



순천향대학교
SOON CHUN HYANG
UNIVERSITY

자료구조 실습 HW 7

자료구조2 실습

담당교수	홍민 교수님
학과	컴퓨터소프트웨어공학과
학번	20204005
이름	김필중
제출일	2021년 10월 26일

| 목 차 |

1. 역 이름 인접행렬

- 1.1 문제 분석
- 1.2 소스 코드
- 1.3 소스 코드 분석
- 1.4 실행 화면
- 1.5 느낀점

2. 역 이름 인접 리스트

- 2.1 문제 분석
- 2.2 소스 코드
- 2.3 소스 코드 분석
- 2.4 실행 화면
- 2.5 느낀점

1.1 문제 분석

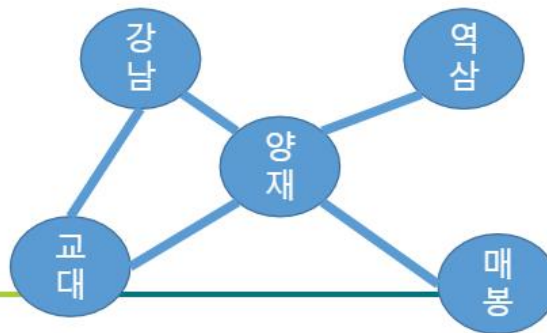
인접 행렬

- 372페이지의 프로그램 10.1을 이용하여 파일 data.txt에서 정점과 에지 정보를 입력받아 인접 행렬을 생성하여 그래프를 생성하는 프로그램을 작성하시오.

- data.txt 파일의 데이터 형식은 에지 정보(정점 정점)로 아래와 같이 구성되어 있음

```
강남 양재
양재 강남
양재 역삼
역삼 양재
강남 교대
교대 양재
```

- 아래의 그래프에 대해서 data.txt 파일들을 직접 생성하여 입력받으시오



SCH 순천향대학교

이번 1번 문제는 인접 행렬을 이용해 그래프를 생성하는 과제이다. 책에 있는 예제는 숫자를 직접 행과 열에 대입하는 형식이지만 과제는 역 이름을 통해 대입을 해야 한다. 그래서 따로 배열을 만들어 인덱스 번호를 체크해 대입하는 방식으로 접근할 예정이다. 또한 이름이 겹치면 안 되기 때문에 이름이 겹치는지 확인을 해야 한다.

우선 역에 대한 구조체를 선언해서 구조체 배열을 선언해 이름을 받은 후 중복을 확인하는 작업이 필요하다. 중복일 경우는 다음 문자열로 넘어가고 아닐 경우는 구조체 배열에 역 이름을 추가하고 역의 총 개수를 증가시켜주는 형식으로 가야 한다.

다음으로는 인접 행렬 구조체를 선언한다. 구조체 안에는 정점의 개수와 그래프의 인접 행렬을 2차원 배열로 선언한다

다음으로는 그래프 초기화 함수를 만든다. 그래프가 들어올 경우 정점의 개수를 0개로 선언하고 행과 열을 모두 0으로 초기화 하는 함수를 만들어 줘야 한다

다음으로는 정점 삽입 연산을 만든다. 현재 정점의 개수 + 1개가 행 혹은 열보다 클 경우 잘못된 것이므로 오류를 출력하고 아닐 경우는 정점의 개수를 1개 늘려 준다.

다음으로는 간선 삽입 연산 함수가 필요하다. 행과 열 그리고 그래프를 받아서 받은 그래프의 행 인덱스와 열 인덱스 위치를 1로 바꿔준다.

인접 행렬 출력 함수는 정점의 개수만큼 반복하면서 행을 고정시키고 열을 1씩 정점의 크기만큼 반복하면서 한 줄로 출력하고 다 출력했으면 행을 하나 증가시키는 이중 반복문을 통해 출력한다.

메인 함수에서는 우선 역 이름을 받아온 후 겹치는 지 확인하고 안 겹칠 경우는 배열에 역 이름 저장을 하면서 역의 개수를 증가시킨다. 이 과정을 반복 한 후 역의 개수 만큼 정점 삽입 연산이 진행되고 2개의 역을 읽어와 역의 인덱스 번호를 가지고 간선 삽입 함수를 진행한다.

