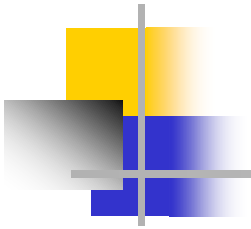
컴퓨터프로그래밍
보충 자료

최광훈

연세대학교 원주캠퍼스
컴퓨터정보통신공학부



목차

- 재귀 함수
- 포인터
- 전 처리기
- 식과 문장

재귀 함수 – 귀납법

- 귀납법(induction) 증명의 예
  - 모든 n에 대해 명제 P(n)이 참임을 증명
    - " 1부터 n까지의 합은 n(n+1)/2 "
    - P(n) : $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
  - n=1일 때 : $1 = \frac{1(1+1)}{2}$ (cf. P(1)이 참)
  - n=k일 때 : 이 명제가 성립한다고 가정하고 n=k+1일 때도 성립함을 보임 (cf. P(k)가 참이면 P(k+1)도 참)
    - $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ 을 참이라고 가정하고,
    - $\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$
- 귀납법(induction) 증명
  - 모든 n에 대해 주어진 명제 P(n)이 참임을 아래의 단계로 증명
    - P(1)이 참임을 증명하고, (base case)
    - P(k)가 참이라 가정하고 P(k+1)이 참임을 증명 (inductive case)
      - cf. P(k) => P(k+1)을 증명

재귀 함수

- 재귀함수와 귀납법(induction)의 유사성 활용
  - 1부터 N까지 합을 구하는 함수 sum(N)을 작성하는 예

sum(N)=N(N+1)/2임을 증명

N=1)
  sum(1) = 1*2/2 = 1

N=k-1)
  sum(k-1)=(k-1)((k-1)+1)/2
  라고 가정하면,
N=k)
  sum(k)=sum(k-1)+k
  = (k-1)k/2+k
  = k(k+1)/2 가 성립

따라서 모든 N에 대해
sum(N)=N(N+1)/2이다.

```
int sum(N) {

  if (N == 1)
      return 1;

  else {
    /* 함수 호출 sum(N-1)이
       1부터 N-1까지의 합을
       리턴 한다 가정하고,  */
      return sum(N-1) + N
   }
}
```

따라서 모든 N에 대해
sum(N)은 1~N까지 합이다.

재귀 함수

- 재귀함수를 이해하는 한 가지 요령
  - 재귀함수 안에서 “base case”와 “inductive case”를 구분해서 파악
    - cf. 귀납법에서 n=1, n=k의 경우 구분
  - “inductive case”에서 이뤄지는 재귀함수 호출에서 결과 값이 제대로 반환된다고 가정하고 살펴봄
    - cf. 귀납법에서 k의 경우에 성립한다고 가정하고 k+1의 경우도 성립하는지 살펴봄

- 아래 재귀함수 예제를 위의 요령으로 이해하기
  - 재귀함수 fib (교재 5장, 종합문제 8번)
  - 재귀함수 gcd (교재 5장 9절, 예제 5.19번)
  - 재귀함수 printdigit (교재 5장 9절, 예제 5.20번)
  - 재귀함수 hanoi (교재 5장 8절, 예제 5.13번)

포인터

- 포인터 자료형의 *를 변수 이름 앞이 아닌 자료형 뒤에 붙임
  - int *p;
    - int *p; ===> int* p;
  - int main(int argc, char *argv[]) { ... }
    - char *argv[] ===> char* argv[]
- 포인터 규칙과 오류 여부 판단
  - e의 자료형이 T ===> &e의 자료형은 T* (e는 식, T는 자료형)
    - 예: int x; x의 자료형이 int ===> &x의 자료형은 int*
  - e의 자료형이 T* ===> *e의 자료형은 T
    - 예: int* p; p의 자료형이 int* ===> *p의 자료형은 int
  - 배열 타입은 포인터 타입과 동일 (cf. const 수정자의 유무는 다름)
    - argv는 char*[] 타입 또는 char** 타입
    - *argv는 char* 타입, **argv는 char 타입
  - 치환문(=)의 오른쪽과 왼쪽의 자료형은 동일해야 함
    - p = 10; => p는 int* 타입, 10은 int 타입.
    - printf("%c", argv[0]); => %c는 char 자료형, argv[0]은 char* 자료형.

전처리기

- 전처리기를 통해 MyFile.c가 MyFile.i로 변환 (교재 11장에서 발췌)

MyFile.c

```
#define PI 3.141592
#define PRTERR() \
      printf("에러가 발생\n");
#define LOOP while (1)

a = 2 * PI * r;
PRTERR();
LOOP { ... }

#define mul(a,b) (a)*(b)
#define putint(x) \
       printf("%d", (x));

x = mul(100,n);
y = mul(a+b,c-d);
putint(d);
```

MyFile.i

```
a = 2 * 3.141592 * r;
printf("에러가 발생\n");
while(1) { ... }

x = (100) * (n);
y = (a+b) * (c-d);
printf("%d", (d));
```

식과 문장

❑ 식과 문장의 유사점과 차이점

- 유사점 : 식에 ;을 붙이면 문장이 됨
  - 예) x = 1 ; // 치환문 ===> 치환식에 ;을 붙임
  - 예) f() ; // 함수 호출문 ===> 함수 호출 식에 ;을 붙임
- 차이점 : 식을 계산하면 반드시 결과값이 있음
  - 예) y = x = 3; // 식 “x = 3”의 결과값은 3
  - 예) y = x == 10 ; // 식 “x == 10”의 결과값은 0 또는 1

- 결과 값과 Side effect는 서로 독립적인 개념
  - x == 1 : 식의 결과값은 0 또는 1, x의 값은 불변(side effect는 없음)
  - x = 1 : 치환식의 결과값은 1, x에 1을 할당(side effect)
  - x++ : 결과값은 변수 x의 값, x의 값을 1 증가(side effect)
    - 예: x=7; (x++) + 1 ===> 결과값 8, x의 값 8
  - ++x : 결과값은 x+1의 값, x의 값을 1 증가 (side effect)
    - 예: x=7; (++x) + 1 ===> 결과값 9, x의 값 8