기본 개념(복습)

◆ 자료구조(data structure)의 정의

- 프로그램에서 자료를 효율적으로 사용하기 위한 논리적인 구조
- 자료의 특성에 따라 분류하고 구성하여
  프로그램에서 활용할 수 있는 저장 및 처리의 체계를 제공
- 자료구조를 구현한 대표적인 예 : data type
- data type = object + operation
- C언어에서 제공하는 built-in type : char, int, float, double
- user-defined type : C의 typedef에 의하여 필요한 자료구조를 생성하여 활용
- program = data structure + algorithm

자료구조와 프로그래밍

◆ 자료 타입(Data Type)

- 기본 자료 타입

  자료 형태 + 연산

  int, foat, double, char. . .

  * java 등 고급 프로그램 언어에서 제공하는 자료타입을 알아보자.

- 군집 자료 타입

  배열(array)

  기본자료타입의 데이터가 여러 개인 경우

- 사용자 정의 자료 타입(구조체)

  C언어에서 제공하는 사용자 정의 자료 타입의 선언과 활용에 대하여 알아보자.

배열의 개념

배열 : 자료타입이 모두 같고 같은 이름으로 **참조**되는 데이터들의 집합

- 각 데이터들은 메모리 안에 인접한 위치를 차지한다.
- 배열의 각 요소를 구별하기 위하여 첨자(subscript) 사용

  cf. 구조체 : 서로 다른 데이터 형의 데이터를 묶어놓은 집합(레코드개념)

배열의 선언

type var_name[size]

예) int a[20];

이때 프로그램에서 색인(첨자)을 이용하여 20개의 각 변수를 사용

a[0], a[1], ... , a[19]

배열의 개념

배열의 대표적인 정보

배열의 이름 = 배열의 첫데이터의주소

a = &a[0]

a+i = &a[i]

***(a+i) = a[i]**

배열변수의 사용

첨자를 이용하여 각 변수를 처리하기 위하여 for문으로 데이터처리

```
sum = 0;
for (i=0; i < n; i++)
        sum = sum + a[i];
```

배열의 첨자 : for 반복문의 lcv

구조체

레코드

하나의 객체를 활용하기 위하여 정의된 여러 타입의 객체 속성으로 구성된 자료 구조

◆ 구조체의 개념

구조체 : 여러 형의 데이터를 하나의 객체로 선언하여 사용함

레코드 자료구조를 구현함

```
typedef struct member {
        int id;
        char name[20];
        float score;
} example;
```

에 의하여 struct member 라는 구조체 형이 선언됨
동시에 typedef 에 의하여 example이라는
데이터 형을 정의함

```
struct member {
    int id;
    char name[20];
    float score;
};
typedef struct member example;
```

구조체

◆ 구조체의 활용

example onep, exmember[20];

와 같은 선언문에 의하여 구조체를 프로그램에서 변수로 사용할 수 있게 됨.

```
onep.id = 2102;
strcpy(onep.name, “Hong Gildong” );
onep.score = 4.23;
```

// exmember중에 성적이 4.00이상인 학생의 id를 출력하시오.

```
for (k=0; k < 20; k++)
        if (exmember[k].score >= 4.0)
            printf( “%d\n” , exmember[k].id);
```

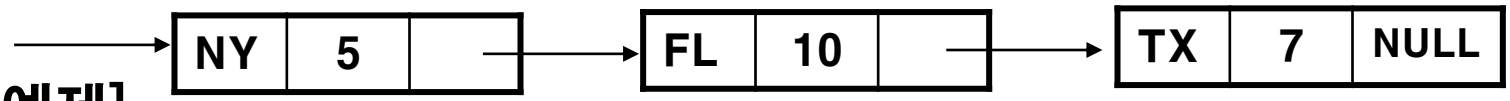
단순 연결 리스트의 구현

◆ 연결 리스트를 생성하기 위해 필요한 기능

(1) 노드의 구조 정의 : 자기참조구조체

(2) 노드 생성 : malloc() 함수 사용

(3) 노드의 데이터 필드와 링크 필드에 값을 할당



[연결리스트예제]

```
typedef struct ex_node *expointer;
struct exnode {
        char state[3];
        int sdata;
        expointer nlink;
};
```