1.2 소스 코드

```
1  //-----
2  // 제작 기간 : 2021년 10월 13일 ~ 10월 20일
3  // 제작자 : 20204005 김필중
4  // 프로그램명: 역 이름 인접행렬
5  //-----
6
7  // 필요한 헤더파일 선언
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11
12 // 오류 방지 구문 선언
13 #pragma warning(disable : 4996)
14
15 // MAX_VERTICES 정의
16 #define MAX_VERTICES 50
17
18 // 역 구조체 선언
19 typedef struct Station
20 {
21     char name[10]; // 역 이름 변수
22 } Station;
23
24 // 인접 행렬 구조체
25 typedef struct GraphType
26 {
27     int n; // 정점의 개수
28     int adj_mat[MAX_VERTICES][MAX_VERTICES]; // 그래프 인접 행렬
29 } GraphType;
30
31 // 그래프 초기화
32 void init(GraphType* g)
33 {
34     int r, c; // 변수 r, c 선언
35     g->n = 0; // g의 개수를 0개로 바꿈
36     for (r = 0; r < MAX_VERTICES; r++) // 행을 전역변수의 크기만큼 반복한다
37         for (c = 0; c < MAX_VERTICES; c++) // 열을 전역변수의 크기만큼 반복한다
38             g->adj_mat[r][c] = 0; // r행의 c열을 0으로 초기화한다.
39 }
40
41 // 정점 삽입 연산
42 void insert_vertex(GraphType* g, int v)
43 {
44     if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점의 개수 + 1개가 전역 변수 MAX_VERTICES 보다 크면 오류 발생 후 종료
45     {
46         fprintf(stderr, "그래프: 정점의 개수 초과");
47         return;
48     }
49     g->n++; // 정점의 개수를 1개 증가시킨다
50 }
51
```

```

51 // 간선 삽입 연산
52 void insert_edge(GraphType* g, int start, int end)
53 {
54     if (start >= g->n || end >= g->n) // 만약 행의 값, 혹은 열의 값이 정점의 개수보다 크거나 같을 경우 오류 발생 후 종료
55     {
56         fprintf(stderr, "그래프: 정점 번호 오류");
57         return;
58     }
59     g->adj_mat[start][end] = 1; // 받은 행과 열에 1 삽입
60 }
61
62 // 인접 행렬 출력 함수
63 void print_adj_mat(GraphType* g, Station* s)
64 {
65     for (int i = 0; i < g->n; i++) // 정점의 개수만큼 반복한다
66     {
67         printf("%2s ", s[i].name); // 배열에 저장된 역 이름을 출력한다
68     }
69     printf("\n");
70     for (int i = 0; i < g->n; i++) // 정점의 개수만큼 반복한다
71     {
72         printf("%s ", s[i].name); // 배열에 저장된 i번째 역 이름을 출력한다
73         for (int j = 0; j < g->n; j++) // j를 선언하고 정점의 개수만큼 반복한다
74         {
75             printf("%2d ", g->adj_mat[i][j]); // 그래프에서 i행 j열의 값을 출력한다
76         }
77         printf("\n");
78     }
79 }
80
81 // 메인 함수
82 int main(void)
83 {
84     GraphType *g; // 그래프 포인터 g를 선언
85     FILE *fp; // 파일포인터 fp를 선언
86
87     Station s[10]; // 구조체 배열 s를 선언
88     char temp1[10]; // 임시 저장 문자열 temp1을 선언
89     char temp2[10]; // 임시 저장 문자열 temp1을 선언
90
91     int tmp = 0; // 역의 총 개수 확인 정수형 변수
92     int check = 0; // 겹치는지 확인하는 변수
93     int i, j; // 반복문에 사용할 변수 선언
94
95     fp = fopen("data.txt", "r"); // 파일을 오픈한다.
96
97     // 만약 파일 오픈에 실패할 경우 오류를 출력하고 종료한다
98     if (fp == NULL)
99     {
100         printf("파일 오픈 실패\n");
101         return 0;
102     }
103
104     // 파일 끝까지 반복하면서 구조체 배열에 역 이름 집어넣기
105     while (!feof(fp))
106     {
107         fscanf(fp, "%s ", temp1); // 문자열을 파일에서 읽어 온다
108         for (i = 0; i < 10; i++) // 10번 만큼 반복한다
109         {
110             if (strcmp(s[i].name, temp1) == 0) // 만약 i번째 문자열 배열과 받아온 문자열이 같을 경우
111             {
112                 check = 1; // 겹친다는 뜻이므로 1로 변경
113                 break; // 반복문 종료
114             }
115         }
116         if (check == 0) // 겹치지 않을 경우는
117         {
118             strcpy(s[tmp].name, temp1); // tmp 번째 배열에 문자열 저장
119             tmp++; // tmp 증가
120         }
121         check = 0; // 0으로 변경
122     }
123
124     rewind(fp); // 파일 포인터를 처음으로 돌린다
125
126     g = (GraphType *)malloc(sizeof(GraphType)); // g를 동적할당을 한다
127     init(g); // g를 초기화 시켜준다
128     for (int i = 0; i < tmp; i++) // 총 역의 개수만큼 반복한다
129     {
130         insert_vertex(g, i); // 정점 삽입 연산을 진행한다
131     }
132

```

```

133 // 파일 끝까지 반복하면서 구조체 배열을 읽어 간선 삽입하기
134 while (!feof(fp))
135 {
136     fscanf(fp, "%s %s", temp1, temp2); // 텍스트 파일에서 2개의 문자열 읽어온다
137     // 행의 인덱스 번호를 알기 위한 반복문
138     for (i = 0; i < tmp; i++) // tmp 만큼 반복한다
139     {
140         if (strcmp(s[i].name, temp1) == 0) // 만약 배열의 i번째 이름과 temp1의 문자열이 같으면
141             break; // 종료한다
142     }
143     // 열의 인덱스 번호를 알기 위한 반복문
144     for (j = 0; j < tmp; j++) // tmp 만큼 반복한다
145     {
146         if (strcmp(s[j].name, temp2) == 0) // 만약 배열의 j번째 이름과 temp2의 문자열이 같으면
147             break; // 종료한다
148     }
149
150     insert_edge(g, i, j); // i번째 행과 j번째 열을 간선 삽입 연산을 진행한다
151 }
152
153
154 print_adj_mat(g, s); // 인접행렬을 출력 함수를 실행한다
155
156 free(g); // 동적할당을 해제한다
157 fclose(fp); // 파일을 닫는다
158 return 0; // 종료한다
159 }

```

1.3 소스 코드 분석

```
// 필요한 헤더파일 선언
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

1. 필요한 헤더파일을 선언한다.

