COMP319 Algorithms
Lecture 2
Introduction to Algorithms
Basic Algorithms

Introduction to Algorithms

Summation / Product / Recursion / Search

Instructor: Gil-Jin Jang

Original slides:

이지영, 한빛아카데미(주), C로 배우는 쉬운 자료구조;

Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜, C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

개요: 알고리즘

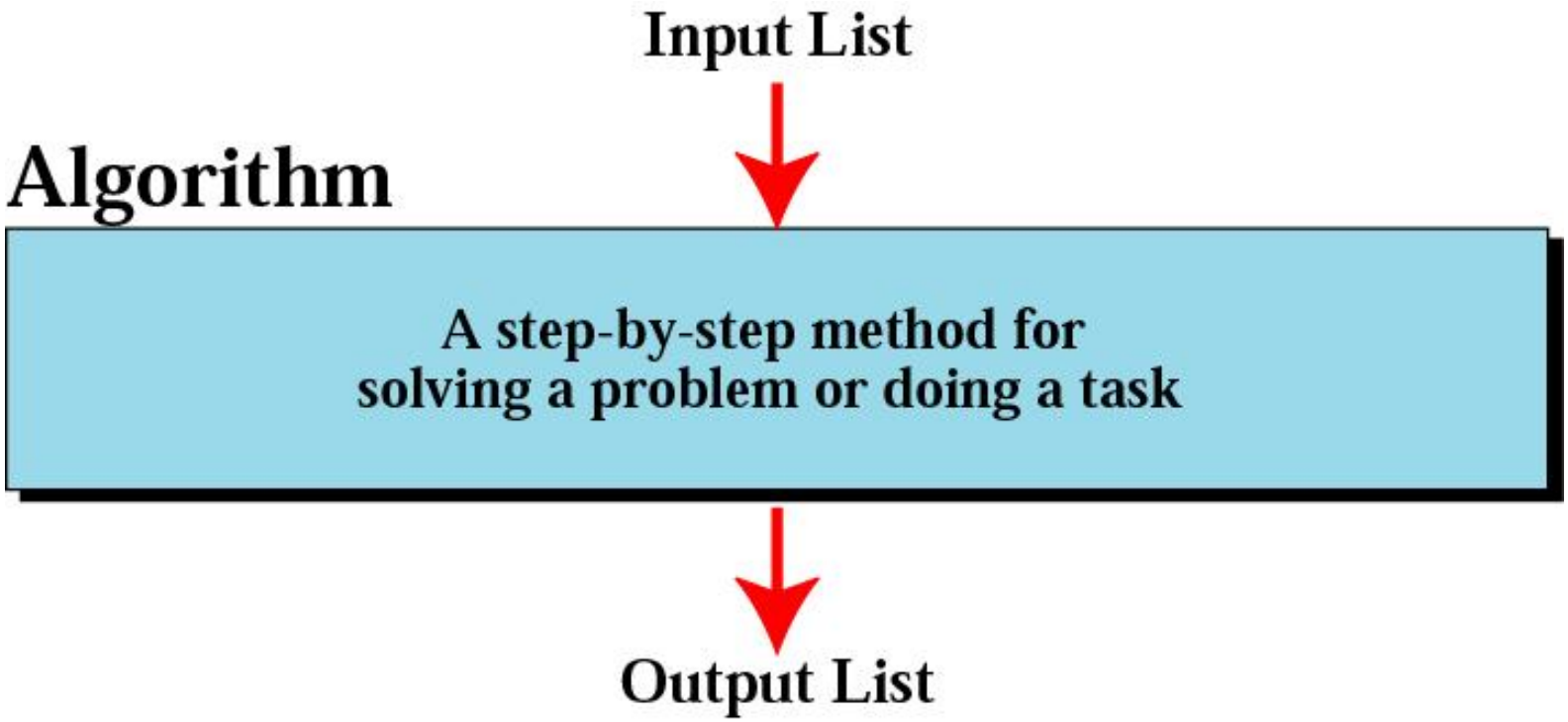
이지영, 한빛아카데미(주),
C로 배우는 쉬운 자료구조

알고리즘의 이해

알고리즘의 표현방법

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

알고리즘의 정의



Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

알고리즘의 이해

❖ 알고리즘

- 문제해결 방법을 추상화하여 단계적 절차를 논리적으로 기술해 놓은 명세서

❖ 알고리즘의 조건

- 입력input : 알고리즘 수행에 필요한 자료가 외부에서 입력으로 제공될 수 있어야 한다.
- 출력output : 알고리즘 수행 후 하나 이상의 결과를 출력해야 한다.
- 명확성definiteness : 수행할 작업의 내용과 순서를 나타내는 알고리즘의 명령어들은 명확하게 명세되어야 한다.
- 유한성finiteness : 알고리즘은 수행 뒤에 반드시 종료되어야 한다.
- 효과성effectiveness : 알고리즘의 모든 명령어들은 기본적이며 실행이 가능해야 한다.

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

알고리즘의 이해

요리 재료 — **자료**

케이크 시트(20cm×20cm) 1개, 크림치즈 무스(크림치즈 200g, 달걀 2알, 설탕 3큰술, 레몬즙 1큰술, 바닐라 에센스 1큰술), 딸기 시럽(딸기 500g, 설탕 1½컵, 레몬즙 1작은술), 딸기 1개, 플레인 요구르트 2큰술

알고리즘

요리법 — **절차**

1. 케이크 틀에 유산지를 깔고 케이크 시트를 놓는다.
2. 달걀 2알을 잘 푼다. 볼에 크림치즈를 넣고 거품기로 젓는다. 달걀 푼 물과 설탕 3큰술을 세 차례로 나누어 넣으면서 크림 상태가 되도록 거품기로 젓는다.
3. ❷에 레몬즙과 바닐라 에센스를 넣고 살짝 저은 다음 ❶에 붓는다. 180℃로 예열된 오븐에 전체를 넣고 20분 정도 굽는다.
4. 딸기를 얇게 자르고 냄비에 넣은 다음 설탕 1½컵을 넣고 약한 불로 끓인다. 눌어붙지 않도록 계속해서 젓고 거품이 생기면 걷어 낸다. 되직해지면 레몬즙을 넣고 차갑게 식힌다.
5. 치즈케이크 한 조각을 접시에 담고 ❹를 뿌린 다음 플레인 요구르트와 딸기를 얹는다.

연산

그림 1-26 딸기 시럽을 얹은 치즈케이크 만들기

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

알고리즘의 이해

자료

[**요리 재료**]

스펀지케이크(20×20cm) 1개, 크림치즈 200g, 달걀 푼 물 2개 분량, 설탕 3큰술, 레몬즙·바닐라에센스 1큰술씩, 딸기시럽(딸기 500g, 설탕 1½ 컵, 레몬즙 1작은술), 딸기 1개, 플레인 요구르트 2큰술

[**요리법**] >> 알고리즘

절차

① 케이크 틀의 가장자리에 필름을 돌린 다음 스펀지케이크를 놓는다.

② 볼에 크림치즈를 넣고 거품기로 젓다가 달걀 푼 물과 설탕 3큰술을 세번에 나누어 넣으면서 크림 상태로 만든다.

③ ②에 레몬즙과 바닐라에센스를 넣고 살짝 저은 다음 ①에 붓는다. 이것을 180℃의 오븐에 넣고 20분 정도 굽는다.

④ 냄비에 슬라이스한 딸기와 설탕 1½ 컵을 넣고 끓이다가 약한 불에서 눌어붙지 않도록 저으면서 거품을 걷어낸다. 되직해지면 레몬즙을 넣고 차게 식힌다.

⑤ 접시에 치즈케이크를 한 조각 담고 ④의 시럽을 뿌린 다음 플레인 요구르트와 딸기를 얹어낸다

연산

알고리즘의 표현 방법

- 알고리즘의 표현 방법의 종류
  - 자연어를 이용한 서술적 표현 방법
  - 순서도Flow chart를 이용한 도식화 표현 방법
  - 프로그래밍 언어를 이용한 구체화 방법
  - 가상코드Pseudo-code를 이용한 추상화 방법