단순 연결 리스트의 구현 예제(1)

```
typedef struct exnode *expointer;
struct exnode {
        char state[3];
         int sdata
        expointer nlink;
};

void make_list1(expointer inode)
{
    expointer ptr=head, before;

    while (ptr != NULL) {
        before = ptr;
        ptr = ptr -> nlink;
    }
    before -> nlink = inode;
    inode -> nlink = NULL;
}
```

```
main()
{
        expointer inode;
        int k;

        head = (expointer) malloc(sizeof(struct exnode));
        strcpy(head->state, "NY");
        head->sdata = 5;
        head->nlink = NULL;

        for (k=0; k < 3; k++){
            inode = (expointer) malloc(sizeof(struct exnode));
            printf("Enter state_name and order : ");
            scanf("%s %d", (inode->state), &(inode->sdata));
            make_list1(inode); // 함수를 call하여 노드를 연결한다
        }
        printf("======================\n");
        print_list(); //print_list 함수를 call하여 출력한다
}
```

```
H:₩[1]수업2020-2학기_0920₩자료구조수업자료₩[현재]자료구조2020₩11-12주차_트리₩state_llist.exe
Enter state_name and order :  CA 15
Enter state_name and order :  TX 17
Enter state_name and order :  FL 10
======================
The singly linked list contains :
NY  :  5
CA  :  15
TX  :  17
FL  :  10
```

단순 연결 리스트의 구현 예제(2)

```
typedef struct exnode *expointer;
struct exnode {
        char state[3];
         int sdata
        expointer nlink;
};


 void make_list2(expointer inode)
 {
     inode->nlink = head;
     head = inode;
 }
```

```
main()
{
      expointer inode;
      int k;

      head = (expointer) malloc(sizeof(struct exnode));
      strcpy(head->state, "NY");
      head->sdata = 5;
      head->nlink = NULL;

      for (k=0; k < 3; k++){
          inode = (expointer) malloc(sizeof(struct exnode));
          printf("Enter state_name and order : ");
          scanf("%s %d", (inode->state), &(inode->sdata));
          make_list1(inode); // 함수를 call하여 노드를 연결한다
      }
      printf("======================\n");
      print_list(); //print_list 함수를 call하여 출력한다
}
```

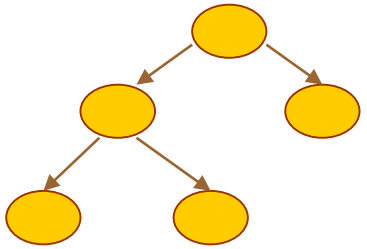
```
H:₩[1]수업2020-2학기_0920₩자료구조수업자료₩[현재]자료구조2020₩11-12주차_트리₩state_llist.exe
Enter state_name and order :  CA 15
Enter state_name and order :  TX 17
Enter state_name and order :  FL 10
======================
The singly linked list contains :
FL  :  10
TX  :  17
CA  :  15
NY  :  5
```

트리의 개념

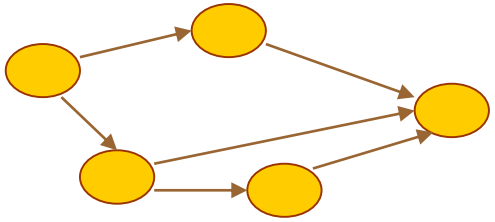
◆ 트리(tree)

- 구성하고 있는 데이터들 간에 1:多 관계를 가지는 비선형 자료구조
- 레벨의 개념을 가지는 계층형 자료구조

선형구조



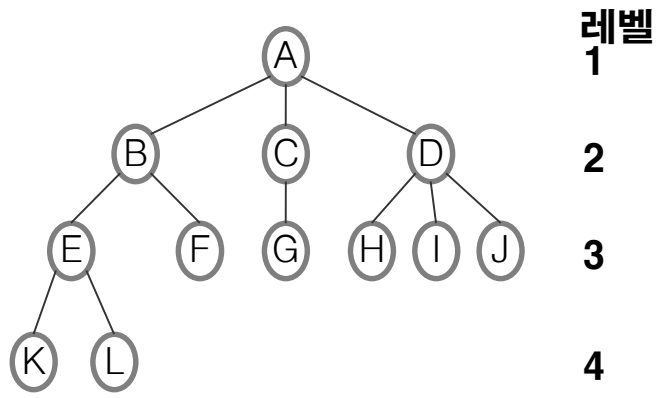
트리구조



그래프구조 : 노드와 노드사이의 연결선의 집합

트리의 정의

- 그래프의 특수한 형태
- 두 정점(노드) 사이에 순환(cycle)이 존재하지 않는 연결 그래프 (connected graph)
- 두 노드 사이에 하나의 경로만 존재
- 다음 조건을 만족하는 하나이상의 노드 집합으로 이루어짐
  (1) 하나의 루트노드가 있으며
  (2) 나머지 노드들은 n≥0 개의 서브트리(subtree) T1, … , Tn으로 구성되어 있다.