```
// 오류 방지 구문 선언
#pragma warning(disable : 4996)

// 전역 변수 MAX_VERTICES 선언
#define MAX_VERTICES 50
```

2. 오류 방지 구문을 선언한다

3. MAX_VERTICES를 50으로 정의한다.

```
// 역 구조체 선언
typedef struct Station
{
    char name[10]; // 역 이름 변수
}Station;
```

4. 역 구조체를 선언해 역 이름 변수를 선언한다

```
// 인접 행렬 구조체
typedef struct GraphType
{
    int n; // 정점의 개수
    int adj_mat[MAX_VERTICES][MAX_VERTICES]; // 그래프 인접 행렬
} GraphType;
```

5. 인접 행렬 구조체를 선언한다

6. 정점의 개수와 그래프 인접 행렬을 2차원 배열로 선언한다


```

// 그래프 초기화
void init(GraphType* g)
{
    int r, c; // 변수 r, c 선언
    g->n = 0; // g의 개수를 0개로 바꿈
    for (r = 0; r < MAX_VERTICES; r++) // 행을 전역변수의 크기만큼 반복한다
        for (c = 0; c < MAX_VERTICES; c++) // 열을 전역변수의 크기만큼 반복한다
            g->adj_mat[r][c] = 0; // r행의 c열을 0으로 초기화한다.
}

```

7. 그래프 초기화 함수를 만든다

8. 변수 r, c를 선언한 후 그래프 g의 정점의 개수를 0개로 초기화 한다

9. 이중 for문을 통해 g의 인접 행렬 2차원 배열을 0으로 모두 초기화 한다

```

// 정점 삽입 연산
void insert_vertex(GraphType* g, int v)
{
    if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점의 개수 + 1개가 전역 변수 MAX_VERTICES 보다 크면 오류 발생 후 종료
    {
        fprintf(stderr, "그래프: 정점의 개수 초과");
        return;
    }
    g->n++; // 정점의 개수를 1개 증가시킨다
}

```

10. 만약 현재 정점의 개수 + 1, 즉 1개 증가시킨 개수가 max_vertices보다 크다면 오류 발생 후 종료

11. 아닐 경우는 정점의 개수를 1개 증가시킨다

```

// 간선 삽입 연산
void insert_edge(GraphType* g, int start, int end)
{
    if (start >= g->n || end >= g->n) // 만약 행의 값, 혹은 열의 값이 정점의 개수보다 크거나 같을 경우 오류 발생 후 종료
    {
        fprintf(stderr, "그래프: 정점 번호 오류");
        return;
    }
    g->adj_mat[start][end] = 1; // 받은 행과 열에 1 삽입
}

```

12. 행의 값과 열의 값이 개수보다 크거나 같을 경우는 오류 발생후 종료한다

13. 아닐 경우는 받은 행과 열에 1을 삽입한다

```

// 인접 행렬 출력 함수
void print_adj_mat(GraphType* g, Station* s)
{
    for (int i = 0; i < g->n; i++) // 정점의 개수만큼 반복한다
    {
        printf("%%2s ", s[i].name); // 배열에 저장된 역 이름을 출력한다
    }
    printf("\n");
    for (int i = 0; i < g->n; i++) // 정점의 개수만큼 반복한다
    {
        printf("%s ", s[i].name); // 배열에 저장된 i번째 역 이름을 출력한다
        for (int j = 0; j < g->n; j++) // j를 선언하고 정점의 개수만큼 반복한다
        {
            printf("%%2d ", g->adj_mat[i][j]); // 그래프에서 i행 j열의 값을 출력한다
        }
        printf("\n");
    }
}

```

14. 인접 행렬 출력 함수는 우선 정점의 개수만큼 반복하면서 배열에 저장된 역 이름을 출력한다

15. 정점의 개수만큼 반복하면서 배열에 저장된 i번째 역 이름을 출력한다.

16. j를 가진 반복문을 통해 i행과 j열의 배열의 값을 출력한다

```

GraphType *g; // 그래프 포인터 g를 선언
FILE *fp; // 파일포인터 fp를 선언

Station s[10]; // 구조체 배열 s를 선언
char temp1[10]; // 임시 저장 문자열 temp1을 선언
char temp2[10]; // 임시 저장 문자열 temp1을 선언

int tmp = 0; // 역의 총 개수 확인 정수형 변수
int check = 0; // 겹치는지 확인하는 변수
int i, j; // 반복문에 사용할 변수 선언

```

17. 필요한 변수들을 선언한다

18. 그래프 포인터 g, 파일 포인터 fp를 선언한다

19. 구조체 배열 s를 선언하고 임시 저장 문자열 temp를 선언한다.

20. 역의 총 개수 확인하는 변수, 겹치는지 확인 변수, 반복문 변수를 선언한다

```

fp = fopen("data.txt", "r"); // 파일을 오픈한다.

