Computational Thinking

2017-2학기

5장-알고리즘 사고

박종혁 교수 (컴퓨터공학과)

jhpark1@seoultech.ac.kr

http://www.parkjonghyuk.net

서울과학기술대학교 SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

학습개요

- 5장 알고리즘적 사고
  1. 알고리즘
  2. 소프트웨어와 프로그래밍 언어
  3. 동작

- 조별발표
  - 조별 10분 발표

학습목표

- 소프트웨어와 프로그램의 실행에 대한 개념을 파악한다.
- 알고리즘은 항상 선택이라는 것을 수반하며 이 선택이라는 것은 논리적 조건문의 모습을 띄고 있다는 것을 파악한다.
- 알고리즘에서 명령어가 반복되는 것이 흔한 일이라는 것을 이해한다.
- 알고리즘이 어떻게 모듈화 되는 지를 파악한다.
- 네이밍, 선택, 반복과 같은 명령문을 표현할 수 있는 플로우차트 요소들의 형태와 기능을 파악한다.
- 컴퓨팅 계산적 상태, 이벤트, 액션의 개념을 파악한다.
- 10개 또는 그 이하의 상태 개수를 가지는 순차 알고리즘을 모델링한다.

❖ 알고리즘 (algorithm)

- 목적을 달성하거나 문제해결을 하도록 하는 독립된 동작들의 순서
- 페르시아의 수학자 - 알콰리즈미의 이름에서 따온 것
- 동영상
  https://www.youtube.com/watch?v=S8PA_VCJiow
  https://www.youtube.com/watch?v=PYB32Qx-etk&t=6s

- 보이지 않는 혁명 컴퓨터 알고리즘 매매
  https://www.youtube.com/watch?v=gUBtVwnm908

❖ **알고리즘 (컴퓨터분야의 정의)**

- 어떤 입력을 원하는 출력으로 변환시키는 컴퓨터 동작의 순서
- 예) 초코렛 칩 쿠키 요리법
  - 입력: 요리 재료, 알고리즘: 요리법
  - 출력: 초코렛 칩 쿠키

[요리 재료]

- 1 컵 분량의 버터 녹인 것
- 컵 분량의 갈색 설탕
- 계란 2개
- 3 컵 분량의 밀가루
- 1 티스푼 분량의 베이킹 파우더
- 1 티스푼 분량의 베이킹 소다
- 2 2 컵 분량의 초콜렛 칩

[요리법]

1. 오븐을 화씨 375도로 미리 데운다.
2. 양피지 종이로 만든 쿠키 용지를 준비한다.
3. 한 용기에 버터, 갈색 설탕, 계란을 넣고 함께 젓는다.
4. 다른 용기에 밀가루, 베이킹 파우더와 베이킹, 소다를 넣고 섞는다. 점차적으로 설탕 혼합물을 섞는다.
5. 용기에 초콜렛 칩을 넣는다.
6. 한 스푼 용량의 쿠키 반죽으로 쿠키 용지를 채운다.
7. 9분간 오븐에 굽는다.
8. 쿠키 용지를 오븐에서 꺼내어 5분간 상온에서 식힌다.

❖ 알고리즘의 4대 요소

- 알고리즘의 모든 동작들은 의미(semantic)가 있어야 한다.
- 알고리즘의 모든 동작들은 모호하지 않아야 한다. (unambiguous)
  - 어떤 행위도 그 해석이 상충되어서는 안 된다.
- 알고리즘의 동작들은 수행되는 순서가 확실하게 정의되어야 한다.
- 알고리즘은 반드시 유한한 숫자의 동작들을 실행한 후 종료되어야 한다. (halt)

❖ **컴퓨터 소프트웨어는 프로그래밍 언어 (programming language)로 작성된다.**

- 프로그래밍 언어 – 알고리즘을 계산적으로 정확하고 간결하게 표현할 수 있도록 설계된 언어

❖ **컴퓨터는 프로그래밍 언어로 작성된 단계들을 그대로 수행할 뿐이다.**

❖ **프로그래밍 언어의 종류**

- 명령형 언어 (imperative lang.)
  - 컴퓨터 명령어들을 순서적으로 실행함
  - 컴퓨터 계산을 표현
- 함수형 언어 (functional lang.)

