Python 자료구조 알고리즘

2019.12.21

강진혁

ROBOT MEDIA LABORATORY

강의 진행 방식

Robot Media Laboratory

1일차(4h)	파이썬 기억 되살리기 - 오리엔테이션, 개발 환경 구축, Python기본적내용 정리
2일차(4h)	파이썬 기억 되살리기 - 클래스의 전반적인 내용학습 및 Python코드를 data관점에서 기본적 전 기능 사용 코딩
3일차(4h)	알고리즘과 자료구조의 기초 이론학습 -python을 이용한 구현
4일차(4h)	스택 구조 이해 및 구현
5일차(4h)	큐 구조 이해 및 구현
6일차(4h)	트리 구조 이해 및 구현
7일차(4h)	그래프의 개념정리, 정렬 및 검색 알고리즘 전반적인 이론 복습
8일차(4h)	

강의 진행 방식

Robot Media Laboratory

- **수업 진행**
  - 수업 시간: 4시간, 50분수업 10휴식 09:30~13:20(쉬는시간 40분)
  - Python 기반으로 자료구조 진행
  - 2가지 방식의 강의 자료 활용

lecture_kjh@naver.com

- **0월 오전 자료구조 수업 듣는 OOO입니다.**

동적할당

동적할당
Robot Media Laboratory

- 동적 메모리 공간 할당
- 정적 할당과는 다른 방식의 메모리 공간 할당
- 프로그램 동작 중 메모리 공간을 가변 할당 가능
- 필요 정도에 따라 메모리 할당 후 제거 하는 방식
- 메모리 공간 필요 정도가 불분명 할 시 사용
- 임시적인 메모리가 필요 할 시 사용
- 동적 할당된 메모리는 이름이 없는 변수와 동일

메모리 영역

메모리 영역-메모리 관리 원칙

Robot Media Laboratory

① 메모리 관리의 주체는 운영체제이다

- 반드시 운영체제를 통해서만 메모리를 할당

② 운영체제는 메모리가 있는 한은 할당요청을 거절하지 않는다.

③ 한번 할당된 메모리 공간은 절대로 다른 목적을 위해서 재할당되지 않는다.