// 만약 파일 오픈에 실패할 경우 오류를 출력하고 종료한다
if (fp == NULL)
{
    printf("파일 오픈 실패\n");
    return 0;
}

```

21. 파일 오픈을 한 후 파일 오픈 실패 시 오류 메시지 출력하고 종료한다

```

// 파일 끝까지 반복하면서 구조체 배열에 역 이름 집어넣기
while (!feof(fp))
{
    fscanf(fp, "%s ", tmp1); // 문자열을 파일에서 읽어 온다
    for (i = 0; i < 10; i++) // 10번 만큼 반복한다
    {
        if (strcmp(s[i].name, tmp1) == 0) // 만약 i번째 문자열 배열과 받아온 문자열이 같을 경우
        {
            check = 1; // 겹친다는 뜻이므로 1로 변경
            break; // 반복문 종료
        }
    }
    if (check == 0) // 겹치지 않을 경우는
    {
        strcpy(s[tmp].name, tmp1); // tmp 번째 배열에 문자열 저장
        tmp++; // tmp 증가
    }
    check = 0; // 0으로 변경
}

```

22. 파일 끝까지 반복하면서 문자열 1개를 읽어온다

23. 10번만큼 반복하면서 만약 i번째 문자열 배열의 값과 받아온 문자열이 같을 경우는 겹치므로 check를 1로 바꾸고 반복문 빠져나온다

24. 만약 check 가 0 즉 안겹칠경우는 s배열에 tmp 인덱스에 역 이름을 저장하고 tmp 를 증가시킨다

25. 다시 check = 0을 한다

```

rewind(fp); // 파일 포인터를 처음으로 돌린다

g = (GraphType *)malloc(sizeof(GraphType)); // g를 동적할당을 한다
init(g); // g를 초기화 시켜준다
for (int i = 0; i < tmp; i++) // 총 역의 개수만큼 반복한다
    insert_vertex(g, i); // 정점 삽입 연산을 진행한다

```

26. 파일 포인터를 처음으로 돌린 후 g를 동적할당을 한다

27. g를 초기화 시키고 총 역의 개수만큼 반복을 진행한다

28. 정점 삽입 연산을 진행한다

```

// 파일 끝까지 반복하면서 구조체 배열을 읽어 간선 삽입하기
while (!feof(fp))
{
    fscanf(fp, "%s %s", temp1, temp2); // 텍스트 파일에서 2개의 문자열 읽어온다
    // 행의 인덱스 번호를 알기 위한 반복문
    for (i = 0; i < tmp; i++) // tmp 만큼 반복한다
    {
        if (strcmp(s[i].name, temp1) == 0) // 만약 배열의 i번째 이름과 temp1의 문자열이 같으면
            break; // 종료한다
    }
    // 열의 인덱스 번호를 알기 위한 반복문
    for (j = 0; j < tmp; j++) // tmp 만큼 반복한다
    {
        if (strcmp(s[j].name, temp2) == 0) // 만약 배열의 j번째 이름과 temp2의 문자열이 같으면
            break; // 종료한다
    }

    insert_edge(g, i, j); // i번째 행과 j번째 열을 간선 삽입 연산을 진행한다
}

```

29. 파일 끝까지 반복하면서 이번에는 2개의 문자열을 읽어온다

30. 첫번째 문자열과 i번째 문자열 배열의 값이 같으면 반복문을 종료한다

31. 두번째 문자열과 j번째 문자열 배열의 값이 같으면 반복문을 종료한다

32. i 행과 j 열을 간선 삽입 연산에 넣어준다

```

print_adj_mat(g, s); // 인접행렬을 출력 함수를 실행한다

free(g); // 동적할당을 해제한다
fclose(fp); // 파일을 닫는다
return 0; // 종료한다

```

33. 인접 행렬을 출력 함수를 실행하고 동적할당 해제하고 파일을 닫은 후 종료한다

1.4 실행 결과

Microsoft Visual Studio 디버그 콘솔

	강남	양재	역삼	교대	매봉
강남	0	1	0	1	0
양재	1	0	1	1	1
역삼	0	1	0	0	0
교대	1	1	0	0	0
매봉	0	1	0	0	0

C:\Users\korca\Desktop\20204005_김필중_자료구조실습 7주차 과제\Debug (프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

data - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

강남 양재
양재 강남
양재 역삼
역삼 양재
강남 교대
교대 강남
교대 양재
양재 교대
매봉 양재
양재 매봉

1.5 느낀점

우선 이번 과제를 하면서 그래프는 처음 만들어 본 것이다. 그래프를 표현하는 것 중 인접행렬을 이용했다. 인접행렬을 이용하면 두 정점을 연결하는 간선의 존재 여부를 즉시 알 수 있는 장점이 있다. 이 문제에서는 역끼리 연결 되어있는 선을 바로 알 수 있다는 장점이 있다. 역이 많아지면 많아 질수록 1이 보이는 수가 많아질 수 있으므로 파악이 빨리 가능하다는 장점이 있는 문제이다. 숫자가 아닌 단어나 문자열을 이용할 경우 배열을 만들어 이용하면 더 보기가 쉬워지고 꼭 숫자로 치환할 필요가 없다는 것을 이번 과제를 하면서 느꼈다.

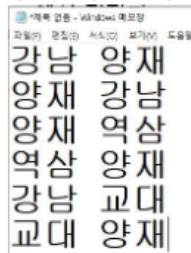
2.1 문제 분석



■ 인접 리스트

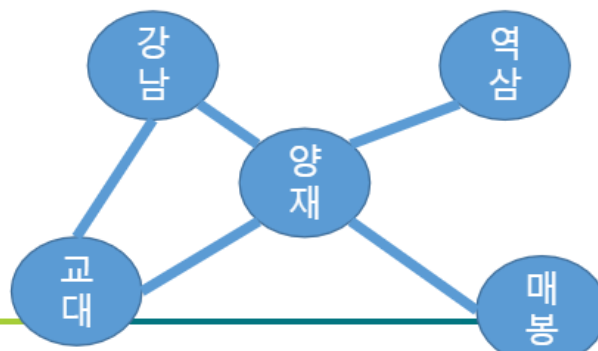
- 375페이지의 프로그램 10.2을 이용하여 파일 data.txt에서 정점과 에지 정보를 입력받아 인접 행렬을 생성하여 그래프를 생성하는 프로그램을 작성하시오.

- data.txt 파일의 데이터 형식은 에지 정보(정점 정점)로 아래와 같이 구성되어 있음