  입력값과 출력값의 관계로 프로그램을 표현
- 선언형 언어 (declarative lang.)
  - 입력값에 연관있는 사실을 사용하여 출력값을 생성하도록 표현함
  - 논리 프로그래밍, 객체지향 프로그래밍, 병렬 프로그래밍

❖ **모든 실행 가능한 (possible) 알고리즘은 가능한 동작들의 순서로 표현된다.**

- 컴퓨터 (computable) 동작 : 컴퓨터가 실행할 수 있는 동작
- 계산가능한 함수 (computable functions) : 컴퓨터가 실행할 수 있는 함수

연비 효율적인 여행 알고리즘

1. 남서쪽으로 50 야드 뛰어간다.
2. 2440 마일을 점프한다.
3. 로스엔젤리스에 착지한다.

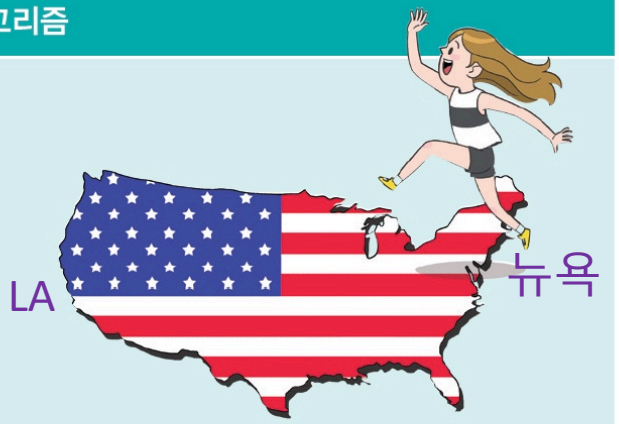


그림 5.1 연비 효율적 여행을 가능하게 하는 비현실적인 알고리즘

❖ 값 이름 연동 = 식별자(identifier)

이름(Name)에 값을 연결(bind)시키는 것

- Identifier ← expression

네임 바인딩
식별자 ← 표현식

그림 5.2 네임 바인딩을 하는 방법. 화살표 우측의 값은 화살표 좌측의 식별자에 연동된다.

```
X ← 3
Y ← 4
Z ← 3+ 4
```

```
X ← 3
Y ← 4
Z ← X + Y
```

5.3.1 값/이름 연동 (적절한 이름짓기)

❖ 이름은 프로그래밍 요소들의 본질을 잘 반영하도록 지어야 한다.

두 휘발유 가격 알고리즘	
1. DollarsPerGallon ← 3.75	1. Cents ← 3.75
2. TankCapacity ← 10	2. Size ← 10
3. DollarsToFill ← TankCapacity × DollarsPerGallon	3. Money ← Size × Cents

그림 5.3 데이터의 이름만 다른 두 알고리즘

❖ 우측 알고리즘 보다 좌측 알고리즘이 프로그램의 내용을 더 이해하기 쉽게 작성되어 있다.

❖ 프로그램이 실행됨에 따라 값 이름 연동의 상태가 변하게 된다.

```
X ← 3
X ← 4
X ← X+ X
```

식별자 X의 값이 프로그램이 실행됨에 따라 3, 4, 8로 계속 바뀐다

❖ 계산 상태 (computational state)

- 실행 중 어느 한 순간의 활성화된 값 이름 연동의 집합

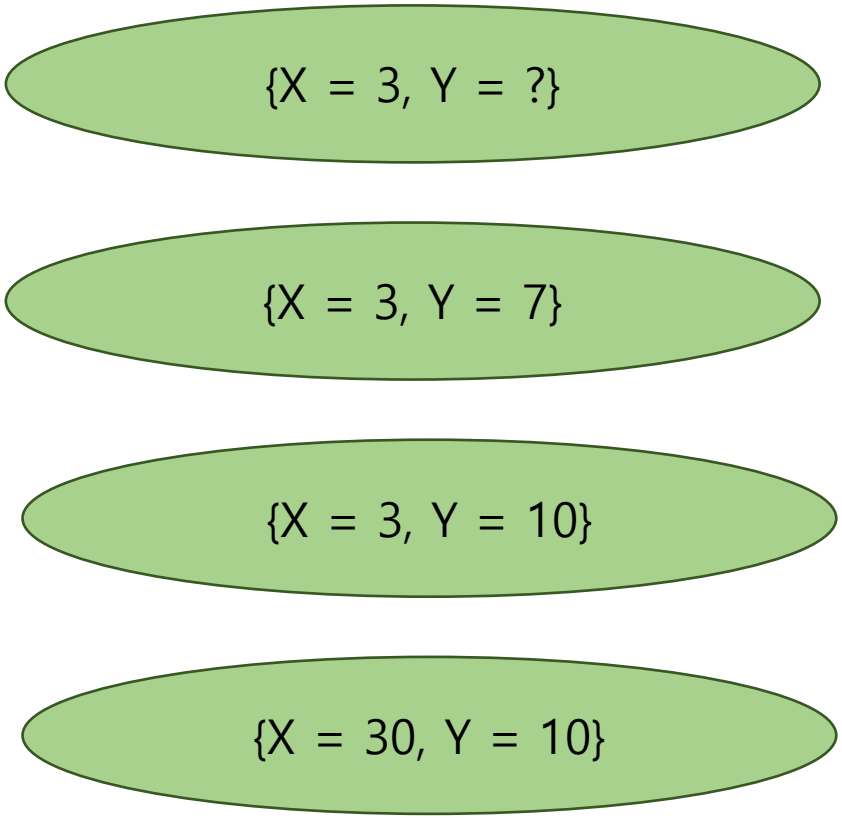
5.3.1 값/이름 연동 (계산 상태)

