결리스트이다. 임의의 데이터에 대해서 구동되도록 C++ template을 이용하여 이중연결리스트를 확장한 인터페이스 파일 (GenericDoublyListP.h)과 구현 파일(GenericDoublyListP.cpp)을 작성하고, 문자열 (string), 정수형 (int), 실수형 (double) 에 구동되는 테스트 파일 (GenericDoublyListPTest.cpp)을 작성하시오. (물론, 리스트와 관련된 STL (standard template library)은 사용하지 말것)

C++ Template관련한 자료로 다음을 참조하시오.

<http://www.cs.bham.ac.uk/hxt/2016/c-plus-plus/intro-to-templates.pdf>

<http://www.cplusplus.com/doc/oldtutorial/templates/>

<http://users.cis.fiu.edu/weiss/Deltoid/vcstl/templates>

Sample test코드의 예는 다음과 같다.

```
void main(){
```

```
listClass<string> str_list;
listClass<int> int_list;
listClass<double> double_list;

// test 수행
}
```

1.3.3 이중 연결 리스트 구현 – java버전

리스트의 ADT의 규약을 준수하면서, 위의 이중 연결 리스트에 대한 Java 코드를 작성하시오 (DoublyLinkedList.java). main함수에서는 이중 연결리스트의 각 함수를 test하는 코드가 포함되어야 한다. (java버전은 template으로 작성될 필요는 없음)

1.3.4 스택과 큐 구현: template 이중 연결 리스트 기반

앞서 구현한 **template 이중연결리스트 class listClass**를 이용하여 교과서 코드 [6-9]와 코드 [7-6]처럼 스택 class stackClass (genericStackDL.h/genericStackDL.cpp)와 큐 class queueClass (genericQueueDL.h/genericQueueDL.cpp)를 완성하시오.

스택과 큐의 구동 및 예외를 테스트하는 테스트코드 stackClassTest.cpp와 queueClassTest.cpp도 함께 작성하시오.)문자열 (string), 정수형 (int), 실수형 (double) 각각에 대해서 동작이 확인되어야 한다)

1.4 산술식 평가: calculator구현

앞서 구현한 generic stackClass를 이용하여 **산술식을 계산하는 프로그램의 pseudo-code와 c++ 코드 calculator.cpp**을 작성하시오.

피연산자를 저장하는 stackClass<double>과 연산자를 저장하는 stackClass<string>를 이용하면 된다.

입력 token은 white space로 구분되며, 피연산자는 실수형, 연산자는 사칙연산인 *, +, -, /외에 log, sqrt, pow함수로 추가로 주어지고, 계산의 우선순위를 위해 열린 괄호와 닫힌 괄호 (,)가 하나의 token으로 주어진다. argument 인자간의 구분을 위해 ,이 사용된다.

다음은 입력의 예이다.

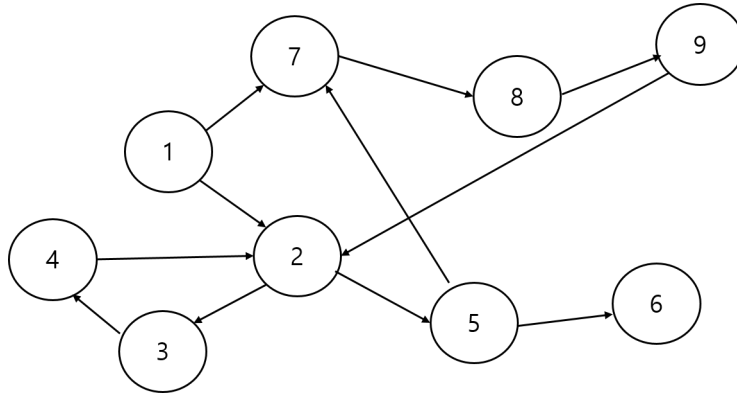
```
( 1 + ( ( 5 + 8 ) * ( 10 + 7.5 ) ) )
( ( 2 + pow ( 5.0 , 3.0 ) + log ( 7.0 ) ) / 3.0 )
( ( 2 + sqrt ( 10.0 ) ) * ( log ( 2.0 ) + 5 ) )
```