트리 관련용어

- 차수

✓ 노드의 차수 :

노드에 연결된 자식 노드의 수.

어떤 노드의 서브트리수

A의 차수=3, K의 차수=1, P의 차수=0

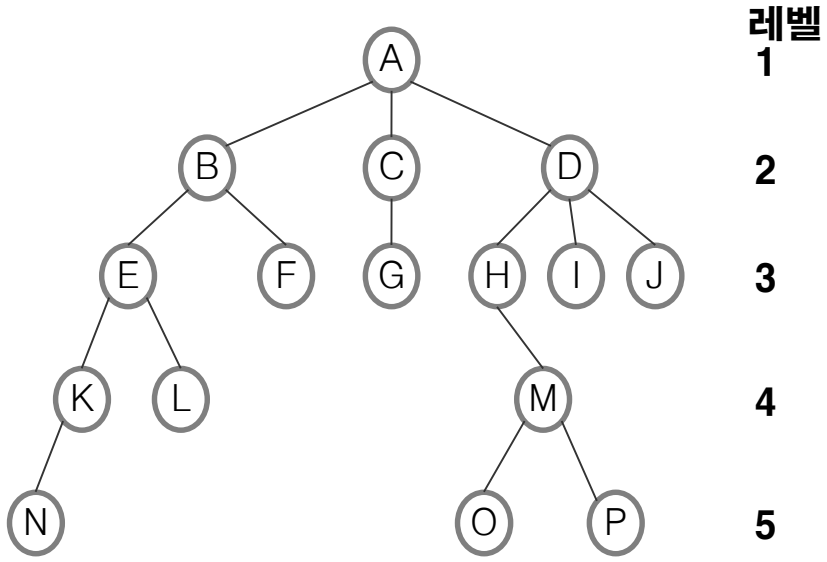
✓ 트리의 차수 :

트리에 있는 노드의 차수 중에서 가장 큰 값

트리의 차수=3

✓ 단말 노드(leaf 노드, terminal 노드) :

차수가 0인 노드. 자식 노드가 없는 노드

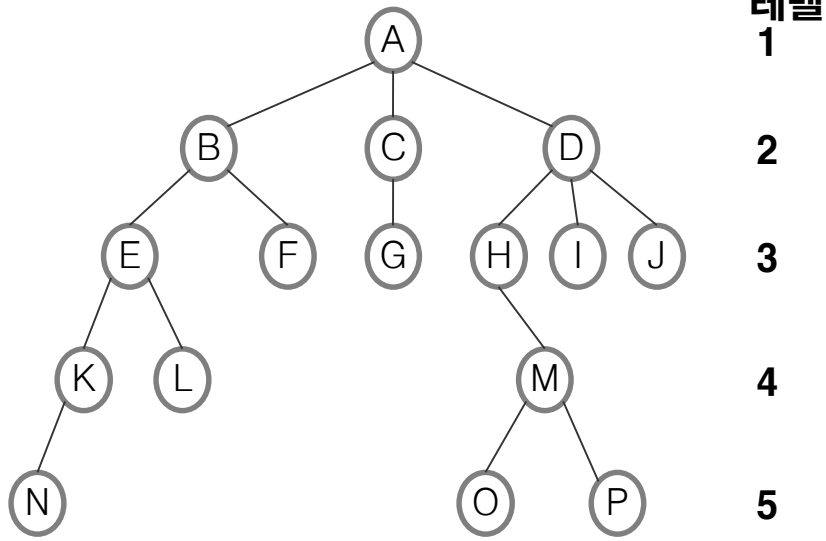


트리 관련용어

- H의 sibling은 I,J (부모가 같은 노드)
- M의 ancestor (A, D, H)
- B의 descendant (E F K L N)
- 높이 height(깊이 depth) =
  트리에 속한 노드의 최대 레벨 (루트레벨 1로 가정)
  - ✓ 노드의 레벨 = 부모노드 레벨 + 1 (루트 제외)