```
강남 양재
양재 강남
양재 역삼
역삼 양재
강남 교대
교대 양재
```

- 아래의 그래프에 대해서 data.txt 파일들을 직접 생성하여 입력받으시오



이번 2번 문제는 인접 리스트를 이용해 그래프를 생성하는 과제이다. 1번과 마찬가지로 구조체 배열을 통해 역의 이름을 대입하고 판단하는 방식으로 진행할 것이다.

우선 역에 대한 구조체를 선언해서 구조체 배열을 선언해 이름을 받은 후 중복을 확인하는 작업이 필요하다. 중복일 경우는 다음 문자열로 넘어가고 아닐 경우는 구조체 배열에 역 이름을 추가하고 역의 총 개수를 증가시켜주는 형식으로 가야 한다.

다음으로는 인접 리스트 노드 구조체를 선언한다. 여기에는 정점의 값과 링크를 만들어 준다

다음으로는 인접 리스트 구조체를 선언한다. 여기에는 정점의 개수와 1차원 배열의 배열을 생성한다

다음으로는 그래프 초기화 함수를 만든다. 그래프가 들어올 경우 정점의 개수를 0개로 선언하고 선언한 값만큼 리스트를 null로 초기화 한다

다음으로는 정점 삽입 연산을 만든다. 현재 정점의 개수 + 1개가 행 혹은 열보다 클 경우 잘못된 것이므로 오류를 출력하고 아닐 경우는 정점의 개수를 1개 늘려 준다.

다음으로는 간선 삽입 연산 함수가 필요하다. 노드 포인터를 만든 후 동적할당을 하고 노드 정점의 값에 받아온 값을 넣어주고 연결하는 함수이다

인접 리스트 출력 함수는 정점의 개수만큼 반복하면서 p포인터를 선언하고 i번째 리스트를 가리키게 한다. 그 후 역 이름을 출력하고 p가 널이 아닐때까지 반복하면서 $p \rightarrow \text{vertex}$ 인덱스의 값의 이름을 출력하는 방식으로 만든다

메인 함수에서는 우선 역 이름을 받아온 후 겹치는 지 확인하고 안 겹칠 경우는 배열에 역 이름 저장을 하면서 역의 개수를 증가시킨다. 이 과정을 반복 한 후 역의 개수 만큼 정점 삽입 연산이 진행되고 2개의 역을 읽어와 역의 인덱스 번호를 가지고 간선 삽입 함수를 진행한다.

2.2 소스 코드

```
1  //-----
2  // 제작 기간 : 2021년 10월 13일 ~ 10월 20일
3  // 제작자 : 20204005 김필중
4  // 프로그래밍: 역 이름 인접리스트
5  //-----
6
7  // 필요한 헤더파일 선언
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11
12 // 오류 방지 구문 선언
13 #pragma warning(disable : 4996)
14
15 // MAX_VERTICES 정의
16 #define MAX_VERTICES 50
17
18 // 역 구조체 선언
19 typedef struct Station
20 {
21     char name[10]; // 역 이름 변수
22 }Station;
23
24 // 인접 리스트 구조체
25 typedef struct GraphNode
26 {
27     int vertex; // 정점
28     struct GraphNode* link; // 리스트의 링크
29 } GraphNode;
30
31 typedef struct GraphType
32 {
33     int n; // 정점의 개수
34     GraphNode* adj_list[MAX_VERTICES]; // MAX_VERTICES 만큼의 배열 생성
35 } GraphType;
36
37 // 그래프 초기화
38 void init(GraphType* g)
39 {
40     int v; // v 선언
41     g->n = 0; // g의 개수를 0개로 바꿈
42     for (v = 0; v < MAX_VERTICES; v++) // MAX_VERTICES 만큼 반복한다
43         g->adj_list[v] = NULL; // v번째 배열을 null로 초기화 한다.
44 }
45
46 // 정점 삽입 연산
47 void insert_vertex(GraphType* g, int v)
48 {
49     if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점의 개수 + 1개가 전역 변수 MAX_VERTICES 보다 크면 오류 발생 후 종료
50     {
51         fprintf(stderr, "그래프: 정점의 개수 초과");
52         return;
53     }
54     g->n++; // 정점의 개수를 1개 증가시킨다
55 }
```