알고리즘의 표현 방법

- 순서도Flow chart를 이용한 도식화
  - 순서도Flow chart의 예) 1부터 5까지의 합을 구하는 알고리즘

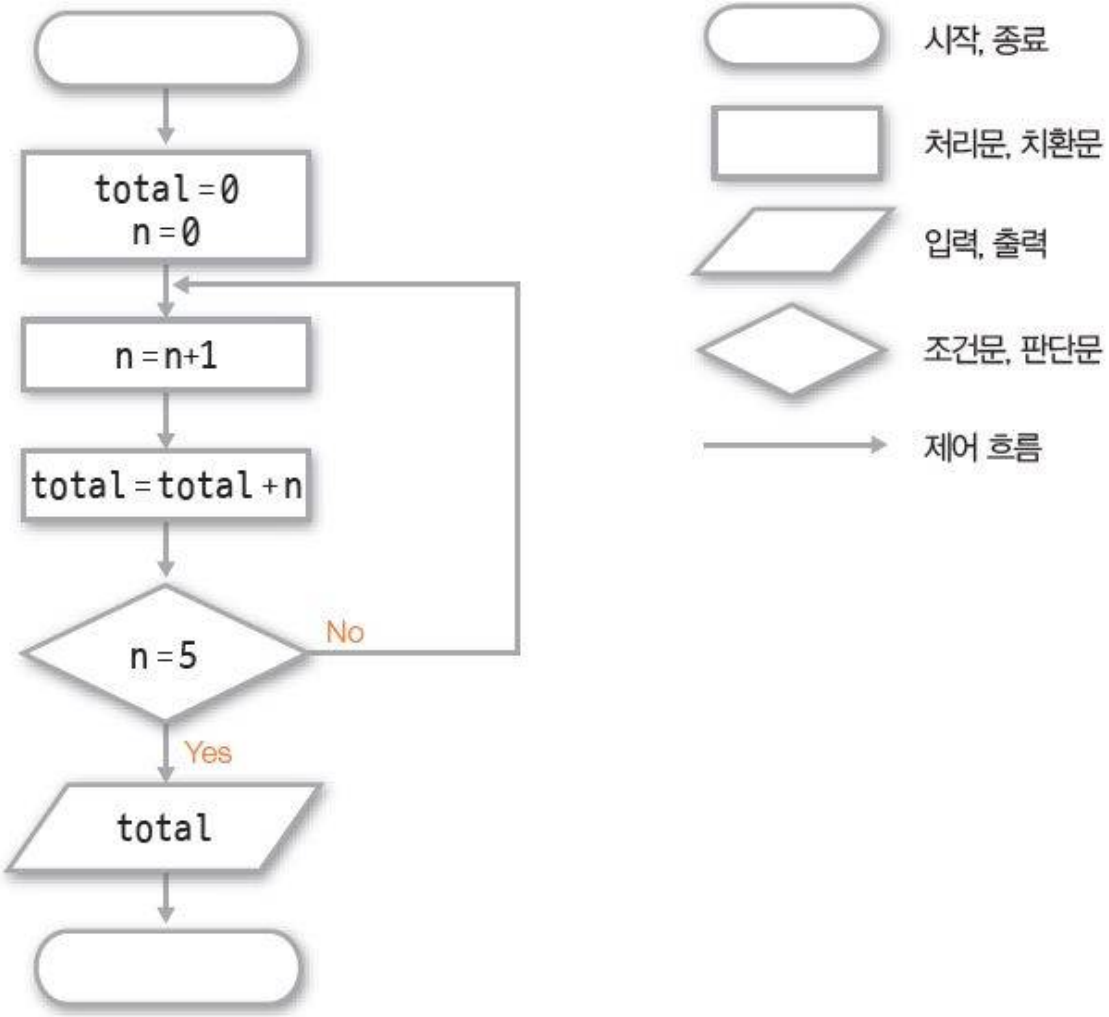


그림 1-27 순서도의 예

알고리즘의 표현 방법

- 가상코드Pseudo-code 를 이용한 추상화
  - 가상코드Pseudo-code, 즉 알고리즘 기술언어ADL, Algorithm Description Language를 사용하여 프로그래밍 언어의 일반적인 형태와 유사하게 알고리즘을 표현
  - 특정 프로그래밍 언어가 아니므로 직접 실행은 불가능
  - 일반적인 프로그래밍 언어의 형태이므로 원하는 특정 프로그래밍
    언어로의 변환 용이

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

알고리즘의 표현 방법

❖ 가상코드Pseudo-code의 형식(기본 요소)

- 기호
  - 변수, 자료형 이름, 프로그램 이름, 레코드 필드 명, 문장의 레이블 등을 나타냄
  - 문자나 숫자의 조합. 첫 문자는 반드시 영문자 사용.
- 자료형
  - 정수형과 실수형의 수치 자료형, 문자형, 논리형, 포인터, 문자열 등의 모든 자료형 사용
- 연산자
  - 산술연산자, 관계연산자, 논리연산자

■ 지정문 형식과 예

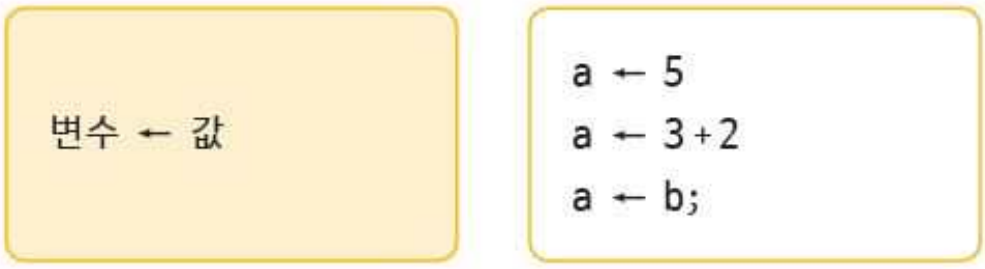
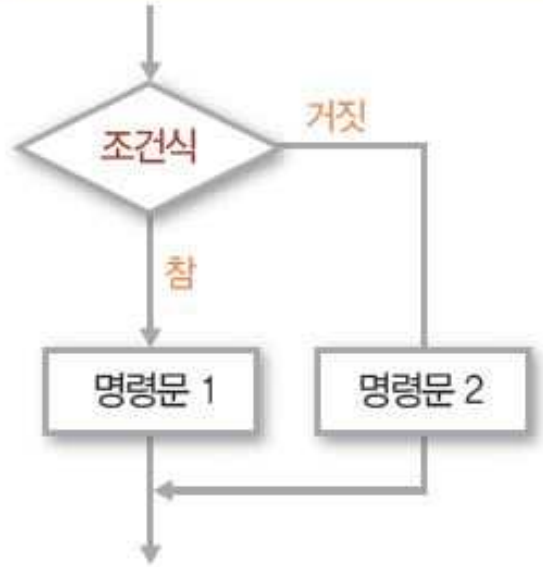


그림 1-28 지정문 형식과 예

조건문

- 조건에 따라 실행할 명령문이 결정되는 선택적 제어구조
- if 문의 형식과 제어흐름

```
if (조건식) then 명령문 1;
else 명령문 2;
```



(a) if-then-else 형

```
if (조건식) then 명령문 1;
```

(b) if-then 형

그림 1-29 기본 if 문의 형식과 제어 흐름

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

다단계 조건문

- 중첩 if 문의 형식과 제어 흐름

```
if (조건식 1) then 명령문 1;
else if (조건식 2) then 명령문 2;
else 명령문 3;
```

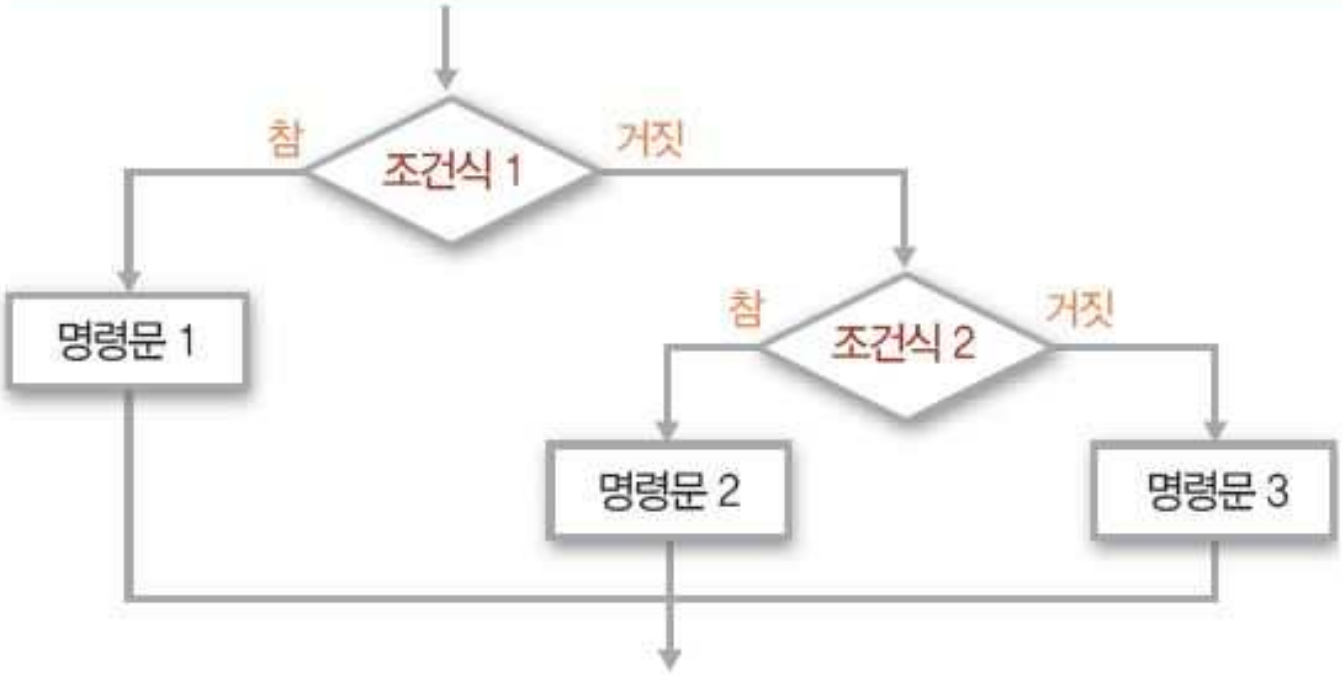


그림 1-30 중첩 if 문의 형식과 제어 흐름

중첩 if문 사용 예

```
if Average >= 90 then grade ← "A";
else if Average >= 80 then grade ← "B";
else if Average >= 70 then grade ← "C";
else grade ← "F";
```