    B의 level=2, F의 level=3
  - ✓ 트리의 높이(깊이)
    트리에 있는 노드의 최대 레벨

    트리의 높이=5



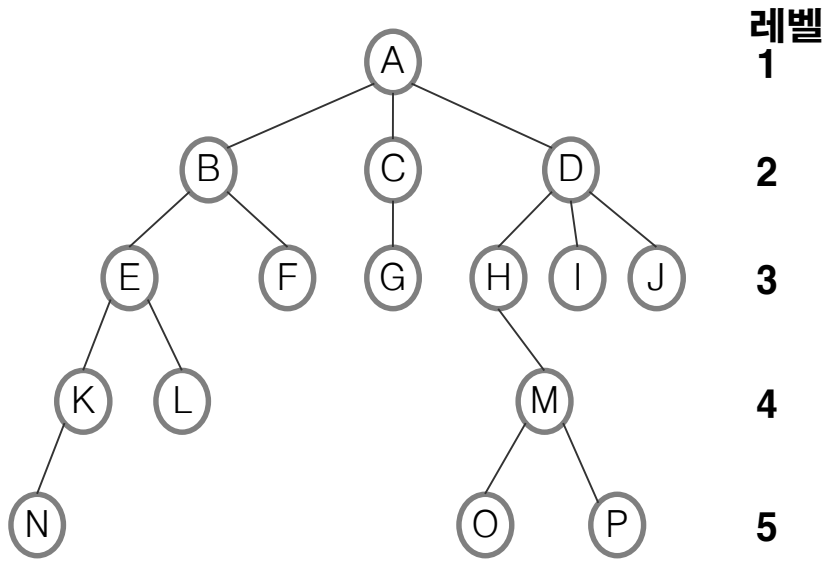
트리의 표현방법

(a) 트리표현

(b) 리스트표현

(c) 집합표현

(d) 들여쓰기 표현



(A (B (E (K(N), L), F), C(G), D(H (M(O, P)), I, J)))

트리의 표현방법

◆ 트리를 기억장소에 표현 → 프로그램에서 활용 가능한 자료

• 연결 리스트로 표현 → 자기참조구조체로 구현

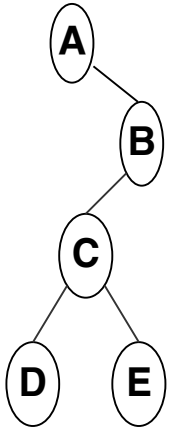
- 한 개의 노드에는 차수만큼의 링크가 필요
- 각 노드마다 서로 다른 링크필드의 수를 갖는다. → 프로그램 복잡도 증가
- 트리 차수만큼 링크필드를 구현 → null link가 많아서 기억장소 낭비

➔ 링크가 2개로 고정되고 프로그래밍의 이진논리와도 맞는 이진트리를 주로 사용

데이터	링크1	링크2	……	링크n

이진트리의 개념

◆ 이진트리

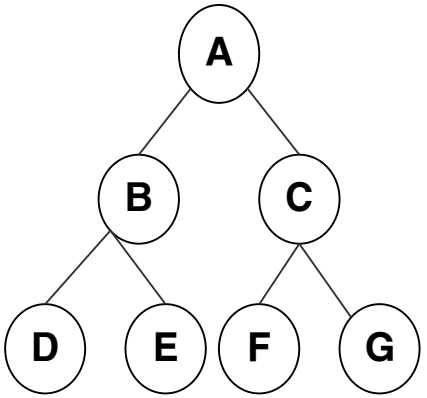


- 트리의 노드 구조의 link수를 2개로 제한 하여
  일반 트리의 구현과 연산이 쉽도록 정의한 트리
- 공집합이거나 루트와 왼쪽 서브트리, 오른쪽 서브트리로
  구성된 노드의 유한 집합
  - 왼쪽 서브트리와 오른쪽 서브트리도 또한 분리된 이진트리
  - 이진 트리의 모든 노드는 왼쪽 자식 노드와 오른쪽 자식 노드 만을 가진다.
  - 공백 노드도 자식 노드로 취급한다.
  - $0 \leq$ 노드의 차수 ≤ 2

이진트리의 종류

◆ 이진 트리의 종류

- 포화 이진 트리 (full binary tree)
  - 모든 레벨에 노드가 포화상태로 차 있는 이진 트리
  - 트리 구성하는 노드의 차수가 0 또는 2인 트리
  - 높이가 h일 때, 최대의 노드 개수인 (2^h-1)개의 노드를 가진 이진 트리
  - 레벨 i에서 노드의 수 : 2^{i-1}

