협성대학교 소프트웨어공학과

자료구조 중간고사 대비

1 ~ 5 주차 수업 정리

이시헌

2023-10-25

1 주차 강의 요약.

1. 자료구조

① 데이터를 저장, 조직, 관리하기 위한 수단.

② 문제 해결에 필요한 부품.

③ 생각하는 방법을 훈련하는 도구. (논리의 구조, 골격을 구성하는 방법 및 스타일)

2. 추상화

① CS 에서 추상화란, 어떤 대상에서 필요한 부분을 찾아내어 간략화 하는 것.

3. 자료구조의 종류

① 선형(Linear)적 구조. (선형 데이터)

i. 배열 : 생성 이후에 크기를 늘릴 수 없음.

ii. 연결 리스트 : 배열의 문제를 해결. 나중에 배움. 참조를 통하여 다음 객체로 넘어갈 수 있도록 만들어진 자료구조.

iii. 스택 : LIFO. 후입선출.

iv. 큐 : FIFO. 선입선출.

② 비선형 데이터

i. 그래프

ii. 트리

iii. 힙 – 최대 힙 / 우선순위 큐 …

iv. 해시 테이블

v. 등등

4. 알고리즘 표기법

   ① 자연어를 이용한 서술적 표현 : 언어가 지닌 모호성으로 인해 일관성, 명확성 유지 어려움.

   ② 순서도를 이용한 도식화 : 복잡한 알고리즘 표기 어려움.

   ③ 프로그래밍 언어를 이용한 구체화 : 추가로 구체화 할 필요가 없으나 해당 언어를 모르면 이해가 어렵고, 다른 언어로 변환해야 함.

   ④ 가상 코드(혹은 의사코드. pseudo code)를 이용한 추상화 : 가장 추천!

5. 분할정복법 : 큰 문제를 작은 다수의 문제로 나누어 해결하는 것.

   ① 예 : 하노이의 탑 문제 해결.

6. 추상 : 세세한 부분을 표기하지 않고 전체적인 이미지를 표현한 것.

7. 추상 데이터 타입 : ADT 타입

   ① 어떤 데이터 타입이 어떤 작업으로 이루어졌는지 표기한 것.

   ② 세부 사항에서 벗어나 추상적으로 정의한 데이터 타입

   ③ 클래스 선언과 유사하다. 특히 ADT 는 구현 방법을 지정하지 않는다는 점에서 클래스 선언과 유사.

2 주차 강의 요약.

1. 재귀호출. (순환호출이라 하기도 함. recursion-순환)

① 자기 자신을 호출하는 것

② 무한루프에 빠지지 않도록 탈출 조건을 잘 설정해야 함.

③ 잘 다루면 강력하지만, 반복문으로 대체 가능하면 재귀호출을 피하는 것이 좋다. (위험하니까!)

④ 관계식(점화식)의 형태로 정의되는 문제들은 재귀호출로 처리 가능하다. (예: $A_n = A_{n-1} + 5$)

⑤ 등차수열의 합을 공식을 쓰지 않고 모두 더하는 경우를 예시로 들자면, 다음의 두 형태가 가능하다.

```
def a2(n):
    sum = 1
        for i in range(n):
            sum += 5
        return sum


print(a2(10))
```

```
def a(n):
    if n==0:
        return 1
    else:
        return a(n-1) + 5


print(a(100))
```

⑥ 왼쪽은 반복문으로 등차수열의 합을 구하고, 오른쪽은 재귀호출로 구했다.

2. 재귀호출의 선택 고려

① 재귀호출은 위험한 방법이며 항상 더 빠른 방법도 아니다.

② 항상 사용을 고려하되 신중히 사용하라.

③ 주로 정렬이나 탐색처럼 크기만 다른 동일한 작업을 할 때 좋다(순환호출이 곧 분할정복이다). 그러나 대부분 경우 반복으로 처리하는 것이 더 좋다.

④ 함수가 다른 함수를 2 회 이상 호출한다면 재귀호출을 사용할 이유가 없다. ($A_n = A_{n-1} + A_{n-2}$ 같은 경우를 말함)

3. n!의 풀이

```
def rec1(n):
    result = 1
     for i in range(1, n+1):
    //1 부터 n+1 까지 n 번 반복.
        result *= i

    return result


print(rec1(10))
```

```
def rec2(n):
    if n==0:
        return 1
    else:
        return n * rec2(n-1)



print(rec2(10))
```

① 반복문과 재귀의 차이가 보이는가? 결국 점화식이다.

4. 하노이탑 코드는 생략하겠다.