평균 점수에 따른 등급 계산하기

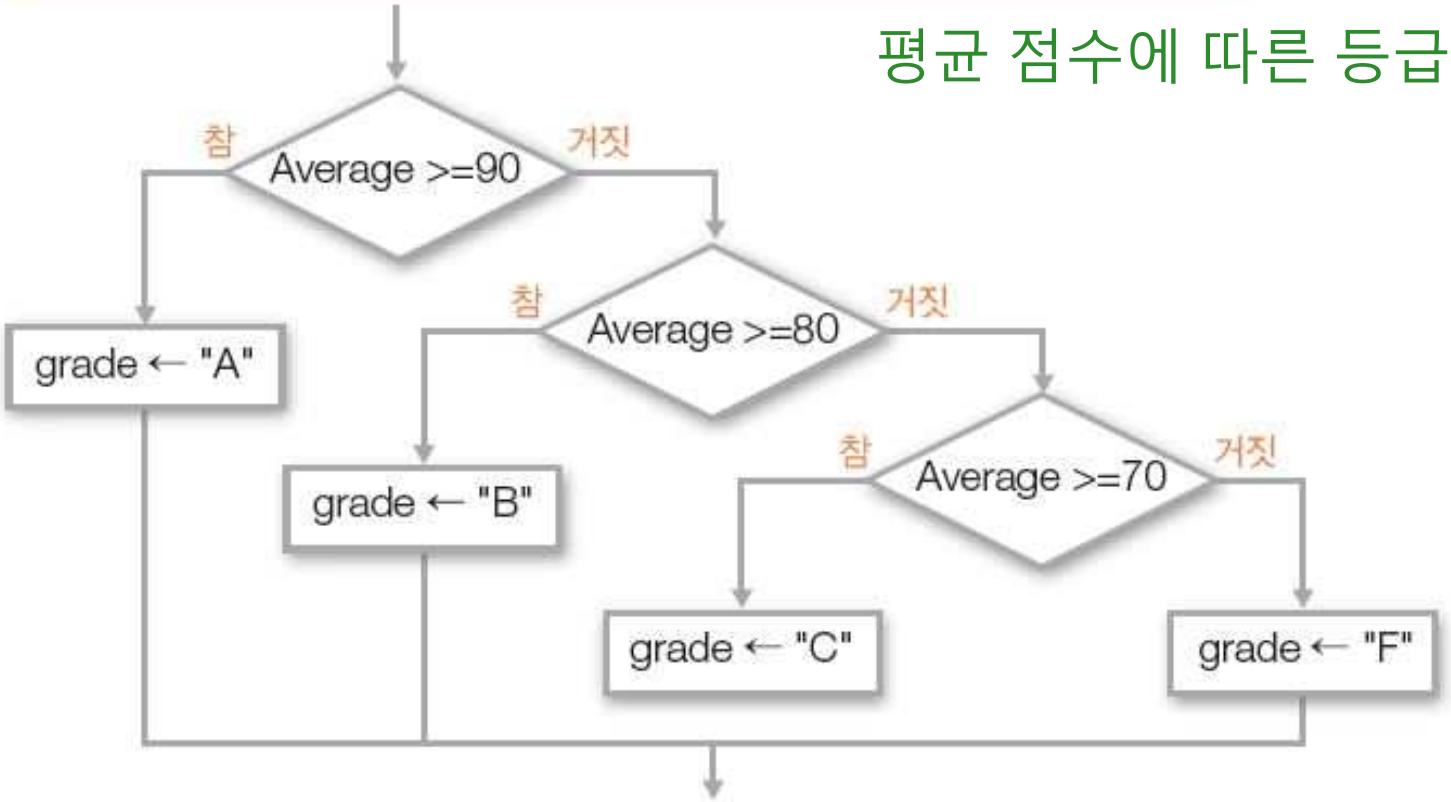


그림 1-31 중첩 if 문의 예

Slide credit
문병로, 쉽

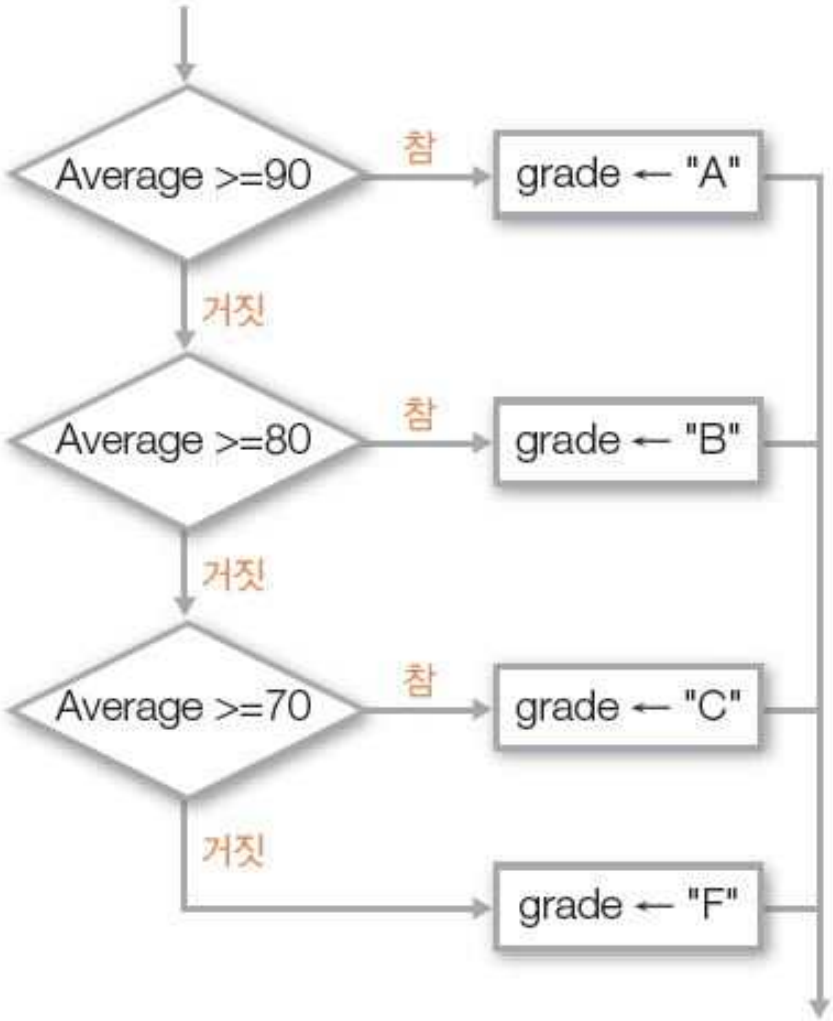
Case 문

- 여러 조건식 중에서 해당 조건을 찾아서 그에 대한 명령문을 수행
- 중첩 if 문으로 표현 가능
- 형식과 제어흐름

```
case {
    조건식 1 : 명령문 1;
    조건식 2 : 명령문 2;
    ...
    조건식 n : 명령문 n;
    else    : 명령문 n+1;
}
```

그림 1-32 case 문의 형식과 제어 흐름

case 문 예) 평균 점수에 따른 등급 계산하기

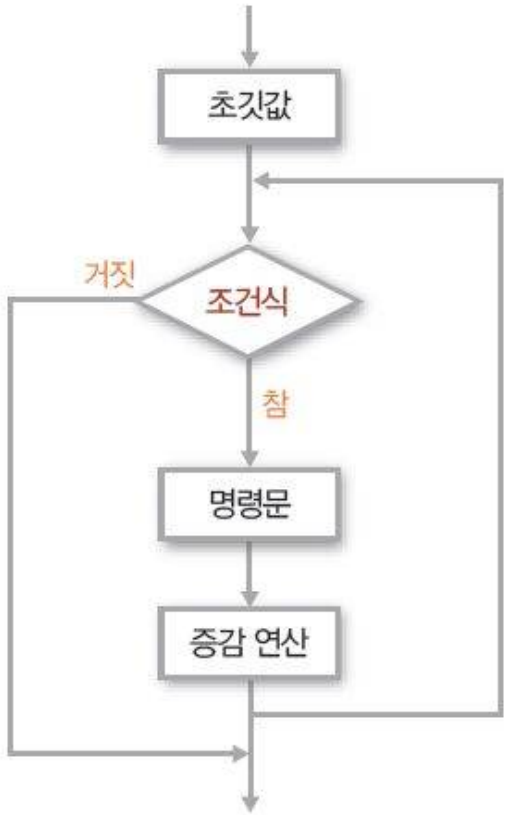



```
case {
    Average >= 90 : grade ← "A";
    Average >= 80 : grade ← "B";
    Average >= 70 : grade ← "C";
    else :          grade ← "F";
}
```

그림 1-33 case 문의 예

반복문

- 일정한 명령을 반복 수행하는 루프(loop) 형태의 제어구조
- for 문




```
for (초깃값; 조건식; 증감값) do 명령문;
```