1. X ← 3

2. Y ← X + 4

3. Y ← X + Y

4. X ← X * Y



프로그램

계산상태

❖ 계산상태 예제 – 섭씨 온도에서 화씨 온도로 바꾸는 프로그램

섭씨에서 화씨로 바꾸는 알고리즘

1. Celsius ← 33.5
2. Fahrenheit ← Celsius * 9
3. Fahrenheit ← Fahrenheit / 5
4. Fahrenheit ← Fahrenheit + 32

그림 5.5 온도 변환 알고리즘은 네임 바인딩에서 어떻게 변하는 것이 허용되는 지를 보여준다.

섭씨 온도에서 화씨 온도로 바꾸기

1. Celsius ← 33.5
   현재 상태는 {Celsius = 33.5}
2. Fahrenheit ← Celsius * 9
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 301.5}
3. Fahrenheit ← Fahrenheit / 5
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 60.3}
4. Fahrenheit ← Fahrenheit + 32
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 92.3}

그림 5.6 알고리즘의 실행에 따른 컴퓨팅 상태의 변화

❖ **제어흐름 (control flow)**
- 컴퓨터 프로그램의 개별적인 동작들이 정해진 순서대로 실행되는 것

❖ **동작들은 프로그래머에 의해 작성된 순서에 따라 실행됨**

❖ **동작의 순서를 유연하게 구성하여 컴퓨터가 다양한 입력에 반응할 수 있도록 해야 함**
- 이러한 유연성은 제어 흐름 명령문에 의해 지원됨.
- 제어 흐름 명령문은 어떤 액션이 수행되어야 할 지를 결정하는 데 관해 컴퓨터가 선택하도록 함
  - 단방향 선택 (one-way selection)
  - 두방향 (two-way) 선택
  - 다방향 (multiway) 선택

5.3.2 선택 (단방향 선택)

❖ 단방향 선택: 컴퓨터가 동작을 실행하거나 동작을 건너뛰도록 해준다

단방향 선택

```
if CONDITION then
ACTIONS
endif                    // 대문자 부분- 알고리즘의 실제 명령어로 대치될 부분
```

그림 5.8 단방향 선택 문법 패턴

배송료를 계산하는 프로그램

```
1.  orderAmount ← shoppingCartTotal()
2.  shippingCost ← 0
3.  if orderAmount < 40 then
4.      shippingCost ← 10
5.  endif
```

그림 5.9 주문량에 따른 배송료를 계산하는 프로그램

5.3.2 선택 (단방향 선택)

❖ 플로우차트 (Flowchart): 활동 다이어그램(activity diagram)
- 알고리즘을 통해 시각적으로 제어 흐름을 강조해주는 플로우차트 사용
- 복잡한 알고리즘의 내부에서 발생하는 동작들의 순서를 편하게 볼수 있도록 도와줌