5. 파이썬 두 변수 교환

① 다른 언어에서는 버퍼로 사용될 변수가 필요하지만, 파이썬은 튜플을 통해 버퍼 없이 교환 가능.

```
A[k], A[last] = A[last], A[k]
//이 경우 배열의 k 와 끝 위치를 교환함.
```

6. 선택정렬 코드

① 가장 큰/작은 수를 찾아서 맨 우측/좌측에 위치시키고, 해당 자리를 무시하고 남은 자리들에 대해서 반복 수행. (재귀 가능)

② 교수님이 제공한 코드를 여기 옮기겠다.

```
def selectionSort(A,n):
    if n>1:
            last=n-1
            k = theLargest(A, last)
            A[k], A[last] =  A[last], A[k]
            selectionSort(A,last)

    def theLargest(A, last):
        largest = 0
        for i in range(last+1):
            if A[i] > A[largest]:
                largest = i
        return largest


print("selectionSort test")
A = [35, 24, 16, 21, 4, 72, 23, 9, 23, 14, 58, 12, 0]
print("A[]:      ", A)
selectionSort(A, len(A))
print("Sorted A[]:", A)
```

③ 큰 값을 찾아서 리스트의 맨 끝에 넣고, 큰 값을 찾을 범위를 끝에서 한 칸 옮긴다. 단순한 코드다. 흥미로운 점은, 리스트의 길이를 따로 넘긴다는 점이다. 두번째 인자로 len(A)를 넘기는데… 이걸 왜 넘기나?

7. 컴퓨터는 중위 연산 못함. 전위나 후위 연산 수식으로 변환 필요. – 스택 사용.

8. 깊이 우선 탐색 (DFS) – 스택 사용.

9. 이진 탐색 – 재귀호출로 표현하기 좋은 구조임. $O(\log n)$의 속도.

10. 재귀 알고리즘의 필수 구비 조건

① 경계 조건(Base condition or 종료 조건) – 재귀호출이 끝나는 지점 정의

② 재귀 호출이 가능해야 함.

③ 관계식이 더 작은 부분을 나타내도록 하는 구조를 갖고 있어야 함. (재귀 과정에서 문제가 계속 작아져야 한다.)

3 주차 강의 요약

1. 알고리즘 : 문제를 해결하는 체계적인 방법.
2. 좋은 알고리즘의 조건
   ① Time – 빨라야 한다. (횟수로 시간 측정)
   ② Space – 리소스를 적게 사용해야 한다.
   ③ 위의 두 조건은 Trade-off 관계이다. 둘 다 좋기는 어렵다는 말이다.
3. Time 측정 – 알고리즘에서 가장 많이 실행되는 부분을 기준으로 측정.
   ① 등급 – $[\, n! > 2^n > n^3 > n^2 > n\log n > n > \log n \,]$
   ② 각 등급 사이에는 넘을 수 없는 벽이 존재한다.
   ③ 최고차항만이 중요하다. 한번 두번 실행되는 부분은 시간측정에서 무시됨.
4. 알고리즘의 점근적 복잡도(Asymptotic Complexity) 표현
   ① O – 표기 : 최악의 경우의 소요시간 표기.
   ② Ω – 표기 : 최상의 경우의 소요시간 표기.
   ③ Θ – 표기 : 정확한 소요시간 표기. 최대와 최소의 교집합.
   ④ 입력 크기가 충분히 클 때를 가정해야 한다.
   ⑤ 최고차항에 어떤 수를 곱하든 같은 등급이 유지된다. ($100n^2$과 n^2은 같다)
5. 이진탐색의 경우 복잡도 분석
   ① $[\ 2^n, 2^{n-1}, 2^{n-2} \dots\dots 2^2, 2^1, 2^0\ ]$ 과 같은 관계를 $\log n$으로 표현 가능.
   ② 즉, 이진탐색에서 최악의 경우는 $\Theta(\log n)$이다. ($\Omega(\log n)$)
   ③ 최선의 경우는 $\Theta(1)$이다. (탐색 시작과 동시에 발견)
   ④ 수식어 없이 종합하여 표현하면, $O(\log n)$이다.
6. 참고 – $\log n$ / n / $n\log x$ / n^2 : 이진탐색 / 순차탐색 / 빠른정렬 / 느린정렬

4 주차 강의 요약 : 파이썬 문법 정리.