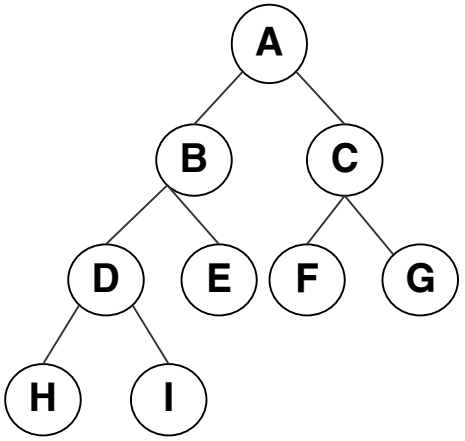


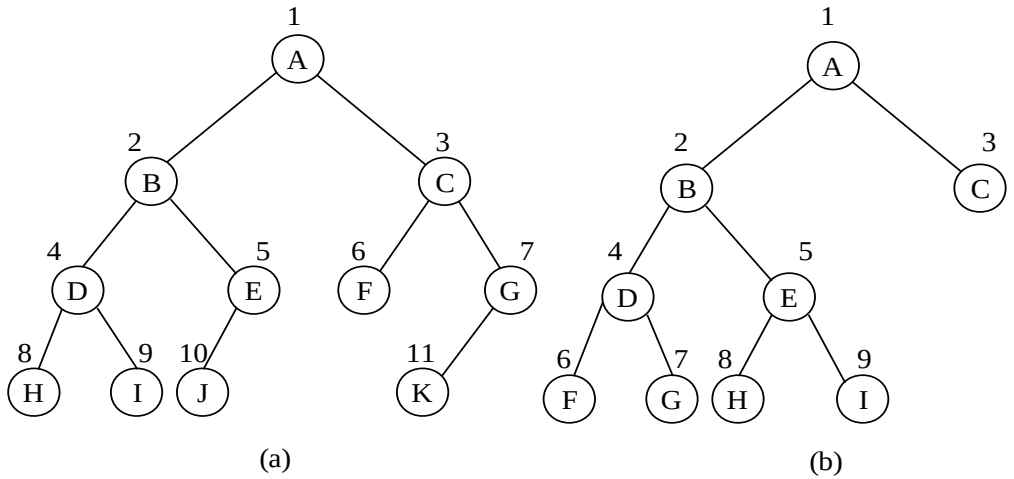
이진트리의 종류

◆ 이진 트리의 종류

- 완전 이진 트리 (complete binary tree)
  - 높이가 h이고 노드 수가 n개일 때

    포화 이진 트리의 노드 번호 1번부터 n번까지 빈 자리가 없는 이진 트리



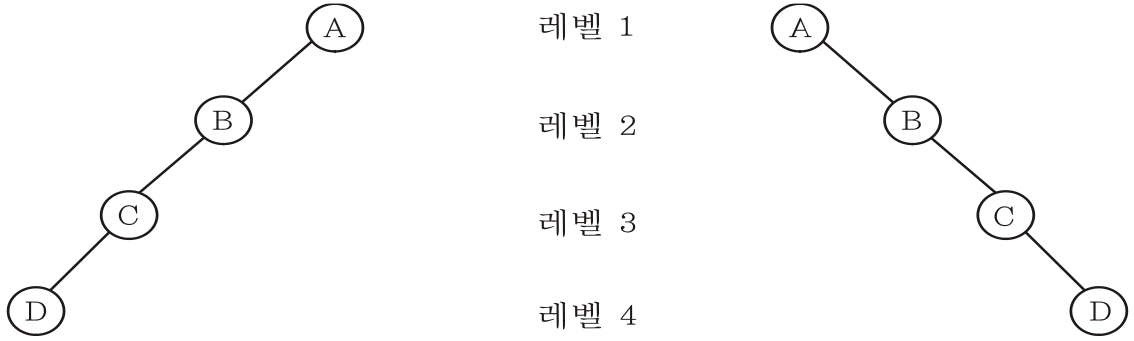


완전 이진 트리가 아닌 경우

이진트리의 종류

◆ 이진 트리의 종류

- 사향 (편향) 이진 트리 (skewed binary tree)
  - 한쪽 방향의 자식 노드만을 가진 이진 트리
  - 왼쪽 사향 이진 트리
    - ✓ 모든 노드가 왼쪽 자식 노드만을 가진 사향 이진 트리
  - 오른쪽 사향 이진 트리
    - ✓ 모든 노드가 오른쪽 자식 노드만을 가진 사향 이진 트리



좌사향이진트리 우사향이진트리

노드의 수가 6인
왼쪽 사향이진트리의
깊이는?