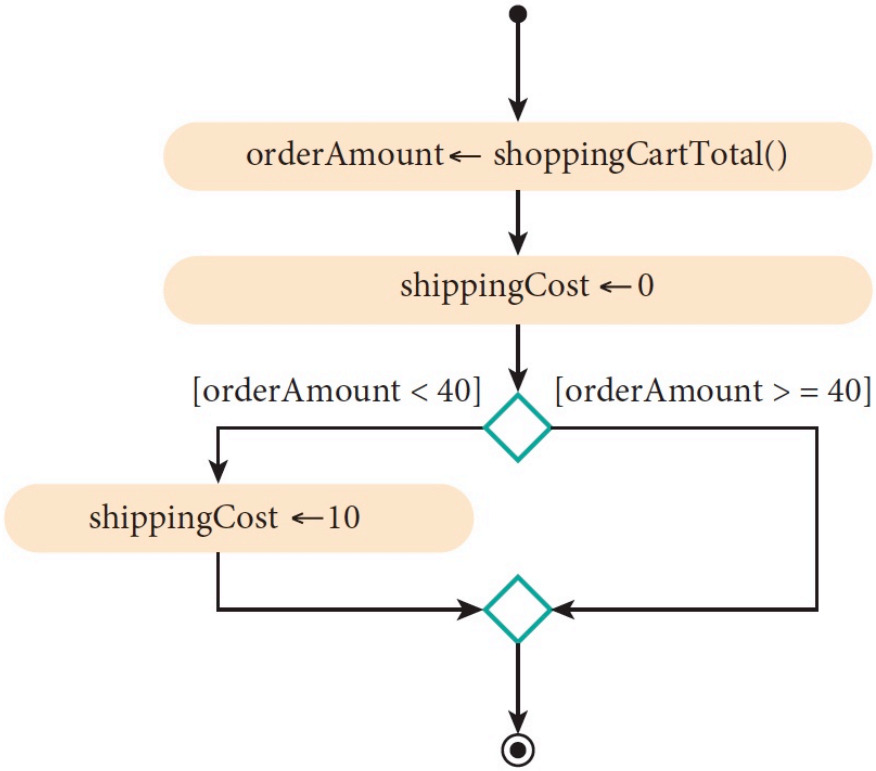


그림 5.7 배송비 계산에 대한 플로우차트

5.3.2 선택 (두방향 선택)

❖ 두방향 선택: 컴퓨터가 정확히 두가지 동작 중 하나를 선택하도록 하는 명령문

배송료 계산 프로그램

```
1.  orderAmount ← shoppingCartTotal()
2.  if orderAmount < 40 then
3.     shippingCost ← 10
4.  else
5.     shippingCost ← 0
6.  endif
```