1. 인터프리터 언어지만 컴파일도 지원.
2. NULL 대신 None 사용.
3. 자료형이 다양함.
   ① 특이하게도 복소수(complex)타입이 내장 자료형으로 존재.
   ② 시퀀스 : 여러 개의 연속적인 데이터 집합. - 조금 있다가 심도 있게 설명 예정.
      i. 리스트(list) : C 언어의 배열을 강화. 특이하게도 대괄호로 초기화 함.
         A. List = [1 , 2 , 3 , 4 , 5]
      ii. 문자열(str)타입을 내장 자료형으로 지원, 문자도 문자열로 처리.
      iii. 튜플(tuple) : 소괄호로 묶은 데이터 집합.
      iv. range 타입
   ③ 딕셔너리(dictionary) : key 와 value 가 한 쌍으로 묶인 데이터들의 집합. 이때 key 는 중복될 수 없다.
      i. 예 : { 3.14 : “phi”, 4.5 : “score” }
      ii. JS 나 PHP 의 연관배열 생각하라.
   ④ 집합 : 중복이 허용되지 않는 순서의 개념이 없는 자료형. 중괄호로 묶어야 함.
      i. 예 : { 1 , 2 , 3 }
4. 변수에 대하여
   ① 파이썬에서는 변수가 물리적 주소가 할당된 공간을 갖지 않는다.
   ② 파이썬에서는 모든 자료가 클래스에서 파생된 객체이고, 변수 이름은 그 객체를 참조할 수 있게 하는 참조자로서 기능한다.
   ③ 즉, 변수가 참조만 수행하기에 타입이 존재할 이유가 없다.
   ④ 이때 변수가 가리키지 않는 객체는 가비지 컬렉터에 의해 소멸된다.
5. 나눗셈 연산자 변경됨.
   ① C 언어 : 5/2 == 2 혹은 (double)5/2 == 2.5

② 파이썬 : 5/2 == 2.5 그리고 5//2 == 2

i. /는 실수 나눗셈. //는 정수 나눗셈의 몫 구함.

6. 이항 연산자 추가됨.

① **은 제곱 연산자.

i. 예 : 5**2 == 25

7. 단항 연산자 제거됨.

① ++, -- 제공 안함. +=과 -=으로 대체해야 함.

8. 논리 연산자 변경

① ||, &&, ! -> or, and, not

9. 'in'과 'not in'연산자 추가됨.

① 'a' in 'banana' == true

② 리스트 타입 같은 시퀀스 자료형에도 사용 가능

```
a = [ 0,1,2,3,4,5 ]
if 3 in a:
    while 4 in a:
```

10. 입력은 input( )함수로 수행.

① 문자열만 입력 받으니 다른 타입을 원한다면 변환 필요.

② 변환 예 : age = int(input("나이는?:"))

i. 이 경우, "나이는?:"이란 문자열 뒤에 사용자 입력 수행. input 함수가 리턴한 문자열을 int()함수가 정수로 변환 수행함. (int 함수는 내장 전역 함수)

11. 출력은 print( )함수로 수행.

① 개행(₩n, Newline)이 기본적으로 포함되어 있음. 개행을 하지 않으려면, "₩n"대신 사용할 문자를 지정해야 함.

i. print(변수명) #두번째 인자의 디폴트 값이 "₩n"임.

ii. print("game", end=" ") #두번째 인자로 원하는 문자 넘기면 됨.

iii. print("값은=", n) #복수의 객체 출력시 쉼표로 연결 가능(n 은 변수)

12. 코드블록

① 기본적으로 들여쓰기로 코드블록이 구분되지만, 한 코드블록을 한 줄로 작성해도 됨.

② 한줄 쓰기 예시 : if value%2==0: print(value)

13. 반복(looping)

① range( )타입의 사용법을 잘 숙지하라.

i. range(시작, 종료+1, step)

ii. 시작과 step 은 생략 가능. 생략하면 **시작은 0, step 은 1 이 기본값이다.**

iii. 종료+1 인 이유는, 0 부터 시작시 해당 인자는 반복 횟수이기 때문이다.

② for n in range(5): print(n) #실행시 01234 출력

③ range 대신 어떤 컬렉션 타입도 사용 가능. (아마도?)

i. for item in [1,2,3,4,5,99,516534]: print("값=", item)

④ range( ) 앞에 있는 것은 일종의 이터레이터 역할을 하는 지역변수임.

14. 컬렉션 자료형 (항목 3 번 심화)

① 데이터가 여럿 모인 자료형을 지칭한다.

② 문자열/리스트/튜플/딕셔너리/집합

③ 문자열(str) : 리스트와 동일함.

④ 리스트(list) : 스마트한 배열임. (C++의 STL 과 유사)

i. 다양한 내장 함수 지원. 특이한점은, 파이썬에서는 음수 인덱스 지원. -1 은 가장 끝의 원소임.

ii. C++의 .push_back( )은 여기서 .append( )임.

⑤ 튜플(tuple) : 크기와 값을 변경할 수 없는 리스트임.

i. 예 : t = (0,3,7)

ii. 변경이 불가능하기에 메모리 효율적으로 동작하게 설계됨.

⑥ 딕셔너리(dict) : 키(key)와 관련된 값(value)으로 이루어진 항목(entry)들의 집합.

i. 다양한 내장 함수가 있다.

ii. 예 : map = {'김연아':'피겨', ……}

⑦ 집합(set)

i. 합, 차, 교집합 등을 만드는 다양한 내장 함수 지원됨.

ii. 집합 연산으로 손쉽게 해결 가능한 다양한 문제들이 있음.

iii. frozenset 은 내용을 변경할 수 없는 집합을 의미한다.