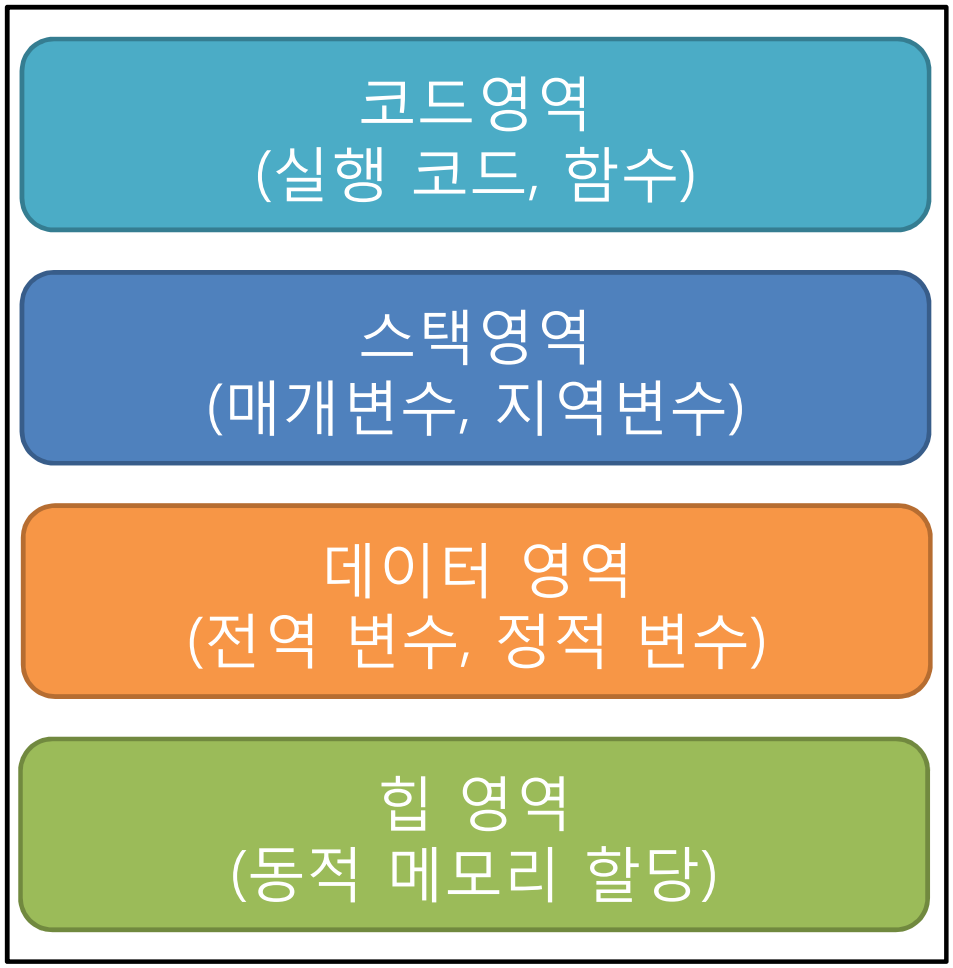
- 할당된 공간은 안정적

④ 응용 프로그램이 할당된 메모리를 해제하면 운영체제는 이 공간을 빈 영역으로 인식한다.

ROBOT MEDIA LABORATORY

메모리 영역

Robot Media Laboratory



:메모리 영역

- 코드영역: 프로그램 실행코드, 함수들이 저장되는 영역, OR 텍스트영역
- 스택 영역: 매개변수,지역변수,중괄호(블록) 내부에 정의된 변수들이 저장되는 영역
- 데이터 영역: 전역 변수, 정적 변수들이 저장되는 영역
- 힙 영역: 프로그램이 실행 되는 동안 동적으로 메모리 할당할 수 있는 영역

ROBOT MEDIA LABORATORY

동적 할당과 정적 할당

동적 할당과 정적 할당

동적 할당

- 힙영역에 할당
- 런타임 중(실행시간)에 이루어진다.

- 장점: 경제적이다.
- 단점: 더 이상 사용하지 않을 때 명시적으로 메모리를 해제해 주어야 한다.

정적 할당

- 컴파일로 고정되는 DATA영역에 할당
- 컴파일 타임에 이루어진다.

- 장점:실행도중 해제되지않고, 자동 회수된다
- 단점:메모리 크기를 조절 할 수 없다.