그림 5.12 주문량에 대한 배송료를 계산하는 프로그램

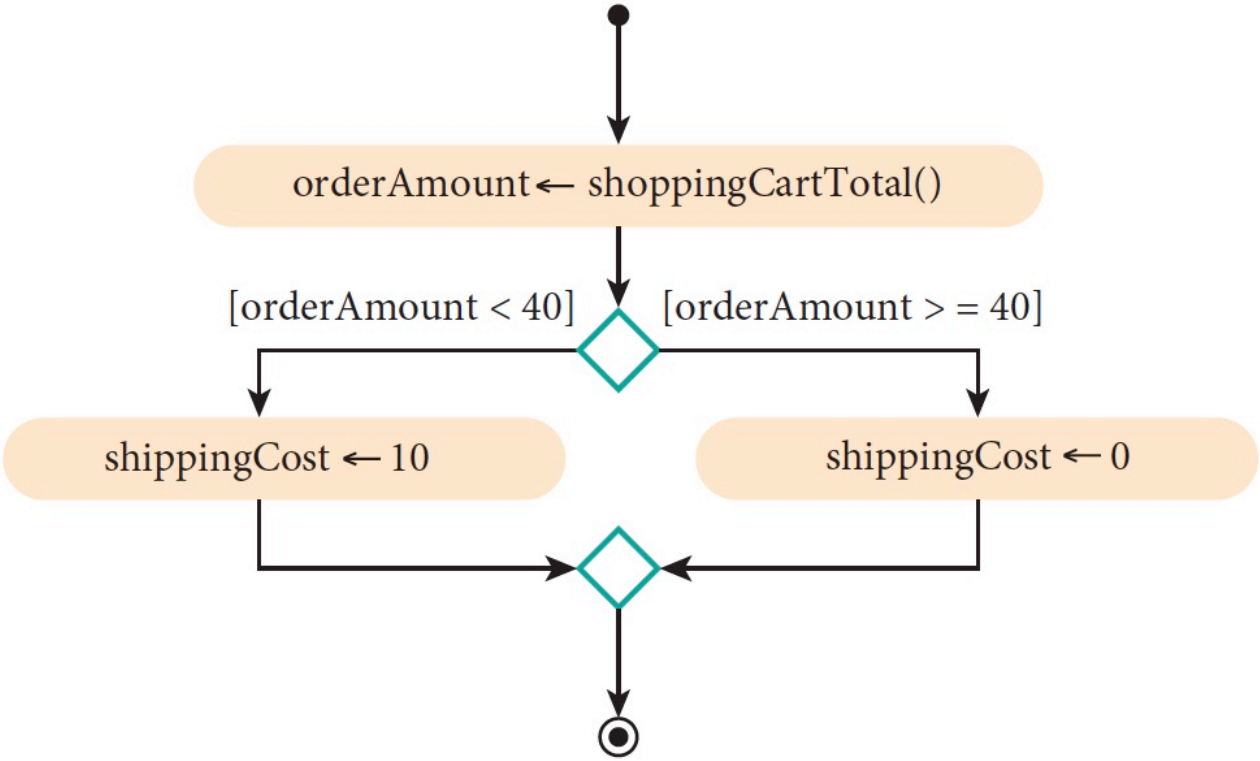


그림 5.10 배송료를 계산하는 것을 나타내는 개선된 플로우차트

5.3.2 선택 (다방향 선택)

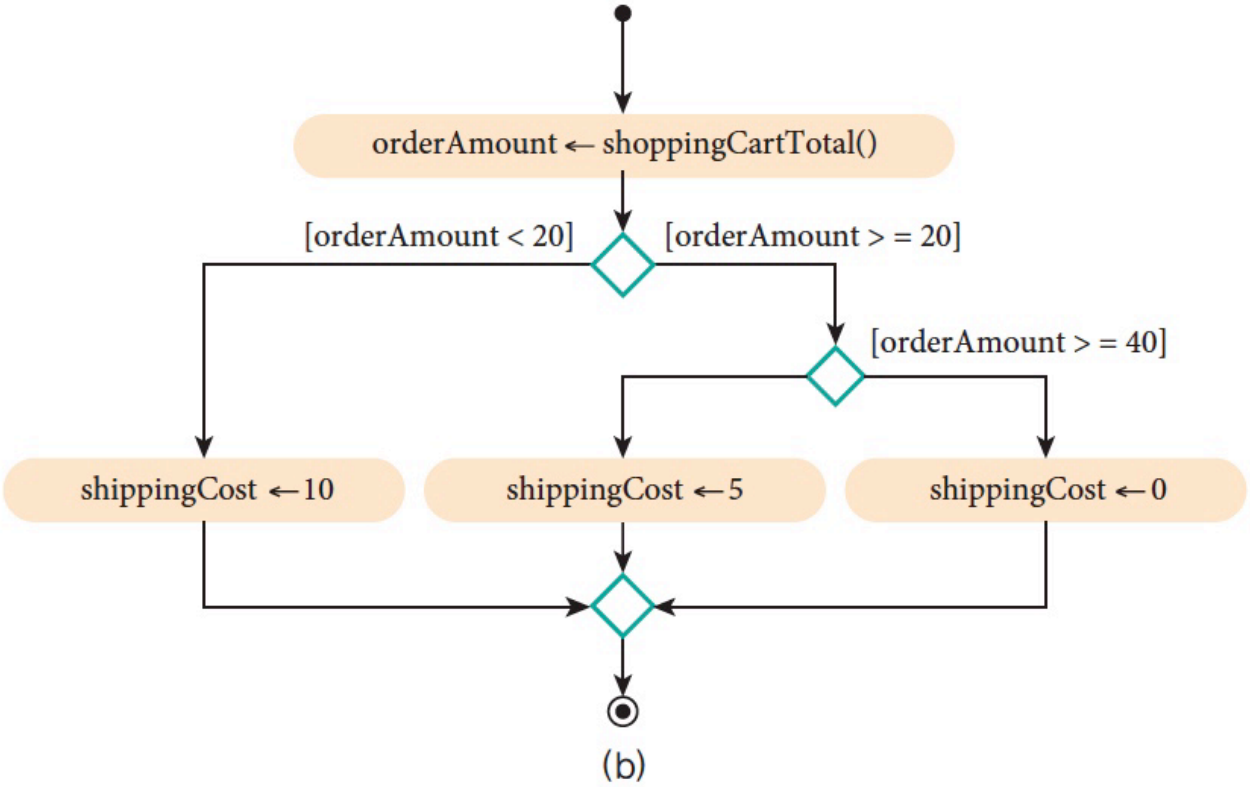
❖ 다방향 선택: 컴퓨터가 여러 대안들 중 하나만을 선택하도록 해주는 명령문