```

57 // 간선 삽입 연산, v를 u의 인접 리스트에 삽입한다.
58 void insert_edge(GraphType* g, int u, int v)
59 {
60     GraphNode* node; // GraphNode 노드 포인터를 만든다
61     if (u >= g->n || v >= g->n) // 만약 u 값, 혹은 v 값이 정점의 개수보다 크거나 같을 경우 오류 발생 후 종료
62     {
63         fprintf(stderr, "그래프: 정점 번호 오류");
64         return;
65     }
66     node = (GraphNode*)malloc(sizeof(GraphNode)); // node를 동적할당 해준다
67     node->vertex = v; // 정점에 v값을 삽입한다
68     node->link = g->adj_list[u]; // 링크를 g 그래프의 u번째에 연결한다
69     g->adj_list[u] = node; // g그래프의 u번째를 노드로 변경한다
70 }
71
72 // 인접 리스트 출력 함수
73 void print_adj_list(GraphType* g, Station* s)
74 {
75     for (int i = 0; i < g->n; i++) // 정점의 개수까지 반복한다
76     {
77         GraphNode* p = g->adj_list[i]; // p 포인터는 i번째 리스트를 가리킨다
78         printf("%s의 인접 리스트 ", s[i].name); // i번째 역 이름을 출력한다
79         while (p != NULL) // p가 null 이 아닐때까지 반복한다
80         {
81             printf("-> %s ", s[p->vertex].name); // 구조체 배열 s의 p->vertex 인덱스 값의 이름을 출력한다
82             p = p->link; // p를 한칸 옮겨준다
83         }
84         printf("\n");
85     }
86 }
87

```

```

88 // 메인 함수
89 int main(void)
90 {
91     GraphType *g; // 그래프 포인터 g를 선언
92     FILE *fp; // 파일포인터 fp를 선언
93
94     Station s[10]; // 구조체 배열 s를 선언
95     char temp1[10]; // 임시 저장 문자열 temp1을 선언
96     char temp2[10]; // 임시 저장 문자열 temp1을 선언
97
98     int tmp = 0; // 역의 총 개수 확인 정수형 변수
99     int check = 0; // 겹치는지 확인하는 변수
100    int i, j; // 반복문에 사용할 변수 선언
101
102    fp = fopen("data.txt", "r"); // 파일을 오픈한다.
103
104    // 만약 파일 오픈에 실패할 경우 오류를 출력하고 종료한다
105    if (fp == NULL)
106    {
107        printf("파일 오픈 실패\n");
108        return 0;
109    }
110
111    // 파일 끝까지 반복하면서 구조체 배열에 역 이름 집어넣기
112    while (!feof(fp))
113    {
114        fscanf(fp, "%s ", temp1); // 문자열을 파일에서 읽어 온다
115        for (i = 0; i < 10; i++) // 10번 만큼 반복한다
116        {
117            if (strcmp(s[i].name, temp1) == 0) // 만약 i번째 문자열 배열과 받아온 문자열이 같을 경우
118            {
119                check = 1; // 겹친다는 뜻이므로 1로 변경
120                break; // 반복문 종료
121            }
122        }
123        if (check == 0) // 겹치지 않을 경우는
124        {
125            strcpy(s[tmp].name, temp1); // tmp 번째 배열에 문자열 저장
126            tmp++; // tmp 증가
127        }
128        check = 0; // 0으로 변경
129    }
130
131    rewind(fp); // 파일 포인터를 처음으로 돌린다
132
133    g = (GraphType *)malloc(sizeof(GraphType)); // g를 동적할당을 한다
134    init(g); // g를 초기화 시켜준다
135    for (int i = 0; i < tmp; i++) // 총 역의 개수만큼 반복한다
136        insert_vertex(g, i); // 정점 삽입 연산을 진행한다
137

```

```

133 g = (GraphType *)malloc(sizeof(GraphType)); // g를 동적할당을 한다
134 init(g); // g를 초기화 시켜준다
135 for (int i = 0; i < tmp; i++) // 총 역의 개수만큼 반복한다
136     insert_vertex(g, i); // 정점 삽입 연산을 진행한다
137
138 // 파일 끝까지 반복하면서 구조체 배열을 읽어 간선 삽입하기
139 while (!feof(fp))
140 {
141     fscanf(fp, "%s %s", temp1, temp2); // 텍스트 파일에서 2개의 문자열 읽어온다
142     // 행의 인덱스 번호를 알기 위한 반복문
143     for (i = 0; i < tmp; i++) // tmp 만큼 반복한다
144     {
145         if (strcmp(s[i].name, temp1) == 0) // 만약 배열의 i번째 이름과 temp1의 문자열이 같으면
146             break; // 종료한다
147     }
148     // 열의 인덱스 번호를 알기 위한 반복문
149     for (j = 0; j < tmp; j++) // tmp 만큼 반복한다
150     {
151         if (strcmp(s[j].name, temp2) == 0) // 만약 배열의 j번째 이름과 temp2의 문자열이 같으면
152             break; // 종료한다
153     }
154
155     insert_edge(g, i, j); // i번째 행과 j번째 열을 간선 삽입 연산을 진행한다
156 }
157
158 print_adj_list(g, s); // 인접 리스트를 출력 함수를 실행한다
159
160 free(g); // 동적할당을 해제한다
161 fclose(fp); // 파일을 닫는다
162 return 0; // 종료한다
163 }

```

2.3 소스 코드 분석

```
// 필요한 헤더파일 선언
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 오류 방지 구문 선언
#pragma warning(disable : 4996)

// MAX_VERTICES 정의
#define MAX_VERTICES 50

// 역 구조체 선언
```

1. 필요한 헤더파일을 선언한다.
2. 오류 방지 구문을 선언한다
3. MAX_VERTICES를 50으로 정의한다.