프로그래밍 언어

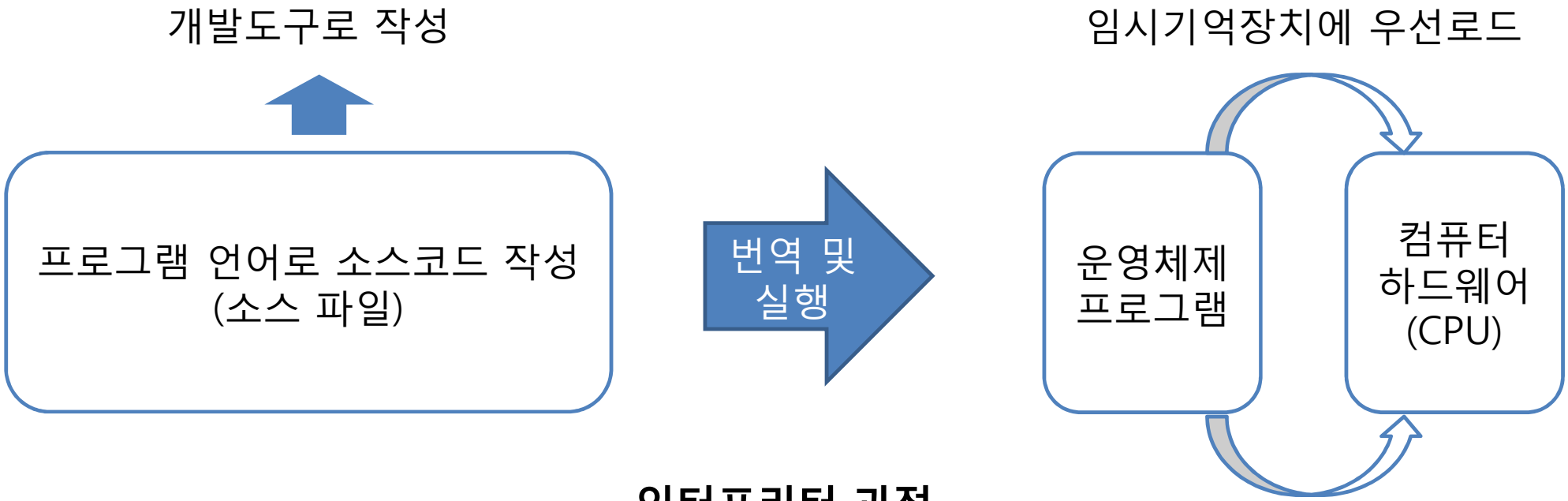
인터프리터 언어

컴파일러 언어

프로그래밍 언어-인터프리터언어

Robot Media Laboratory

- 중간 과정 없이 원시 프로그램을 직접 저급 언어로 바꾸면서 동시에 실행



인터프리터 과정

프로그래밍 언어-인터프리터 언어

Robot Media Laboratory

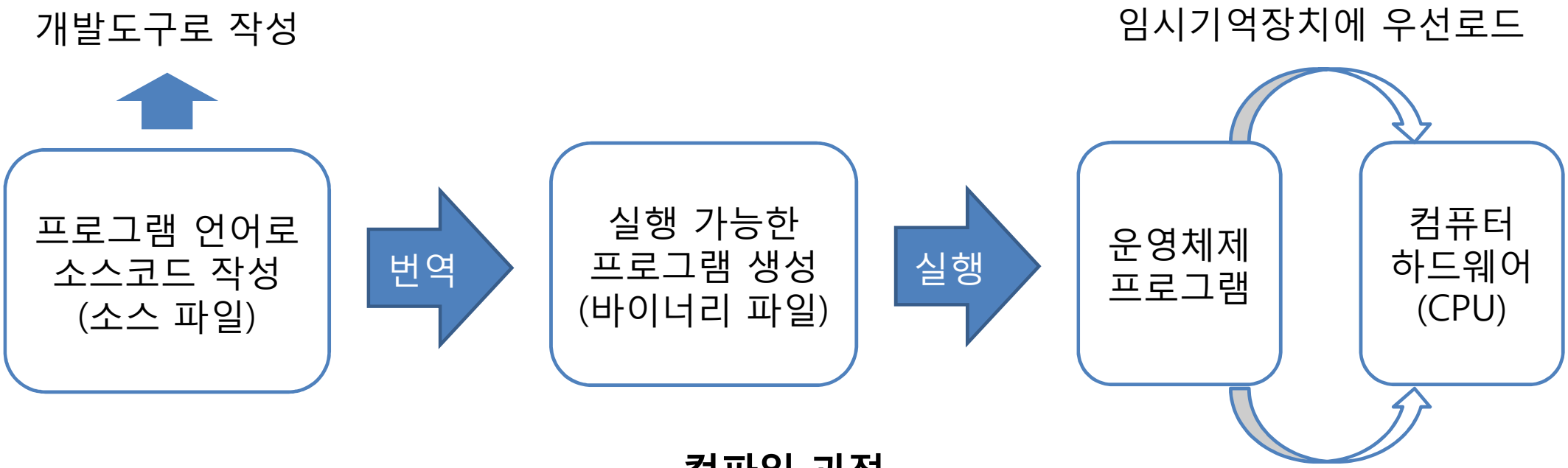
- **인터프리터 언어의 장점**
  - 실행을 위해 완전한 기계어 번역을 기다리지 않고 필요 시 마다 실행
  - 프로그램이 실행될 때까지 원시 언어 형태를 유지하므로 메모리 절약