배송료 계산 프로그램

```
1. orderAmount ← shoppingCartTotal ()
2. if orderAmount ≥ 0 and orderAmount < 20 then
3.    shippingCost ← 10
4. elseif orderAmount ≥ 20 and orderAmount < 40 then
5.    shippingCost ← 5
6. else
7.    shippingCost ← 0
8. endif
```

그림 5.14 주어진 주문량에 대해 0달러, 5달러, 10달러의 배송료를 책정하는 프로그램

5.3.2 선택 (다방향 선택)



(b)

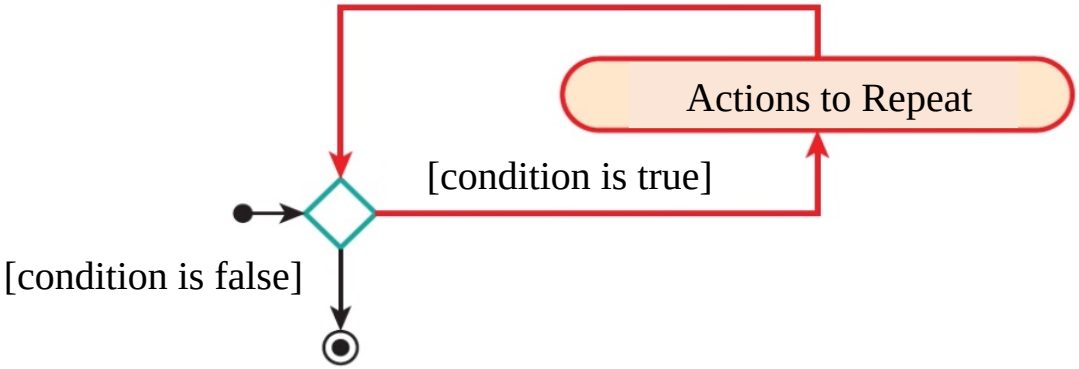
그림 5.13 다방향 선택을 모델링하는 플로우차트; [redacted]
세 가지 대안 중에 하나를 선택하고 있다.

❖ 루프 (loop)

- 일련의 동작들을 반복적으로 실행하는 제어구조
- While 루프

  -특정 논리적 조건이 성립하는 상황에서 일련의 동작을 계속 반복시키는 루프

  - 처음 만나게 되면, 일련의 동작을 한번 이상 실행할지 결정해야 함



[while 루프에 대한 플로우차트 구조]

```
while CONDITON do
ACTION
endwhile
```

[while 루프의 문법 패턴]

5.3.3 반복

고기를 그릴에 굽는 프로그램

```
1. steakTemp ← 75
2. while steakTemp < 135 do
3.     steakTemp ← steakTemp + 13
4. endwhile
```

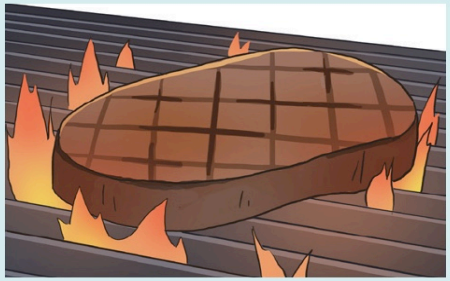


그림 5.19 고기를 그릴에 구울 때 고기의 온도를 모델링하는 알고리즘

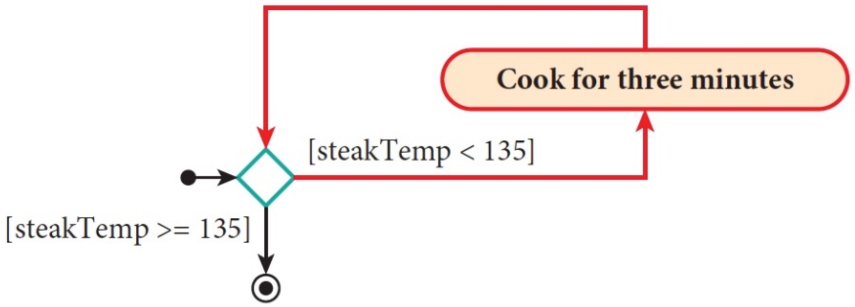


그림 5.17 고기를 그릴에 굽는 과정을 모델링 하는 while 루프를 사용하는 플로우차트

❖ 반복 셈하기 루프 (counting loop)