또한, 다음과 같이 문법에 맞지 않은 식이 입력되면 적절한 message로 **syntax error**를 출력해야 한다.

```
( 5 + 6
pow ( 6 7 ) + log ( 5 )
( 4 + 2 ) * 10 )
```

1.5 깊이 우선 탐색: depth-first search 구현

스택을 이용하여 그래프 파일을 로딩하여, 그래프 구조를 얻은후 **사용자로부터 시작점과 끝점을 입력받아, 깊이 우선 탐색**을 수행하여 경로를 찾는 알고리즘을 구현 (DFS.cpp) 하시오. 스택은 앞서 구현한 `template stackClass<int>`를 이용하라.



그래프 파일은 인접리스트방식을 사용하며, 위 그래프에서 그래프 입력 파일 (sample_graph.txt)은 다음과 같다.

```
1 2 7
2 3 5
3 4
4 2
5 6 7
7 8
8 9
9 2
```

가령, 첫번째 line의 경우 1번 노드는 2번과 7번 노드에 인접해 있다는 것을 의미한다. 즉, 1번 노드로부터 outgoing하는 edge는 $\{(1, 2), (1, 7)\}$ 의 두개의 edge로 구성되어 있다.

깊이 우선 탐색의 pseudo code는 다음과 같다.

Algorithm 1 Depth first algorithm

```
1: procedure DFS( $G = (V, E), v, w$ )           ▷  $v$  and  $w$  are starting and ending
   vertices
2:   create  $S$                                 ▷ create a new stack
3:    $S.Push(v)$                                 ▷ push  $v$  to the stack
4:   while  $S.IsEmpty() \neq 0$  do
5:      $u = S.Pop()$ 
6:     if  $u = w$  then                          ▷  $u$  is the ending vertex
7:       return a path from  $v$  to  $w$ 
8:     end if
9:     for  $u' : (u, u') \in E$  do
10:      if  $visited[u'] = 0$  then                ▷  $u'$  is not visited
11:         $visited[u'] = 1$                     ▷ mark it as visited
12:         $S.Push(u')$                         ▷ and add it to the stack
13:      end if
14:    end for
15:  end while
16:  return NotFound                          ▷ there is no path
17: end procedure
```

위의 pseudo code 17번 라인에서 경로를 출력하기 위해 필요한 절차 및 자료 구조는 pseudo code에 포함되어 있지 않으므로 이를 구상하여 코드를 완성해야 함

다음 프로그램 구동 시나리오이다.

```
$ ./dfs sample_graph.txt
enter staring and ending vertices> 1 6
path length: 3
1 2 5 6
enter staring and ending vertices> 5 1
not found
enter staring and ending vertices> 1 9
path length: 5
1 2 5 7 8 9
enter staring and ending vertices> ...
```

위의 입력 sample파일외에 다른 sample graph들에서 2개 이상 만들어 테스트 할 것.

1.6 제출 내용 및 평가 방식

코드는 c++ (특별한 요구사항이 있을 경우 java)로 작성하도록 하고, c++프로그램의 경우 구동os환경은 **ubuntu 16.04 LTS** 이상을 원칙으로 한다. 본 과제 결과물로 필수적으로 제출해야 내용들은 다음과 같다.

- **코드 전체**
- **테스트 결과:** 각 내용별 테스트 코드 및 해당 로그 파일
- **결과보고서:** 코드 설계(클래스 계층도 등), 구현 방법 및 실행 결과를 요약한 보고서

본 과제의 평가항목 및 배점은 다음과 같다.

- 각 세부내용의 구현 정확성 및 완결성 (80점)
- 코드의 Readability 및 체계성 (10점)
- 결과 보고서의 구체성 및 완결성 (10점)