그림 1-34 for 문의 형식과 제어 흐름

while-do 문

```
while (조건식) do 명령문;
```

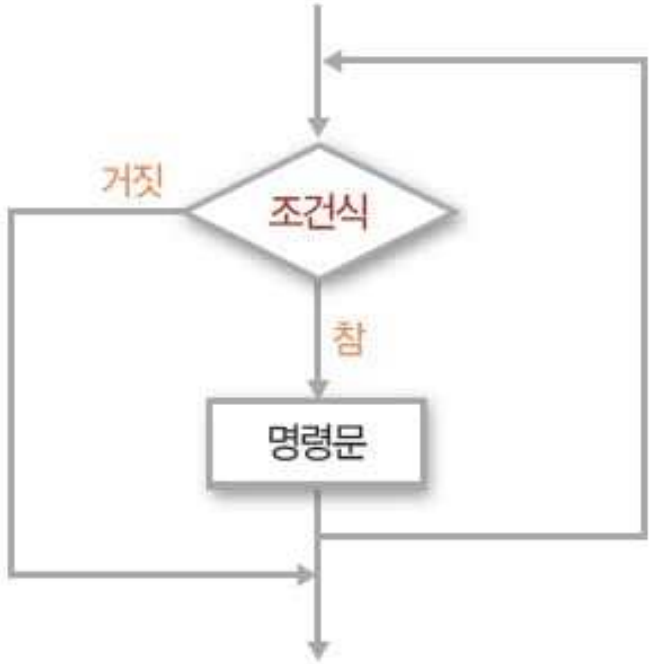


그림 1-35 while-do 문의 형식과 제어 흐름

do-while 문

```
do 명령문;
while (조건식);
```

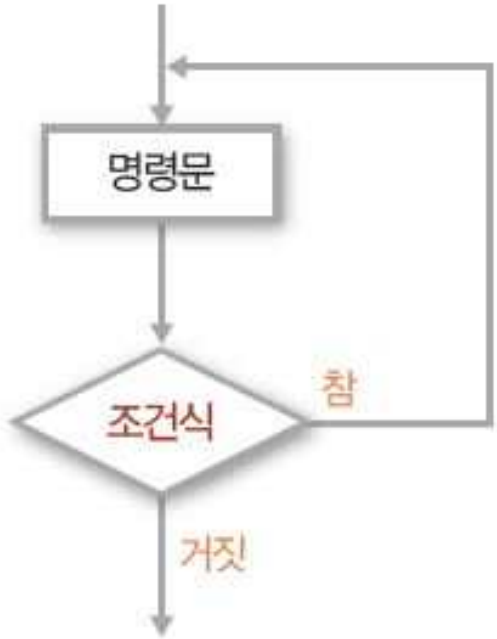


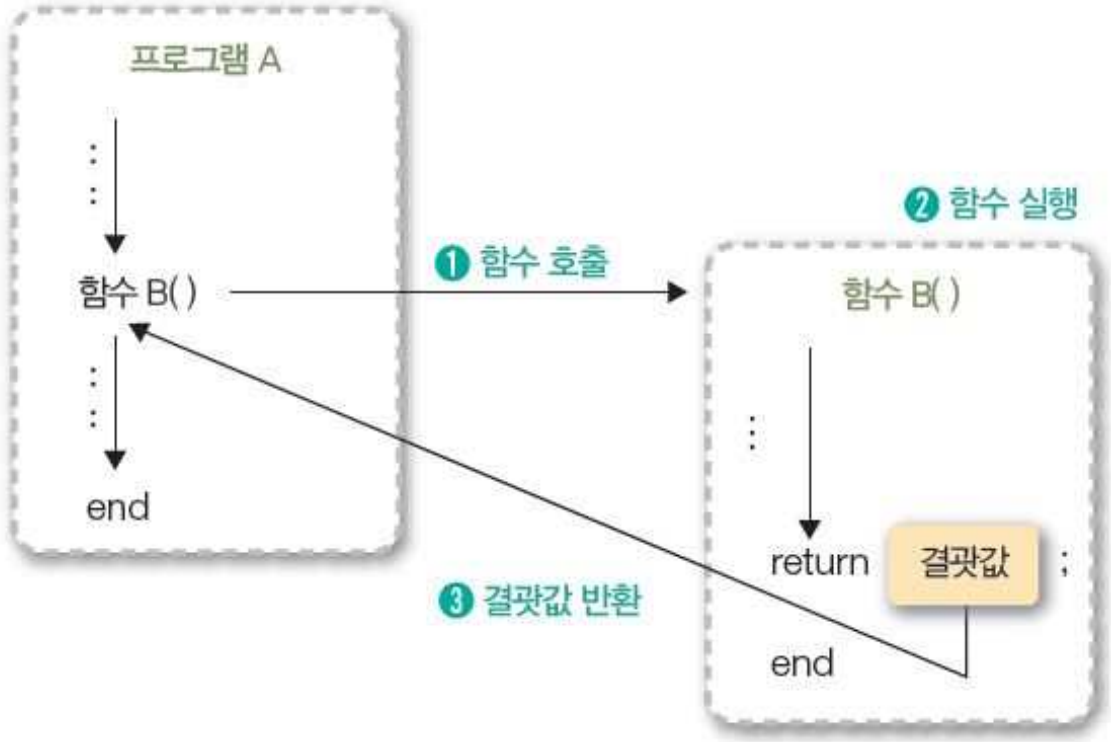
그림 1-36 do-while 문의 형식과 제어 흐름

함수문

❖ 처리작업 별로 모듈화하여 만든 부프로그램

```
함수 이름 (매개변수)
    명령문;
    ...
    return 결괏값;
end
```

(a) 함수의 형식



(b) 함수의 호출과 실행 및 결괏값 반환의 예

그림 1-37 함수의 형식과 예

Example 1

Write an algorithm in pseudocode that finds the average of two numbers

Solution

See Algorithm 8.1 on the next slide.

Algorithm 8.1: Average of two

AverageOfTwo
Input: Two numbers
1. *Add the two numbers*
2. *Divide the result by 2*
3. *Return the result by step 2*

End

Example 2

Write an algorithm to change a numeric grade to a pass/no pass grade.

Solution

See Algorithm 8.2 on the next slide.

Algorithm 8.2: Pass/no pass Grade

Pass/NoPassGrade
Input: One number
1. *if (the number is greater than or equal to 70) then*
    1.1 Set the grade to "pass"
   else
    1.2 Set the grade to "nopass"
   End if
2. *Return the grade*

End

Example 3

Write an algorithm to change a numeric grade to a letter grade.

Solution

See Algorithm 8.3 on the next slide.

Algorithm 8.3: Letter grade

LetterGrade
***Input:* One number**
1. if (the number is between 90 and 100, inclusive) then
1.1 Set the grade to "A"
End if
2. if (the number is between 80 and 89, inclusive) then
2.1 Set the grade to "B"
End if

Continues on the next slide

Algorithm 8.3: Letter grade (continued)

3. if (the number is between 70 and 79, inclusive)
then
3.1 Set the grade to "C"
End if
4. if (the number is between 60 and 69, inclusive)
then
4.1 Set the grade to "D"
End if

Continues on the next slide

Algorithm 8.3: Letter grade (continued)

5. If (the number is less than 60)
then
5.1 Set the grade to "F"
End if
6. Return the grade
End

Example 4

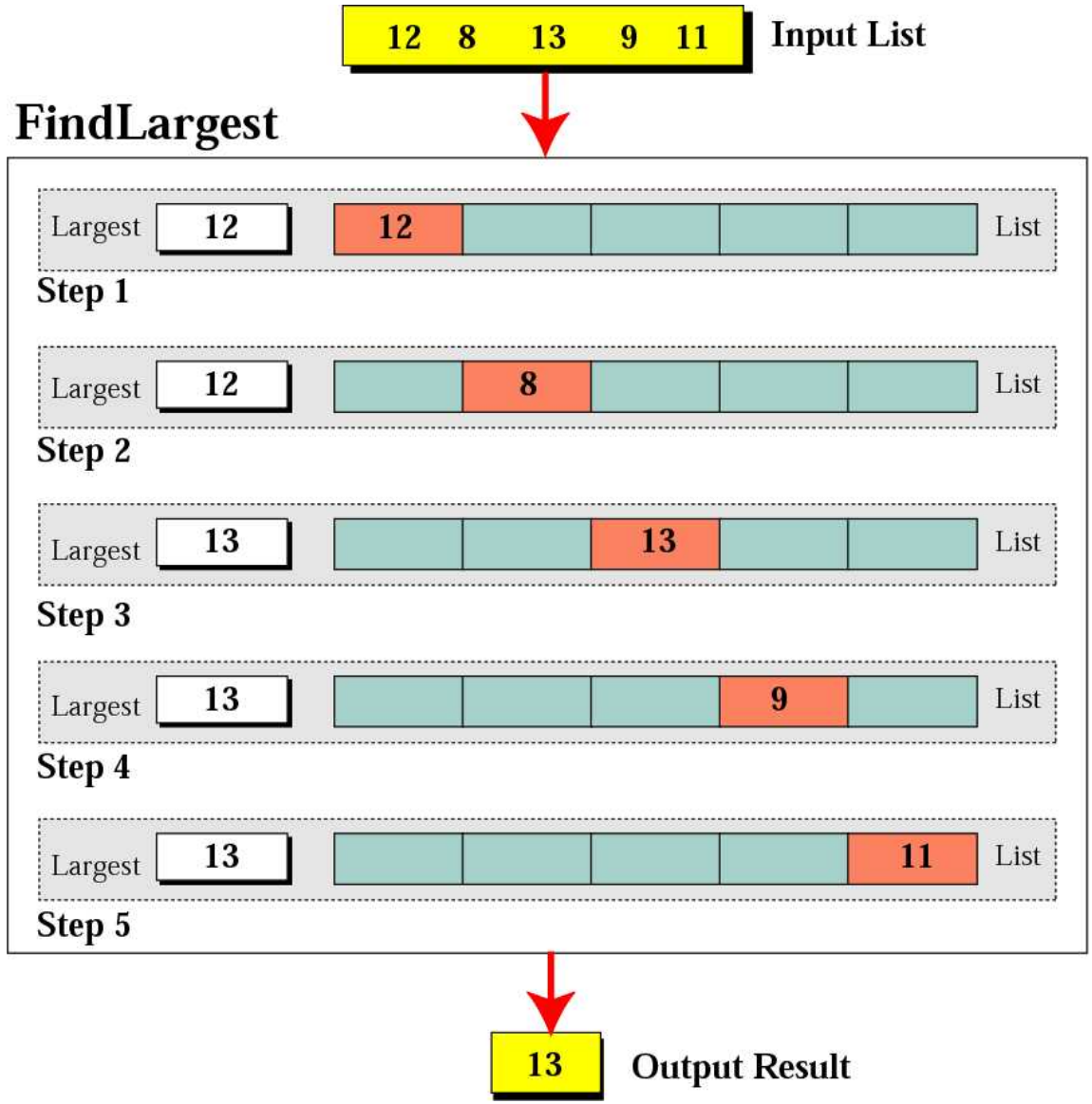
Write an algorithm to find the largest of a set of numbers. You do not know the number of numbers.

Solution

See Algorithm 8.4 on the next slide.