- 루프가 얼마나 반복되는지에 대한 숫자를 세는 변수를 가지는 루프

그림 5. 21 그릴에 고기를 굽는데 걸리는 시간을 계산하는 프로그램

❖ 잘 작성된 루프를 위한 3가지 요소

- 초기화 – 모든 변수는 루프에 진입하기 전 정확한 값을 가지고 있어야 함
  그림 5.21 - 1, 2번째 줄
- 조건문 – 루프를 언제까지 반복할지를 결정
  그림 5.21 - 3번째 줄
- 진전 – 동작들이 반복적으로 실행되어 상황이 진전되어 반복 로프가 종료되어야 함
  그림 5.21 - 4번째 줄

❖ 무한루프

- 논리에 오류가 있어 루프가 종료되지 않음
- 무한루프는 실제 소프트웨어의 많은 오류들의 원인이 됨

고기를 그릴에 굽는 시간을 계산하는 프로그램(잘못됨)

```
1. steakTemp ← 75
2. minOnGrill ← 0
3. while steakTemp ≠ 135 do
4.     steakTemp ← steakTemp + 13
5.     minOnGrill ← minOnGrill + 3
6. endwhile
```

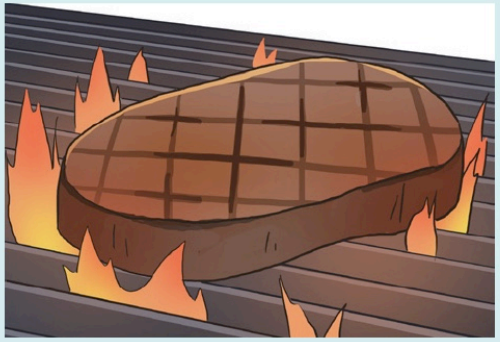


그림 5.22 고기를 그릴에 굽는데 얼마나 오래 시간이 걸리는 지를 계산하는 잘못된 프로그램

// 프로그램 실행시, 고기온도는 75, 88, ... 127, 140 .. 값을 가짐. 135도가 될 수 없음

❖ 모듈화 (Modulization)

- 컴퓨터로 계산이 가능한 어떤 과정들에 하나의 이름을 할당함으로서 새로운 계산 동작을 정의 하는 프로그래밍의 핵심적인 요소
- 알고리즘은 독립적인 부분 과정으로 분해함으로써 모듈화 될 수 있음

❖ 모듈(module) – 부분과정에 이름이 부여된 것

```
module NAME() is
ACTIONS
endmodule
```

그림 5.23 모듈로서 부분 과정에 이름을 붙이는 문법패턴

** "grillSteak()"는 grillSteak라고 이름 붙여진 과정을 실행하라는 것을 의미

고기를 그릴에 굽는 모듈

```
1. module grillSteak() is
2.     steakTemp ← 75
3.     while steakTemp < 135 do
4.         steakTemp ← steakTemp + 13
5.     endwhile
6. endmodule
```

그림 5.24 grillSteak 모듈

Host a Party Module

```
1. module makeDinner() is
2.     bakeCake()
3.     fixSalad()
4.     makeLemonade()
5.     grillSteak()
6. endmodule
```

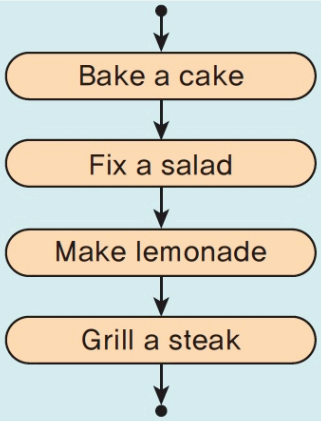


그림 5. 25 부분 모듈에 의해 표기된 'makeDinner' 모듈

❖ **이해가능성 (Understandability)**
- 모든 모듈은 독립적으로 구성
- 외부의 지식 없이 모듈에 대한 이해가 가능해야 함

❖ **캡슐화 (Encapsulation)**
- 모듈은 그 모듈 내부의 데이터에만 영향을 줄 수 있어야 함
- 모듈로 인한 오류는 그 모듈 안에서만 발생해야 함

