Data Structures

Lab # 12

Data & Knowledge Engineering Laboratory
Kyung Hee University, Korea

Chapter 9 Programming Exercise

Lab 11

- 1. Exercise 15-18
- 2. Exercise 23
- 3. Exercise 25

실습 준비

- **샘플 코드 중에서 Chapter9에 있는 소스를 사용함**
  - ...₩labplus_CRLF₩labplus₩Lab, C++ 3rd₩Chapter9₩

- **샘플코드 수정 내역**
  - GraphType.cpp 파일에 문제 발생
  - GraphType.h에 선언된 클래스가 템플릿을 사용하는데도 선언부와 구현부가 분리되어 있음

- **사용 코드**
  - Graph : GraphType.h, QueType.h

1-Exercise 15-18

■ 문 제

◈연습문제 28-31에 나오는 · 방향 그래프는 다음과 같은 정의를 사용한다.

ZooGraph=(V,E)
V(ZooGraph)={dog, cat, animal, vertebrate, oyster, shellfish, invertebrate, crab, poodle, monkey, banana, Dalmatian, dachshund}
E(ZooGraph)={(vertebrate, animal), (invertebrate, animal), (dog, vertebrate), (cat, vertebrate), (cat, vertebrate), (monkey, vertebrate), (shellfish, invertebrate), (crab, shellfish), (oyster, invertebrate), (poodle, dog), (Dalmatian, dog), (dachshund, dog)}

28. ZooGraph의 그림을 그려라.

29. 인접행렬을 사용하여 구현된 ZooGraph을 그려라. 정점값은 알파벳 순서로 저장한다.

30. ZooGraph에 있는 어떤 원소가 또 다른 원소와 X라는 관계를 갖고 있다면 그 사이의 경로를 찾아보자. 인접행렬을 사용하여 다음의 내용이 참인지 보이도록 한다.
a. dalmatian X dog
b. dalmatian X vertebrate
c. dalmatian X poodle
d. banana X invertebrate
e. oyster X invertebrate
f. monkey X inverbrate

31. 앞의 질문에서 X에 해당하는 관계 중에서 가장 적합한 것은 무엇인가?
a. "has a"
b. "is an example of"
c. "is a generalization of"
d "eats"

■ 작성 방법

❖ 한글 또는 워드 문서로 제출

테스트 드라이버 (예제)

```
#include "GraphType.h"

#include <iostream>
using namespace std;

int main()
{
                GraphType<char*> graph;

                graph.AddVertex("dog");
                graph.AddVertex("cat");
                graph.AddVertex("animal");
                graph.AddVertex("vertebrate");
                graph.AddVertex("oyster");
                graph.AddVertex("shellfish");
                graph.AddVertex("invertebrate");
                graph.AddVertex("crab");
                graph.AddVertex("poodle");
                graph.AddVertex("monkey");
                graph.AddVertex("banana");
                graph.AddVertex("dalmatian");
                graph.AddVertex("dachshund");

                graph.AddEdge("vertebrate","animal",10);
                graph.AddEdge("invertebrate","animal",20);
                graph.AddEdge("dog","vertebrate",30);
                graph.AddEdge("cat","vertebrate",40);
                graph.AddEdge("monkey","vertebrate",50);
                graph.AddEdge("shellfish","invertebrate",60);
                graph.AddEdge("crab","shellfish",70);
                graph.AddEdge("oyster","invertebrate",80);
                graph.AddEdge("poodle","dog",90);
                graph.AddEdge("dalmatian","dog",100);
                graph.AddEdge("dachshund","dog",110);

                cout << "Weight of 'vertebrate to animal' is "
<< graph.WeightIs("vertebrate","animal") << endl;
                cout << "Weight of 'poodle to dog' is "
<< graph.WeightIs("poodle","dog") << endl;
```

2. Exercise 23

문 제

- 이 장에서 명시한 GraphType클래스에 DeleteEdge라는 연산을 추가하려고 한다. 이 연산은 주어진 간선(Edge)을 제거한다.
- A. 이 함수의 선언 부분을 작성하고, 적절한 주석을 넣으시오.
- B. 인접 행렬과 (a)의 선언을 사용하여 함수를 구현하여라.