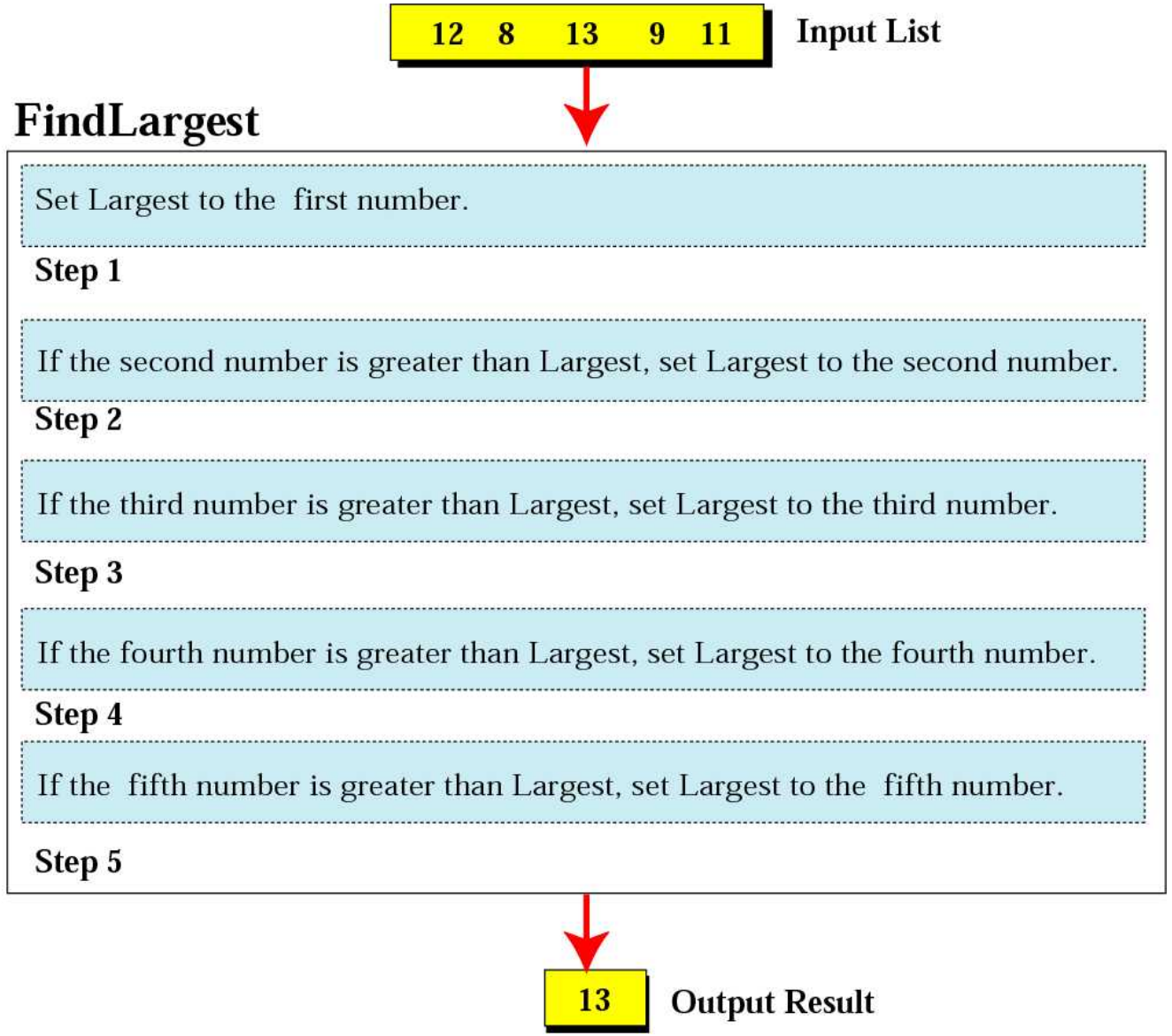
Finding the largest integer among five

Figure 8-2



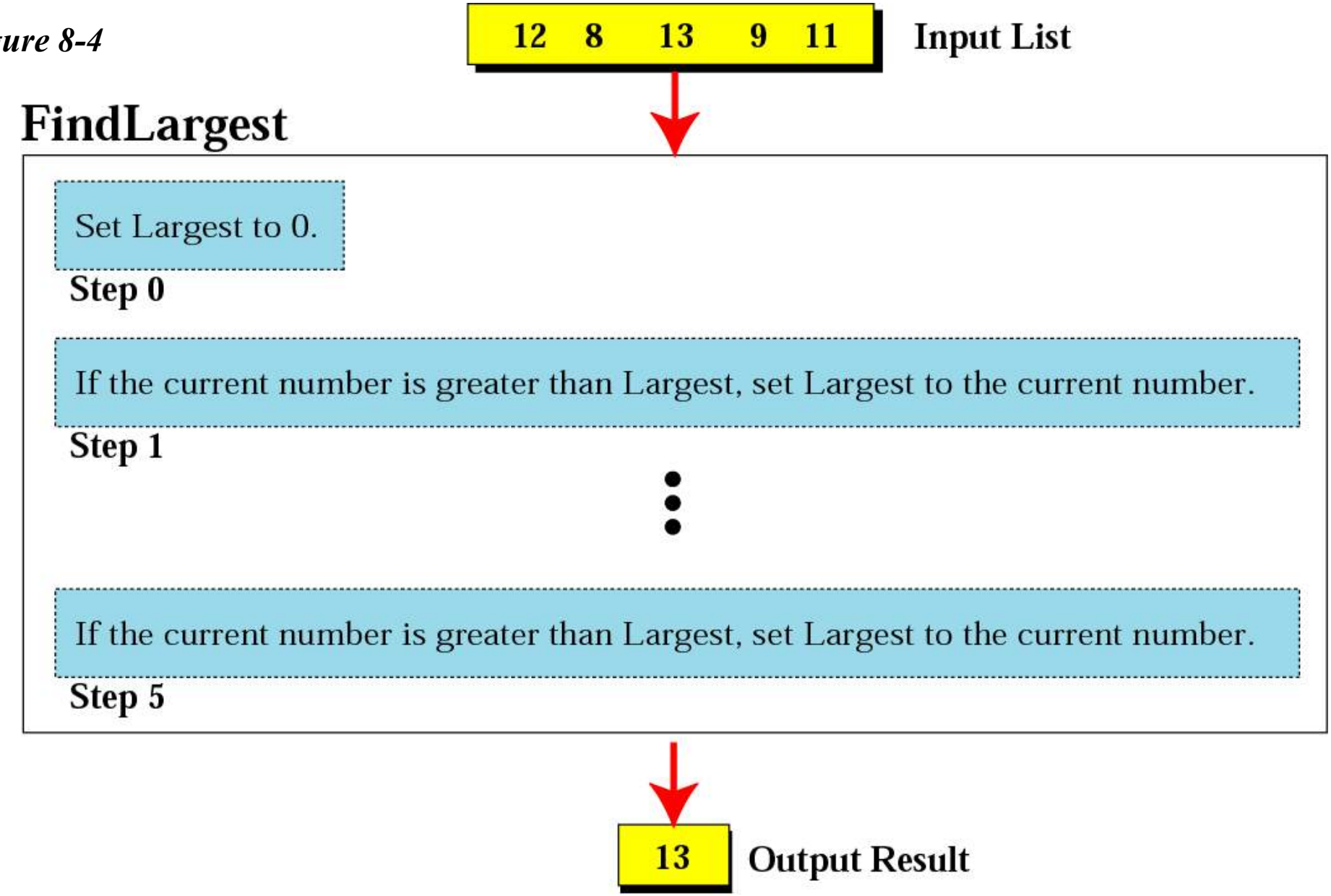
Defining actions in FindLargest algorithm