❖ **합성 (Composition)**
- 내용을 추가하지 않고도 다른 더 큰 모듈에 결합 가능해야 함

❖ **모듈은 다양한 상황에서 사용될 수 있도록 유연하게 만들어 져야 한다.**

❖ **정형 매개변수 (formal parameter)**

- 초기값이 모듈의 입력값에 의해 정해지는 변수
- 인수 (argument) / 실 매개변수 (actual parameter) – 모듈 사용자로부터 주어진 초기값
- 모듈은 정형 매개변수를 가지지 않을 수도 있음

```
module NAME(V1, V2, ···., VN) is
    ACTIONS
endmodule
```

그림 5.27 정형 매개변수를 정의하는 문법 패턴

❖ 그릴의 초기 온도를 모듈 밖에서 제공

- 매개변수를 통해 초기 온도를 설정함

유연하게 작성된, 고기를 그릴에 굽는 내용을 담은 모듈

```
1. module grillSteak(steakTemp) is
2.     while steakTemp < 135 do
3.         steakTempl ← steakTemp + 13
4.     endwhile
5. endmodule
```

그림 5.28 보다 유연한 grillSteak 모듈

- grillSteak(94)

❖ 그릴의 초기 온도, 고기의 최종 온도, 온도 증가값을 모듈 밖에서 제공

- 매개변수를 통해 설정함

고기를 그릴에 굽는 내용을 담은 가장 유연한 모듈

```
1. module grillSteak(steakTemp, targetTemp, increaseAmount) is
2.     while steakTemp < targetTemp do
3.         steakTemp ← steakTemp + increaseAmount
4.     endwhile
5. endmodule
```

그림 5.29 최적으로 유연성을 가지는 grillSteak 모듈

- grillSteak (65, 130, 2)
- grillSteak (94, 155, 13)

❖ 프로그래밍에서 사용하는 알고리즘들

- 자료구조: **정렬, 탐색**, 각종 트리와 힙(heap)
- 알고리즘 패러다임: **백트래킹**, 동적 계획법, 분할 정복법, 분기 한정법, 그리디 알고리즘
- 트리 알고리즘: DFS (Depth-First Search), BFS (Breadth-First Search)
- 그래프 알고리즘: 탐색, **최단 경로 찾기**, 네트워크 흐름(network flow) 알고리즘
- 기타 KMP 등의 **문자열 매칭**, Pollard's rho 등의 정수론 알고리즘, 선형합동법 등의 **난수발생 알고리즘**, 해석기하/**그래픽 알고리즘, 유전 알고리즘, 기계학습 기법** 등
- 다른 분야에의 응용 : 알고리즘 트레이딩, 서포트 벡터 머신

❖ 알고리즘 관련 유용 싸이트 – 트레이닝, 문제 풀이, 경진 대회 등

- CodeUp, http://codeup.kr
- JUNGOL, http://koistudy.net
- Koistudy, http://koistudy.net
- 한국정보 올림피아드, https://www.digitalculture.or.kr/koi/KoiMain.do
- 알고스팟, https://algospot.com
- 프로그래머스, https://programmers.co.kr/learn/challenges

4.7 참고문헌

- 문봉교, 김웅섭 역, 컴퓨팅 사고 (소프트웨어를 통한 문제해결), 인피니티북스, 2017
- 알고리즘, 구글, 2017
- 알고리즘, 나무위키, 2017 https://namu.wiki/w/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

- 발표방식: ppt 자료 활용 2개 조 각각 10분 발표 (질의응답 포함)

- 문제 1 : A그룹
  - 하노이 탑이라고 불리는 유명한 문제는 1883년 프랑스 수학자 루카스가 고안해 낸 것으로 인도 베레나스에 있는 한 사원에 쌓여있는 브라흐마의 탑에 대한 유래로 널리 알려지게 되었다. 이 문제는 오늘날에도 수학과 컴퓨터 등 다양한 분야에서 출제되어지는데, 하노이 탑에 대한 원리를 설명하고, 또한 비슷한 유형의 문제에 대해 조사해보자.

- 문제 2: B그룹
  - 교재 5.3.3 단원을 보면 고기를 그릴에 굽는 프로그램에 대한 알고리즘이 나와 있다.이는 반복이라는 문법 패턴을 가지고 구성된 알고리즘이다. 이처럼 실생활에서 반복 문법 패턴을 가지고 활용되어지는 예로 3색 신호등을 들 수 있다. 이 3색 신호등에 대해 그림 5.17같이 플로우 차트로 표현하고, 또 다른 실생활에서 사용하는 반복 패턴 알고리즘 예시를 4가지 이상 조사해 발표하자.