- **Graph 클래스에 멤버함수로 DeleteEdge()를 추가하고 구현하시오.**

edge 삭제

- **작성의 예)**

```
//인접 행렬을 사용하여 표현하기 때문에,
// Edge를 추가하는 코드와 비슷한 방식. edge의 값(weight)을 초기값으로 reset합니다.

template<class VertexType>
void GraphType<VertexType>::DeleteEdge(VertexType fromVertex, VertexType toVertex)
{
        int row;
        int col;

        row = IndexIs(vertices, fromVertex);
        col = IndexIs(vertices, toVertex);
        ____________________; // edge에 초기값 대입
}
```

테스트 드라이버 (예제)

- DeleteEdge함수를 통하여 간선을 제거하고 GetToVertices함수를 이용하여 간선이 제거 된것을 확인함
  - GetToVertices 함수는 파라메터로 받는 버택스와 간선으로 연결된 모든 버택스를 큐에 담아오는 함수이다.

```
#include "GraphType.h"

#include <iostream>
using namespace std;

int main()
{
        GraphType<char*> graph;

        graph.AddVertex("dog");
        graph.AddVertex("cat");
        graph.AddVertex("animal");
        graph.AddVertex("vertebrate");
        graph.AddVertex("oyster");
        graph.AddVertex("shellfish");
        graph.AddVertex("invertebrate");
        graph.AddVertex("crab");
        graph.AddVertex("poodle");
        graph.AddVertex("monkey");
        graph.AddVertex("banana");
        graph.AddVertex("dalmatian");
        graph.AddVertex("dachshund");

        graph.AddEdge("vertebrate","animal",10);
        graph.AddEdge("invertebrate","animal",20);
        graph.AddEdge("dog","vertebrate",30);
        graph.AddEdge("cat","vertebrate",40);
        graph.AddEdge("monkey","vertebrate",50);
        graph.AddEdge("shellfish","invertebrate",60);
        graph.AddEdge("crab","shellfish",70);
        graph.AddEdge("oyster","invertebrate",80);
        graph.AddEdge("poodle","dog",90);
        graph.AddEdge("dalmatian","dog",100);
        graph.AddEdge("dachshund","dog",110);

        cout << "Weight of 'vertebrate to animal' is "
<< graph.WeightIs("vertebrate","animal") << endl;
        cout << "Weight of 'poodle to dog' is "
<< graph.WeightIs("poodle","dog") << endl;
```

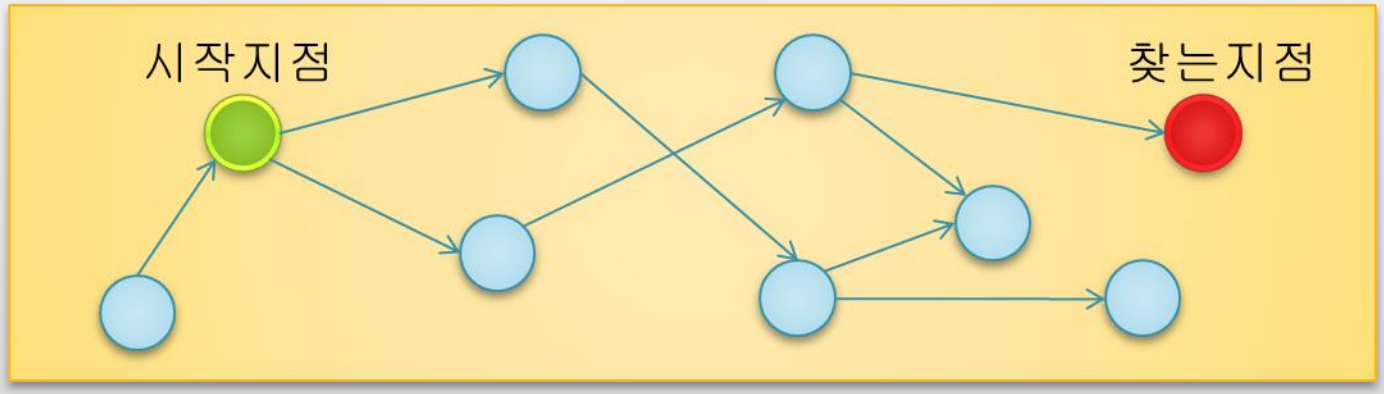
```
//23번 테스트. 삭제되었을 시 결과는 0 출력
graph.DeleteEdge("poodle","dog");
cout << "Weight of poodle to dog is "
<< graph.WeightIs("poodle","dog") << endl;

//GetToVertices()함수 테스트
cout << endl << "GetToVertices(dog,queue)" << endl;
QueType<char*> queue;
graph.GetToVertices("dog",queue);
while(!queue.IsEmpty())
{
        char* item;
        queue.Dequeue(item);
        cout << item << endl;
}
```