Figure 8-3



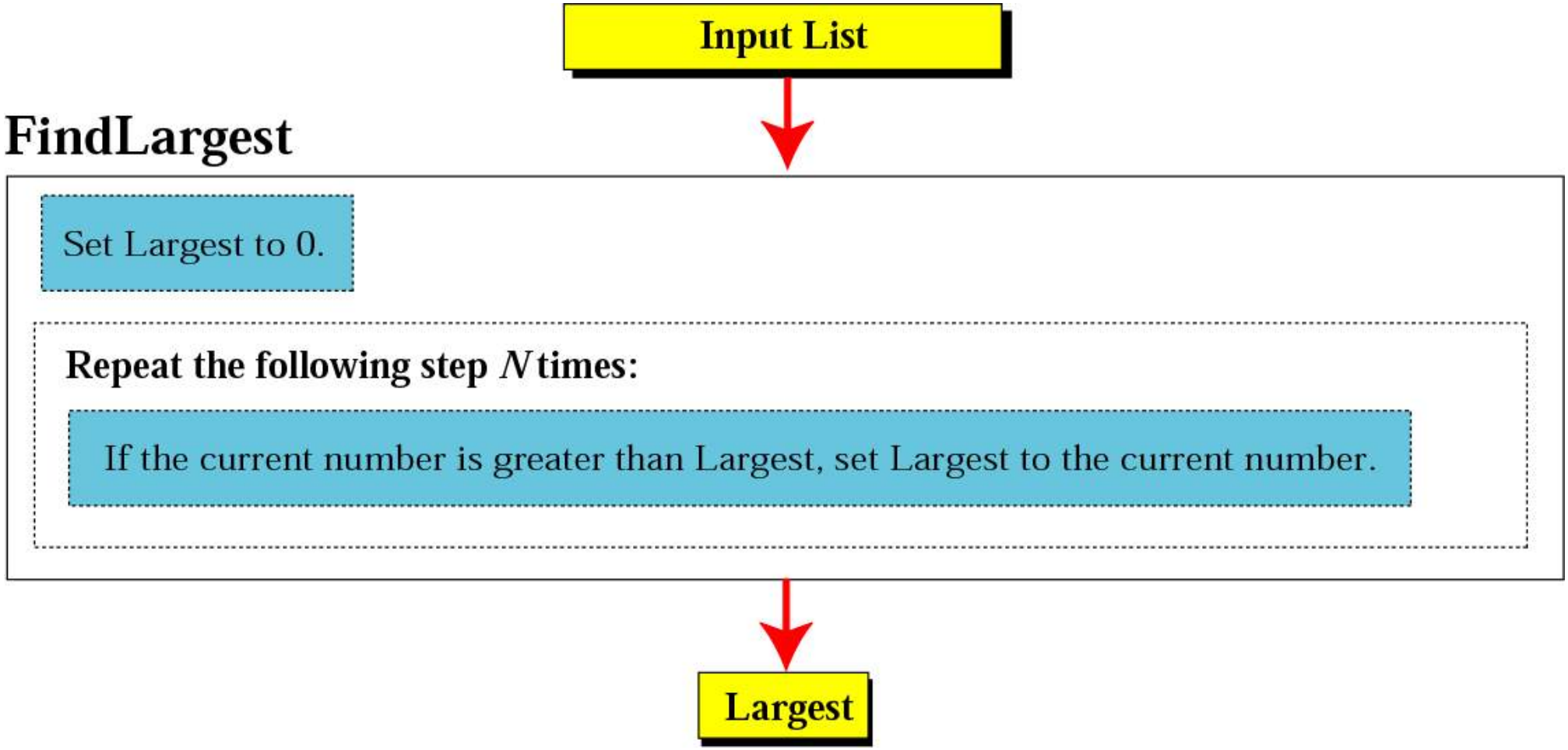
FindLargest refined

Figure 8-4



Generalization of FindLargest

Figure 8-5



Algorithm 8.4: Find largest

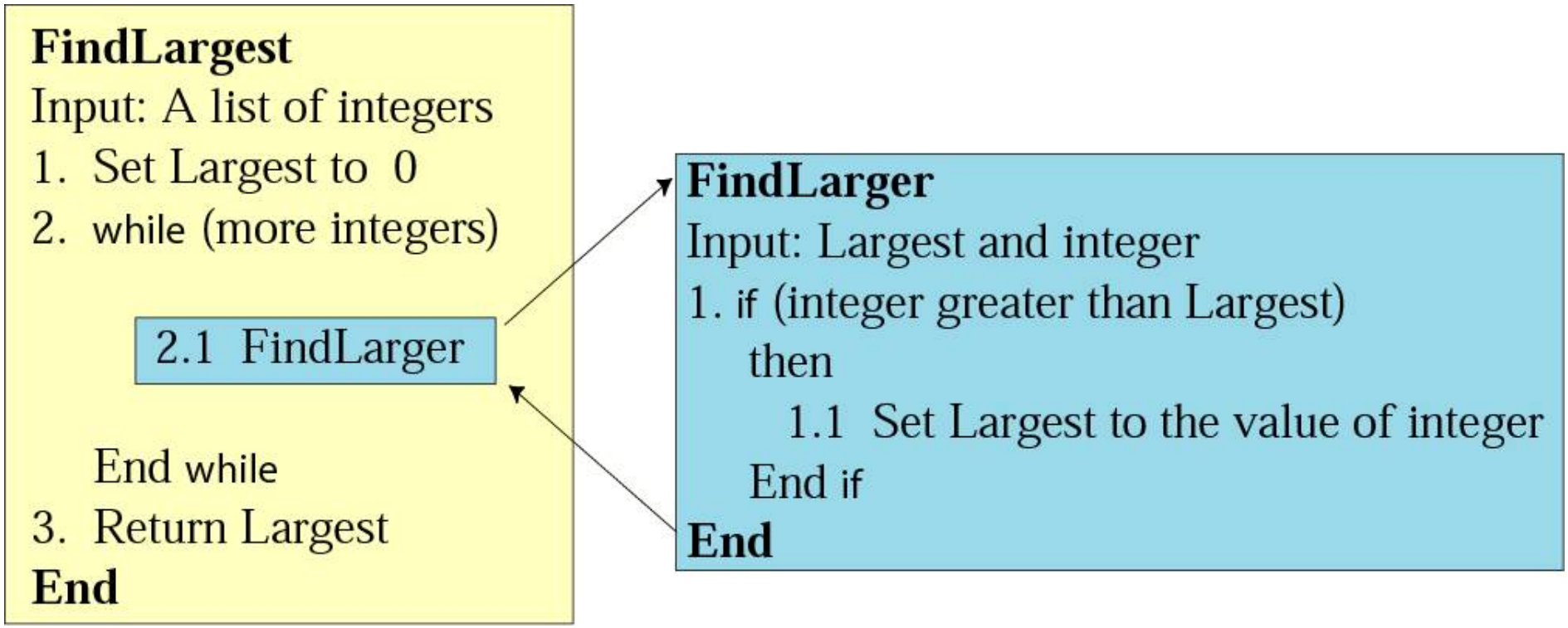
FindLargest
Input: A list of positive integers
1. *Set Largest to 0*
2. *while (more integers)*
   2.1 if (the integer is greater than Largest)
   then
   2.1.1 Set largest to the value of the integer
   End if
   End while
3. *Return Largest*
   End

8.5 SUBALGORITHMS

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

Concept of a subalgorithm

Figure 8-9



Algorithm 8.6: Find largest

FindLargest
Input: A list of positive integers
1. *Set Largest to 0*
2. *while (more integers)*
   2.1 FindLarger
   End while
3. *Return Largest*
   End

Subalgorithm: Find larger

```
FindLarger
Input: Largest and current integer
1.  if (the integer is greater than Largest)
    then
        1.1 Set Largest to the value of the integer
    End if
    End
```

Summation

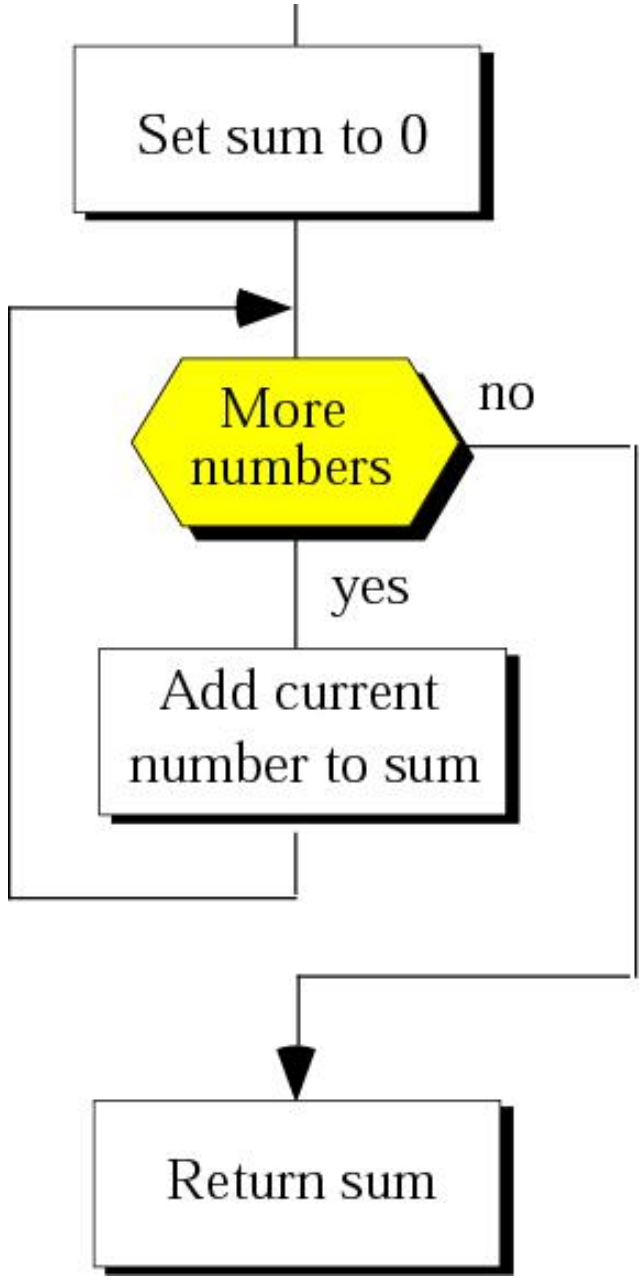
Product (multiplication)

BASIC ARITHMETIC OPERATIONS

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

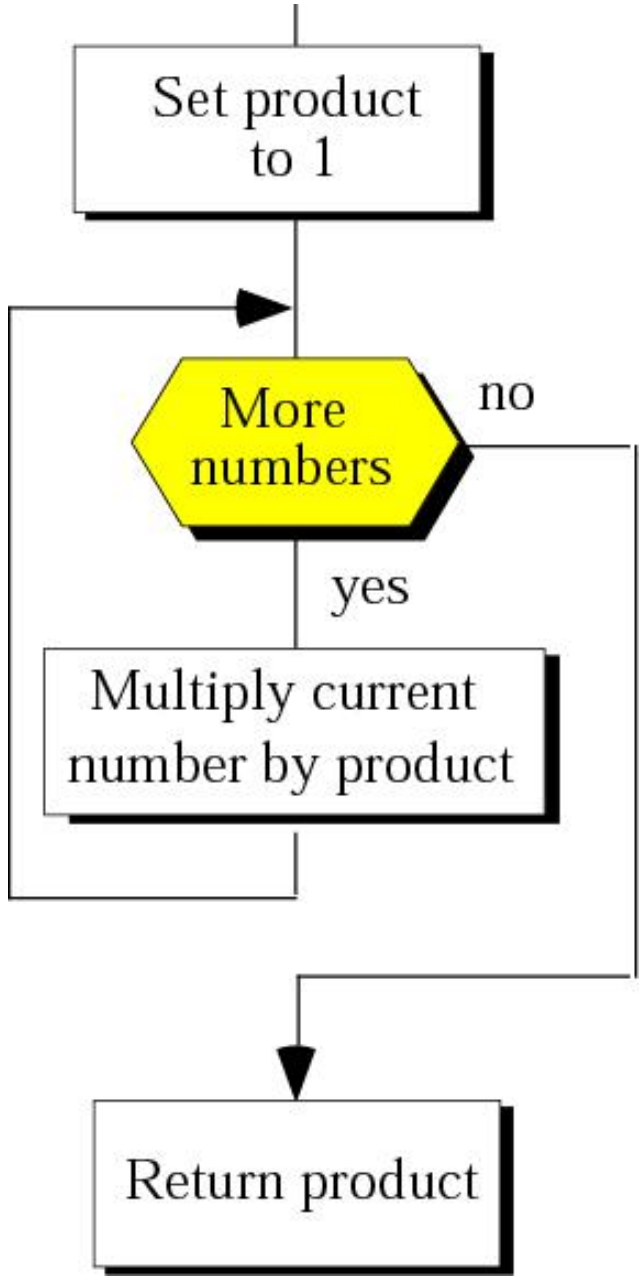
Figure 8-10

Summation



Product

Figure 8-11



RECURSION

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

Review: Factorial

- n에 대한 팩토리얼 함수는 1부터 n까지 모든 자연수를 곱하는 연산

$n! = n \times (n-1)!$

$(n-1)! = (n-1) \times (n-2)!$

$(n-2)! = (n-2) \times (n-3)!$

…

$2! = 2 \times 1!$

$1! = 1$ (베이스케이스)

그림 2-43 n! 연산

Iterative/Recursive Factorial

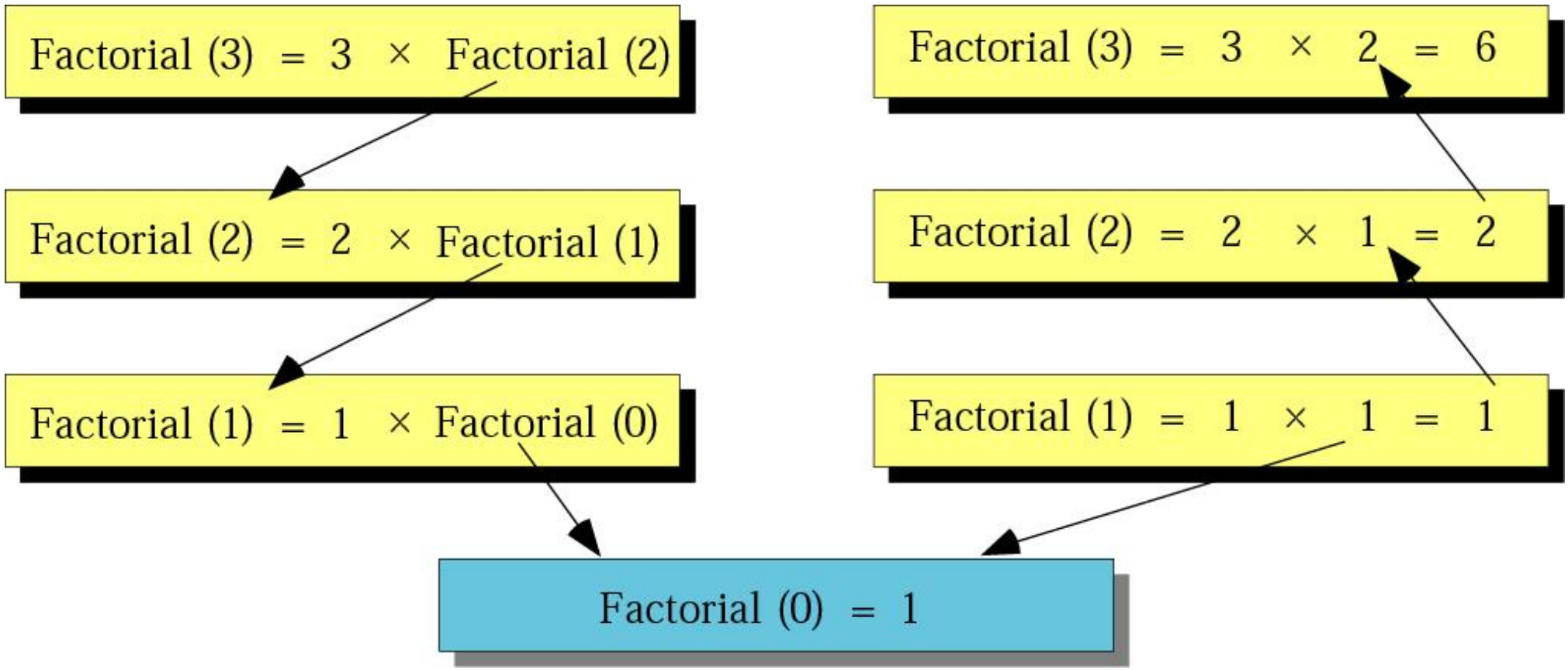
Figure 8-22

$$\text{Factorial } (n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \ \ldots \ \times \ 3 \times 2 \times 1 & \text{if } n > 0 \end{cases}$$

$$\text{Factorial } (n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times \text{Factorial } (n-1) & \text{if } n > 0 \end{cases}$$