```
// 역 구조체 선언
typedef struct Station
{
    char name[10]; // 역 이름 변수
}Station;
```

4. 역 구조체를 선언해 역 이름 변수를 선언한다

```
// 인접 리스트 구조체
typedef struct GraphNode
{
    int vertex; // 정점
    struct GraphNode* link; // 리스트의 링크
} GraphNode;
```

5. 인접 리스트 노드 구조체를 선언한다
6. 정점 변수와 리스트의 링크를 만들어 준다.

```

typedef struct GraphType
{
    int n; // 정점의 개수
    GraphNode* adj_list[MAX_VERTICES]; // MAX_VERTICES 만큼의 배열 생성
} GraphType;

```

7. 그래프 구조체를 만든다.

8. 정점의 개수 변수와 MAX_VERTICES 크기의 배열을 생성한다

```

// 그래프 초기화
void init(GraphType* g)
{
    int v; // v 선언
    g->n = 0; // g의 개수를 0개로 바꿈
    for (v = 0; v < MAX_VERTICES; v++) // MAX_VERTICES 만큼 반복한다
        g->adj_list[v] = NULL; // v번째 배열을 null로 초기화 한다.
}

```

9. v를 선언하고 g의 정점의 개수를 0개로 바꾼다.

10. MAX_VERTICES 만큼 반복하면서 리스트의 v 번째 배열을 null로 초기화한다

```

// 정점 삽입 연산
void insert_vertex(GraphType* g, int v)
{
    if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점의 개수 + 1개가 전역 변수 MAX_VERTICES 보다 크면 오류 발생 후 종료
    {
        fprintf(stderr, "그래프: 정점의 개수 초과");
        return;
    }
    g->n++; // 정점의 개수를 1개 증가시킨다
}

```

11. 만약 현재 정점의 개수 + 1, 즉 1개 증가시킨 개수가 MAX_VERTICES 보다 크다면 오류 발생 후 종료

12. 아닐 경우는 정점의 개수를 1개 증가시킨다

```

// 간선 삽입 연산, v를 u의 인접 리스트에 삽입한다.
void insert_edge(GraphType* g, int u, int v)
{
    GraphNode* node; // GraphNode 노드 포인터를 만든다
    if (u >= g->n || v >= g->n) // 만약 u 값, 혹은 v 값이 정점의 개수보다 크거나 같을 경우 오류 발생 후 종료
    {
        fprintf(stderr, "그래프: 정점 번호 오류");
        return;
    }
    node = (GraphNode*)malloc(sizeof(GraphNode)); // node를 동적할당 해준다
    node->vertex = v; // 정점에 v값을 삽입한다
    node->link = g->adj_list[u]; // 링크를 g 그래프의 u번째에 연결한다
    g->adj_list[u] = node; // g그래프의 u번째를 노드로 변경한다
}

```

13. graphnode 노드 포인터를 만든 후 u, v값 둘 중의 하나가 정점의 개수보다 크거나 같으면 오류 발생 후 종료한다

14. node를 동적할당 해준 후 정점에 v값을 삽입한다

15. 링크를 g 그래프의 u번째 배열에 연결한 후 g 그래프의 u 번째를 노드로 변경한다

```

// 인접 리스트 출력 함수
void print_adj_list(GraphType* g, Station* s)
{
    for (int i = 0; i < g->n; i++) // 정점의 개수까지 반복한다
    {
        GraphNode* p = g->adj_list[i]; // p 포인터는 i번째 리스트를 가리킨다
        printf("%s의 인접 리스트 ", s[i].name); // i번째 역 이름을 출력한다
        while (p != NULL) // p가 null 이 아닐때까지 반복한다
        {
            printf("-> %s ", s[p->vertex].name); // 구조체 배열 s의 p->vertex 인덱스 값의 이름을 출력한다
            p = p->link; // p를 한칸 옮겨준다
        }
        printf("\n");
    }
}

```

16. 정점의 개수까지 반복한다

17. p 포인터는 i번째 리스트를 가리키면서 구조체 배열의 i번째 역의 이름을 출력한다

18. p가 null이 아닐 때까지 반복하면서 구조체 배열 s의 p->vertex값을 인덱스로 사용하고 인덱스 값의 이름을 출력한다.

19. p를 한 칸 옮겨준다

```

// 메인 함수
int main(void)
{
    GraphType *g; // 그래프 포인터 g를 선언
    FILE *fp; // 파일포인터 fp를 선언

    Station s[10]; // 구조체 배열 s를 선언
    char temp1[10]; // 임시 저장 문자열 temp1을 선언
    char temp2[10]; // 임시 저장 문자열 temp1을 선언

    int tmp = 0; // 역의 총 개수 확인 정수형 변수
    int check = 0; // 겹치는지 확인하는 변수
    int i, j; // 반복문에 사용할 변수 선언
}

```

20. 필요한 변수들을 선언한다
21. 그래프 포인터 g, 파일 포인터 fp를 선언한다
22. 구조체 배열 s를 선언하고 임시 저장 문자열 temp를 선언한다.
23. 역의 총 개수 확인하는 변수, 겹치는지 확인 변수, 반복문 변수를 선언한다

```

fp = fopen("data.txt", "r"); // 파일을 오픈한다.