3. Exercise 25

■ 문 제

- ❖ 재귀적 용법(recursion)을 사용하면 스택을 사용하지 않고도 DepthFirstSearch 연산을 구현할 수 있다.
- ❖ A. base case 와 general case를 설명하여라.
- ❖ B. 깊이 우선 탐색의 재귀적 버전 알고리즘을 작성하여라.
  - 샘플 코드중 DFSearch.cpp에 비재귀 버전으로 구현된 내용을 참고



3-help slides (1/2)

구현의 예)

```
// 멤버 함수로 구현된 예
// GetToVertices() 함수에 구현된 내용을 바탕으로 해당 Vertices의 edge를 찾고, edge마다 함수
//를 재귀 호출하여 값을 찾으면 true, 찾지 못하면 false

template<class VertexType>
bool GraphType<VertexType>::DepthFirstSearch(VertexType startVertex, VertexType endVertex)
{
    QueType<VertexType> vertexQ;

    if(____________________) // base case
    {
        cout << endVertex;
        return true;
    }

    ____________________________; // startVertex의 인접노드들을 vertexQ에 집어 넣음
    while (!vertexQ.IsEmpty())
    {
        vertexQ.Dequeue(vertex);
        if (vertex != startVertex){
            if (____________________) { // vertrex를 시작노드로 하여 DepthFirstSearch를 재귀 호출
                cout << " <- " << vertex;
                return true;
            }
        } else
            continue;
    }
    return false;
}
```

3-help slides(2/2)

테스트 드라이버 (예제)

```
#include "GraphType.h"

#include <iostream>
using namespace std;

int main()
{
		GraphType<char*> graph;

		graph.AddVertex("dog");
		graph.AddVertex("cat");
		graph.AddVertex("animal");
		graph.AddVertex("vertebrate");
		graph.AddVertex("oyster");
		graph.AddVertex("shellfish");
		graph.AddVertex("invertebrate");
		graph.AddVertex("crab");
		graph.AddVertex("poodle");
		graph.AddVertex("monkey");
		graph.AddVertex("banana");
		graph.AddVertex("dalmatian");
		graph.AddVertex("dachshund");

		graph.AddEdge("vertebrate","animal",10);
		graph.AddEdge("invertebrate","animal",20);
		graph.AddEdge("dog","vertebrate",30);
		graph.AddEdge("cat","vertebrate",40);
		graph.AddEdge("monkey","vertebrate",50);
		graph.AddEdge("shellfish","invertebrate",60);
		graph.AddEdge("crab","shellfish",70);
		graph.AddEdge("oyster","invertebrate",80);
		graph.AddEdge("poodle","dog",90);
		graph.AddEdge("dalmatian","dog",100);
		graph.AddEdge("dachshund","dog",110);

		cout << "Weight of 'vertebrate to animal' is "
<< graph.WeightIs("vertebrate","animal")  << endl;
		cout << "Weight of 'poodle to dog' is "
<< graph.WeightIs("poodle","dog") << endl;
```

```
//25번에 구현한 함수 테스트
cout << endl << endl;
graph.DepthFirstSearch("dalmatian","animal");
cout << endl;

//엣지 증가, 함수 재 테스트
graph.AddVertex("apple");
graph.DeleteEdge("dog","vertebrate");
graph.AddEdge("dog","oyster",200);
graph.AddEdge("dog","banana",200);
graph.AddEdge("dog","apple",200);
cout << endl << endl;
graph.DepthFirstSearch("dalmatian","animal");
cout << endl;

return 0;
}
```