Tracing recursive solution to factorial problem

Figure 8-24



Algorithm 8.7: Iterative factorial

Factorial
Input: A positive integer num
1. *Set FactN to 1*
2. *Set i to 1*
3. *while (i is less than or equal to num)*
   3.1 Set FactN to FactN x I
   3.2 Increment i
   End while
4. *Return FactN*

End

Algorithm 8.8: Recursive factorial

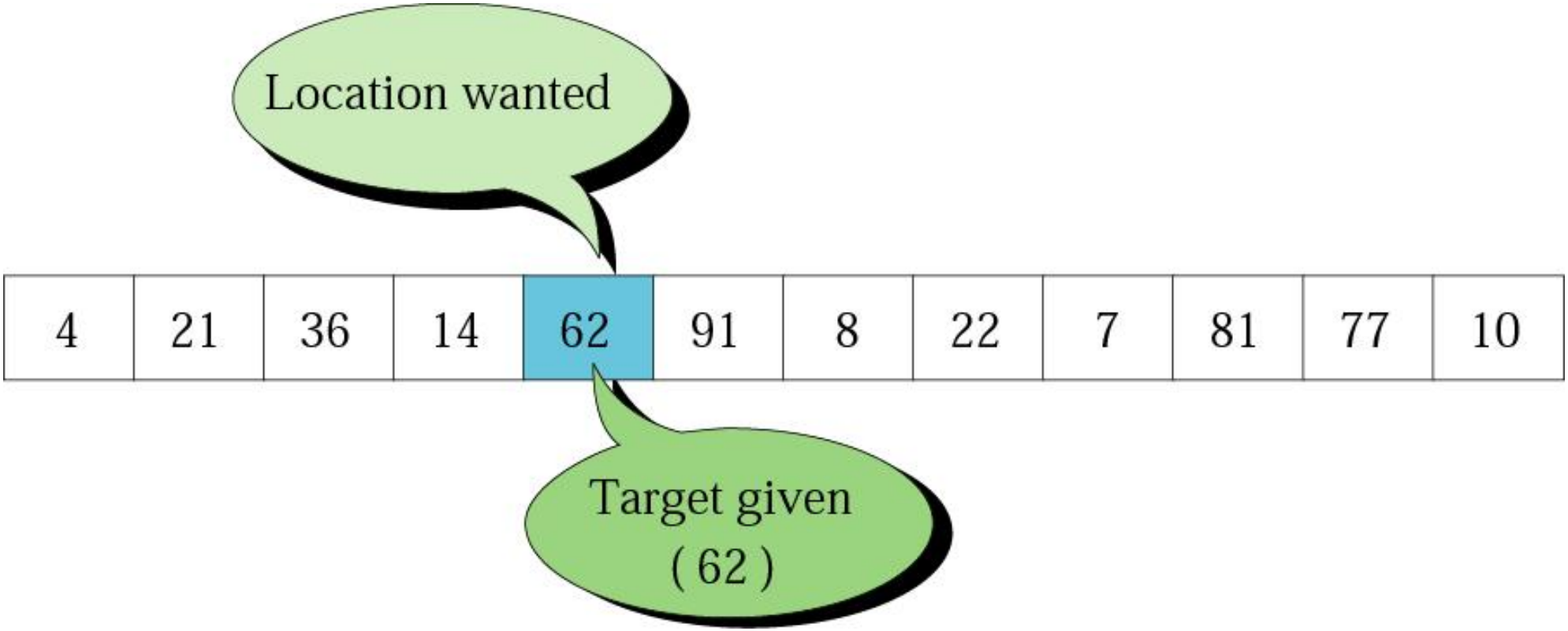
```
Factorial
Input: A positive integer num
1.  if (num is equal to 0)
    then
       1.1  return 1
    else
    1.2  return num x Factorial (num – 1)
    End if
    End
```

SEARCH: 검색

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

Search concept

Figure 8-19



순차 검색(sequential search)

리스트의 다음과 같은 키 값들이 존재한다고 하자.

{4, 21, 36, 14, 62, 91, 8, 22, 7, 81, 77, 10}

이 리스트에서 키 값이 62인 데이터 요소를 찾기 위해서 이 리스트의 첫 번째 키 값부터 차례대로 비교하며 찾아 나가는 것을 순차 검색이라고 한다.

리스트에 N개의 데이터 요소가 존재하면 평균 비교 회수는 N/2이 된다.

Figure 8-20: Part I

Example of a sequential search

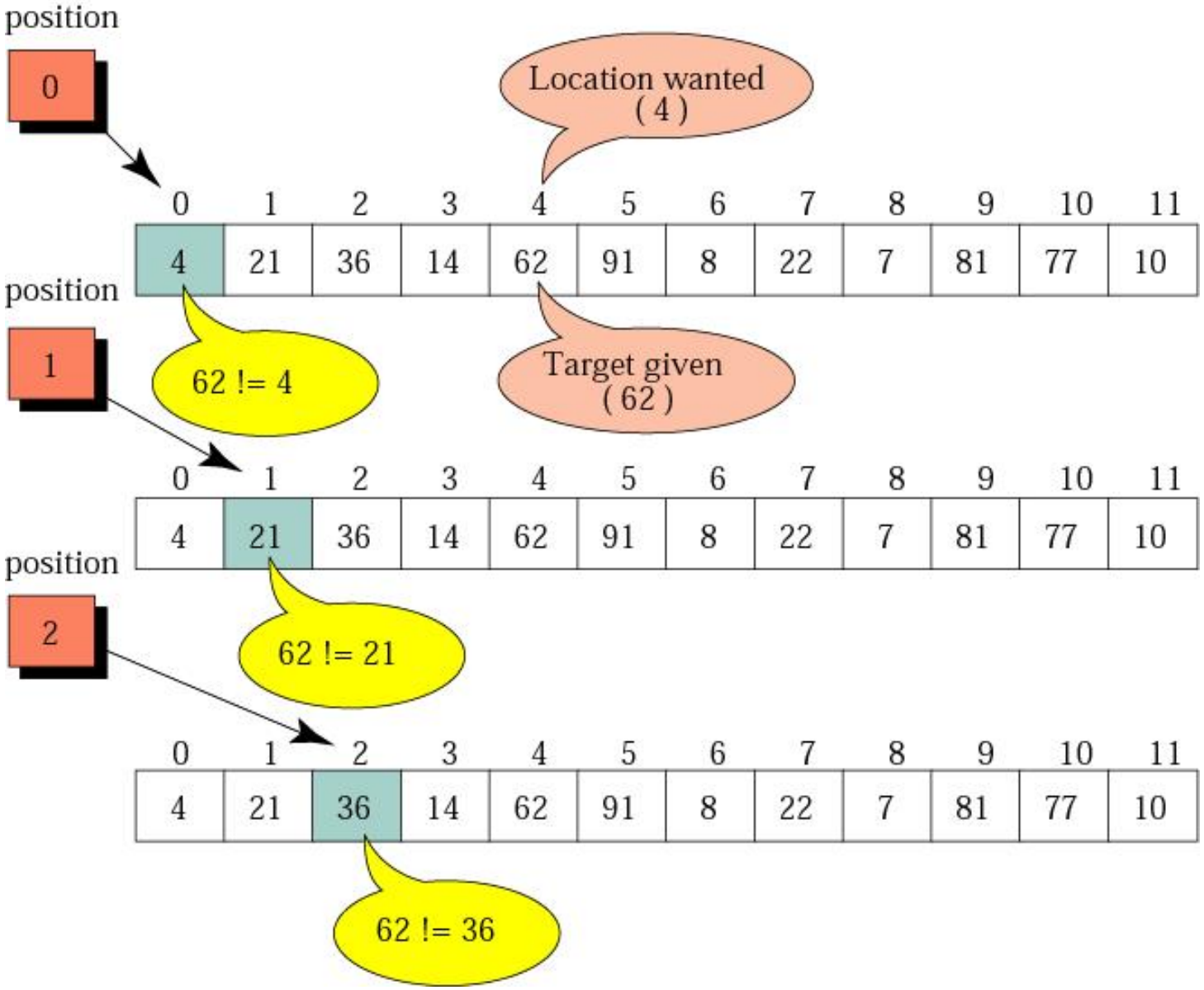
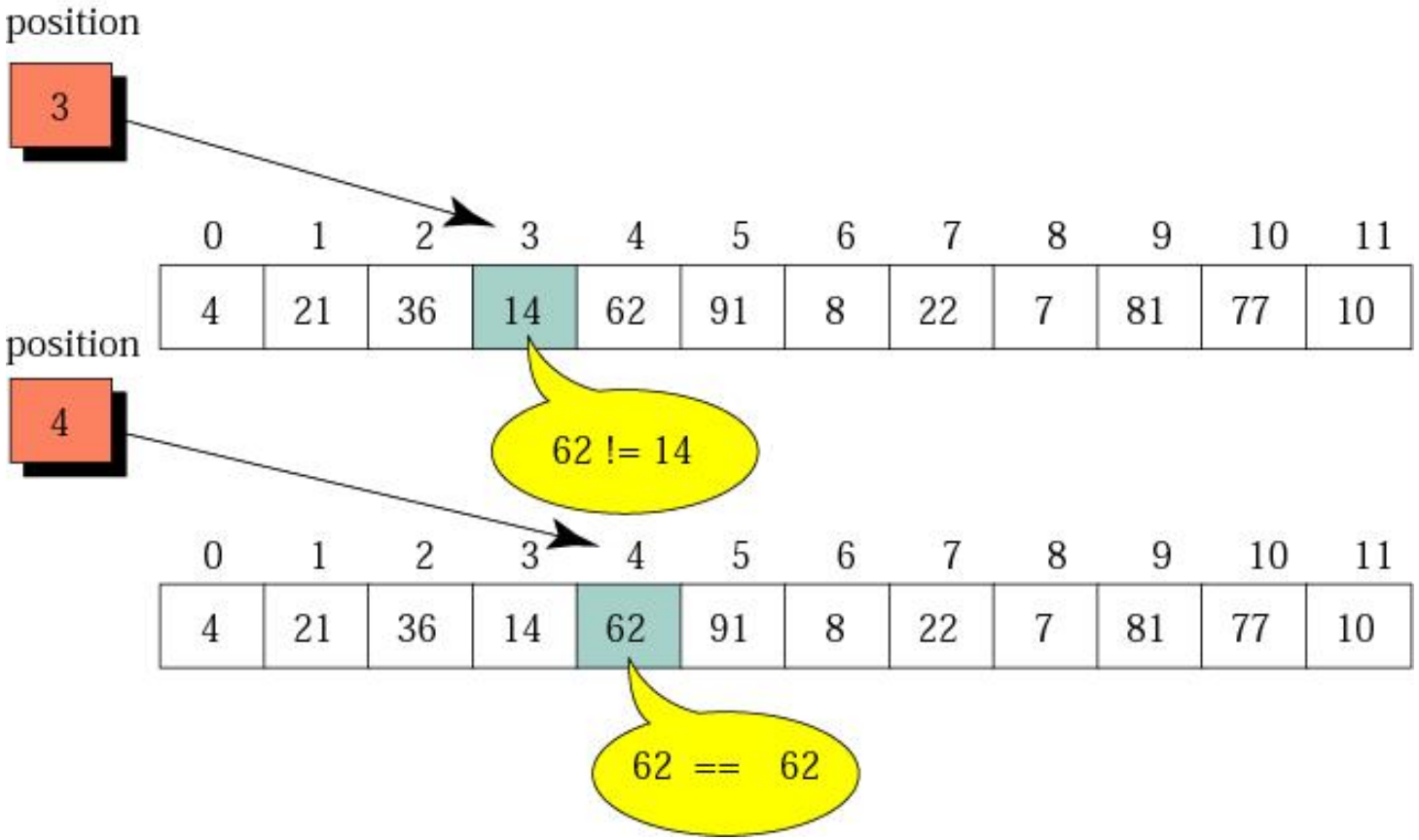