// 만약 파일 오픈에 실패할 경우 오류를 출력하고 종료한다
if (fp == NULL)
{
    printf("파일 오픈 실패\n");
    return 0;
}

```

24. 파일 오픈을 한 후 파일 오픈 실패 시 오류 메시지 출력하고 종료한다

```

// 파일 끝까지 반복하면서 구조체 배열에 역 이름을 집어넣기
while (!feof(fp))
{
    fscanf(fp, "%s ", temp1); // 문자열을 파일에서 읽어 온다
    for (i = 0; i < 10; i++) // 10번 만큼 반복한다
    {
        if (strcmp(s[i].name, temp1) == 0) // 만약 i번째 문자열 배열과 받아온 문자열이 같을 경우
        {
            check = 1; // 겹친다는 뜻이므로 1로 변경
            break; // 반복문 종료
        }
    }
    if (check == 0) // 겹치지 않을 경우는
    {
        strcpy(s[tmp].name, temp1); // tmp 번째 배열에 문자열 저장
        tmp++; // tmp 증가
    }
    check = 0; // 0으로 변경
}

```

25. 파일 끝까지 반복하면서 문자열 1개를 읽어온다

26. 10번만큼 반복하면서 만약 i번째 문자열 배열의 값과 받아온 문자열이 같을 경우는 겹치므로 check를 1로 바꾸고 반복문 빠져나온다

27. 만약 check 가 0 즉 안겹칠경우는 s배열에 tmp 인덱스에 역 이름을 저장하고 tmp를 증가시킨다

28. 다시 check = 0을 한다

```

rewind(fp); // 파일 포인터를 처음으로 돌린다

g = (GraphType *)malloc(sizeof(GraphType)); // g를 동적할당을 한다
init(g); // g를 초기화 시켜준다
for (int i = 0; i < tmp; i++) // 총 역의 개수만큼 반복한다
    insert_vertex(g, i); // 정점 삽입 연산을 진행한다

```

29. 파일 포인터를 처음으로 돌린 후 g를 동적할당을 한다

30. g를 초기화 시키고 총 역의 개수만큼 반복을 진행한다

31. 정점 삽입 연산을 진행한다


```

// 파일 끝까지 반복하면서 구조체 배열을 읽어 간선 삽입하기
while (!feof(fp))
{
    fscanf(fp, "%s %s", temp1, temp2); // 텍스트 파일에서 2개의 문자열 읽어온다
    // 행의 인덱스 번호를 알기 위한 반복문
    for (i = 0; i < tmp; i++) // tmp 만큼 반복한다
    {
        if (strcmp(s[i].name, temp1) == 0) // 만약 배열의 i번째 이름과 temp1의 문자열이 같으면
            break; // 종료한다
    }
    // 열의 인덱스 번호를 알기 위한 반복문
    for (j = 0; j < tmp; j++) // tmp 만큼 반복한다
    {
        if (strcmp(s[j].name, temp2) == 0) // 만약 배열의 j번째 이름과 temp2의 문자열이 같으면
            break; // 종료한다
    }

    insert_edge(g, i, j); // i번째 행과 j번째 열을 간선 삽입 연산을 진행한다
}

```

32. 파일 끝까지 반복하면서 이번에는 2개의 문자열을 읽어온다
33. 첫번째 문자열과 i번째 문자열 배열의 값이 같으면 반복문을 종료한다
34. 두번째 문자열과 j번째 문자열 배열의 값이 같으면 반복문을 종료한다
35. i 행과 j 열을 간선 삽입 연산에 넣어준다

```

print_adj_list(g, s); // 인접 리스트를 출력 함수를 실행한다

free(g); // 동적할당을 해제한다
fclose(fp); // 파일을 닫는다
return 0; // 종료한다

```

36. 인접 행렬을 출력 함수를 실행하고 동적할당 해제하고 파일을 닫은 후 종료한다

2.4 실행 결과

Microsoft Visual Studio 디버그 콘솔

```
강남의 인접 리스트 -> 교대 -> 양재  
양재의 인접 리스트 -> 매봉 -> 교대 -> 역삼 -> 강남  
역삼의 인접 리스트 -> 양재  
교대의 인접 리스트 -> 양재 -> 강남  
매봉의 인접 리스트 -> 양재
```

C:\Users\korca\Desktop\20204005_김필중_자료구조실습 7주
프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

data - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
강남 양재  
양재 강남  
양재 역삼  
역삼 양재  
강남 교대  
교대 강남  
교대 양재  
양재 교대  
매봉 양재  
양재 매봉
```

2.5 실행 결과

이번 과제는 그래프를 인접 리스트를 통해 표현하는 과제이다. 인접 리스트는 인접행렬과 달리 정점에서 연결된 것을 더 한눈에 보이는 것을 실행화면을 통해 확인이 가능했다. 다만 코드가 인접행렬보다 조금 더 복잡하고 생각해야할 부분이 더 생긴다는 단점도 존재했다. 인접행렬은 희소 그래프 표현에 적절하고 인접 리스트는 한눈에 보기 쉽게 만들 수 있는 요소에 적절하다고 생각한다. 최종적으로는 2개 모두 코드가 비슷하기 때문에 들어오는 주제에 따라 잘 선택해야 한다 생각한다. 역을 나타내는 것은 개인적으로는 어디로 연결되어있는지 보기 쉬운 인접 리스트가 적절하다 생각한다