⑧ 빈 객체 생성법 정리

```
l = []  #빈 리스트 생성
s = {}  #빈 딕셔너리 생성
se = set()  #빈 집합 생성  (빈 집합은 중괄호로 생성 불가함. 주의!)
str = str() #빈 문자열 생성
```

15. 파이썬 함수

① def 키워드를 사용하여 정의 가능. 다른 언어와 거의 유사함.

② 특이하게도 함수가 복수의 리턴을 할 수 있다. (튜플로 묶어서 리턴해준다)

③ 인자를 넘길 때, 키워드 사용시 인수의 입력 순서가 상관 없다.

```
def sum_range(begin, end, step=1):
#이런 방식으로 선언하고, step 처럼 디폴트 인수 지정 가능.

sum_range(step=3, begin=2, end=10)  #이런 방식으로 호출 가능.
```

i. 위처럼 디폴트 인수 지정 가능.

ii. print 함수의 end=" " 인수도, 키워드 인수 방식으로 공백을 넘긴 것.

16. 변수의 유효 범위는 다른 언어와 유사함.

① 변수의 유효 범위

내장 범위	언어의 일부로 정의된 변수와 리터럴들 – 프로그램 전체에서 사용 가능.
전역 범위	함수나 클래스 밖에서 생성된 경우 – 어디에서도 사용 가능
지역 범위	함수나 클래스 안에서 생성된 경우 – 해당 코드블록 안에서 사용 가능.
인스턴스 범위	클래스의 데이터 멤버로 생성된 변수 – 클래스 내의 다른 메소드에서만 접근 가능

② 전역변수의 경우 다른 언어와 약간 다르게 동작한다. 파이썬의 변수가 모두 참조자임을 기억하라.

i. 파이썬은 지역 공간에서 전역변수의 이름에 무언가를 대입하려고 시도하면, 동일 이름의 지역변수를 하나 더 만든다. 단, 읽을 때는 다른 언어처럼 전역변수에 잘 접근한다.

ii. 지역 공간에서 전역변수에 쓰기를 수행하고 싶으면, 지역 공간에서 전역변수의 이름 앞에 global 키워드를 작성하면 된다 (global num = 100 처럼)

17. 모듈 : .py 인 파이썬 코드가 들어있는 파일.

18. import 키워드로 파일을 불러올 때, 그 파일의 이름이 모듈의 이름이다.

① min_max.py 파일을 불러올 때는, import min_max 라고 씀. (확장자 안붙임)

i. 이 경우 멤버 참조 연산자('.')로 모듈 내의 함수에 접근 가능.

② from 키워드를 함께 사용시, 점으로 멤버를 참조 안해도 됨.

i. from min_max import * (별 자리에 함수의 이름을 적어도 됨. 현재는 모듈의 모든 것을 불러온 상태.)

19. 클래스

① class 키워드로 정의함. 예시 코드를 보면 이해가 갈 것이다.

```
class Car:
    def __init__(self, color, speed=0):  #생성자 함수 이름은 __init__으로
    예약되어 있다.
        self.color = color
        self.speed = speed

#연산자 오버로딩 예시 (__eq__는 예약어임. 오버로딩 가능한 연산자들에게는 각각
예약어가 지정되어 있음)
        def __eq__(self, carB): return self.color == carB.color

#객체 생성 예시
car1 = Car('black', 0)
car2 = Car('green')    #speed 에 대하여 디폴트 인수 사용됨.
```

② 상속을 위해서는 생성자가 부모 클래스의 생성자를 호출해야 한다. 위의 Car 클래스를 상속받는 SuperCar 클래스를 만들면 다음과 같다.

```
Class SuperCar(Car):  #상속해주는 부모 클래스 명을 괄호에 명시
    def __init__(self, color, speed=0, bTurbo=True):
        super().__init__(color,speed) #부모 클래스 생성자 호출
        self.bTurbo = bTurbo #새 변수 생성 및 초기화
```

③ 이때 super( )함수는 객체의 부모 클래스로 예약되어 있는 함수다. (괄호 안에 적은 객체, 즉, 부모 클래스 자체임)

20. 재정의(Overriding) : 자식 클래스가 상속받은 메소드를 재정의 하는 것.

① def 키워드 쓰고 함수를 한번 더 정의하면 덮어씌워짐.

② 자식 클래스 내부에서 재정의 하면, 자식에서는 재정의된 함수 사용.

5 주차부터 7 주차까지의 내용은 필기와 교재를 직접 보기를 바란다.

파이썬 문법 정리에 너무 많은 시간을 쏟았다.

정리 끝! (*)