Figure 8-20: Part II Example of a sequential search



이진 탐색(binary search)

만약 리스트의 키 값이 정렬되어 있다면 순차 검색 보다 빠른 시간 안에 검색을 할 수 있다.

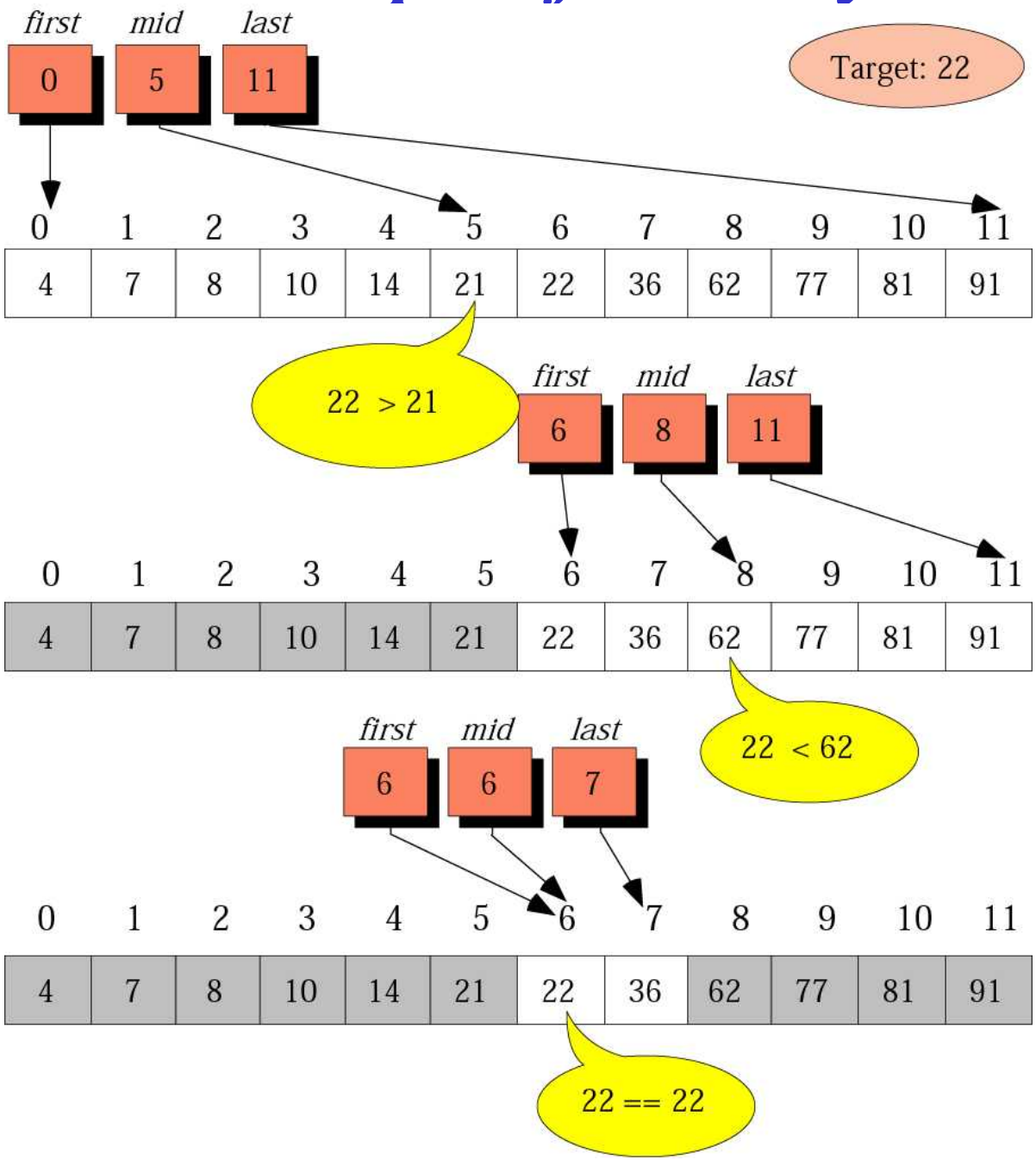
*{4, 7, 8, 10, 14, **21**, 22, 36, 62, 77, 81, 91}*

먼저 찾고자 하는 키 값을 리스트의 중간에 위치한 값인 21과 비교한다. 만약 찾고자 하는 키 값이 21 보다 작으면 이 값은 21보다 왼쪽에 위치하고 있으며, 21보다 크다면 21의 오른쪽에 위치하고 있다. 따라서 다음 단계에서 21보다 왼쪽에 있는 값들, 혹은 오른쪽에 있는 값들을 갖고 같은 절차를 반복한다.

최대 비교 회수는 리스트의 수가 N이라면 $\log_2 N$ 이다.

Example of a binary search

Figure 8-21



Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교

[알고리즘] 이진 탐색

```
while (리스트 구간의 크기 > 0)
   구간의 중간값을 구한다.
   if(구간의 중간값 = 키값)
         탐색 종료
   else if(구간의 중간값 > 키값)
         오른쪽 구간 선택
   else
         왼쪽 구간 선택
```

이러한 이진 검색을 하기 위해서는 먼저 리스트가 순서대로 정렬되어 있어야 한다. 따라서 탐색 전에 순서대로 정렬하기 위한 시간이 요구된다.

```c
/* Iterative Binary Search, https://www.tutorialspoint.com/binary-search-recursive-and-iterative-in-c-program */
#include <stdio.h>
int iterativeBinarySearch(int array[], int start, int end, int
element){
   while (start <= end){
      int middle = start + (end- start )/2;
      if (array[middle] == element)   return middle;
      if (array[middle] < element)    start = middle + 1;
      else                            end = middle - 1;
   }
   return -1;
}
int main(void){
   int array[] = {1, 4, 7, 9, 16, 56, 70};
   int n = 7;
   int element = 16;
   int found_index = iterativeBinarySearch(array, 0, n-1, element);
   if(found_index == -1 ) printf("Element not found in the array ");
   else printf("Element found at index : %d",found_index);
   return 0;
}
```

```c
/* Recursive Binary Search, https://www.tutorialspoint.com/binary-search-recursive-and-iterative-in-c-program */
#include <stdio.h>
int recursiveBinSearch(int array[], int start, int end, int element){
   if (end >= start){
      int middle = start + (end - start )/2;
      if (array[middle] == element) return middle;
      if (array[middle] > element)
         return recursiveBinSearch(array, start, middle-1, element);
      return recursiveBinSearch(array, middle+1, end, element);
   }
   return -1;
}
int main(void){
   int array[] = {1, 4, 7, 9, 16, 56, 70};
   int n = 7;
   int element = 9;
   int found_index = recursiveBinSearch(array, 0, n-1, element);
   if(found_index == -1 ) printf("not found ");
   else printf("Found at index : %d",found_index);
   return 0;
}
```

Next class: Algorithm complexity analysis

END OF LECTURE 2
BASIC ALGORITHMS

Slide credits: Kun-Mao Chao, Intro. Computer Science, Nat'l Taiwan Univ.; J. Lillis, UIC's CS 201, George Bebis;
문병로, 쉽게 배우는 알고리즘, 한빛아카데미㈜; C로 쉽게 풀어쓴 자료구조, 생능출판사; 하상호, 순천향대학교; 김종익, 전북대학교