- **인터프리터 언어의 단점**
  - 재실행 시 매번 원시 프로그램을 디코딩 처리를 위한 시간이 필요

프로그래밍 언어-컴파일러 언어

Robot Media Laboratory

- 고급언어를 저급언어로 번역하여 모듈을 만들고, 모듈을 링크, 로드 하여 실행



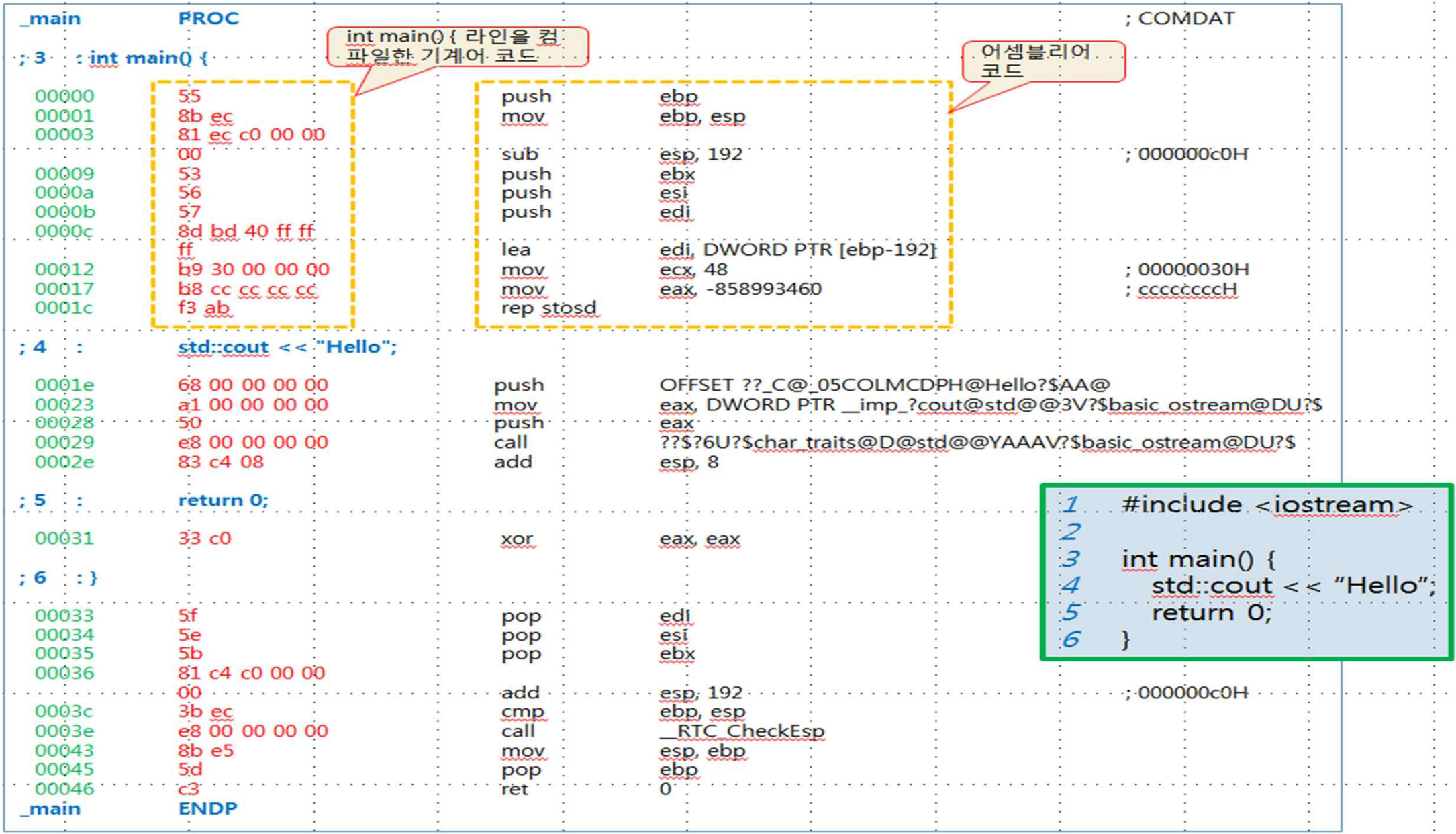
컴파일 과정

프로그래밍 언어-컴파일러 언어

Robot Media Laboratory

- **컴파일 언어의 장점**
  - 번역된 목적 코드 저장 가능
  - 컴파일 한 후 바로 재실행이 가능
  - 재사용 프로그램인 경우, 한번 컴파일 한 후에는 빠르게 재실행에 따라 실행시간 단축

- **컴파일 언어의 단점**
  - 기계어로 변환하는데 많은 시간 소요
  - 한 줄의 원시 프로그램이 때로는 몇 백 줄의 기계어로 번역되어 메모리 낭비 발생


int main() { 라인을 컴파일한 기계어 코드
어셈블리어 코드
_main PROC ; COMDAT
; 3 : int main() {
00000 55 push ebp
00001 8b ec mov ebp, esp
00003 81 ec c0 00 00 00 sub esp, 192 ; 000000c0H
00009 53 push ebx
0000a 56 push esi
0000b 57 push edi
0000c 8d bd 40 ff ff ff lea edi, DWORD PTR [ebp-192]
00012 b9 30 00 00 00 mov ecx, 48 ; 00000030H
00017 b8 cc cc cc cc mov eax, -858993460 ; ccccccccH
0001c f3 ab rep stosd
; 4 : std::cout << "Hello";
0001e 68 00 00 00 00 push OFFSET ??_C@_05COLMCDPH@Hello?$AA@
00023 a1 00 00 00 00 mov eax, DWORD PTR __imp_?cout@std@@3V?$basic_ostream@DU?$
00028 50 push eax
00029 e8 00 00 00 00 call ??$?6U?$char_traits@D@std@@YAAAV?$basic_ostream@DU?$
0002e 83 c4 08 add esp, 8
; 5 : return 0;
00031 33 c0 xor eax, eax
; 6 : }
00033 5f pop edi
00034 5e pop esi
00035 5b pop ebx
00036 81 c4 c0 00 00 00 add esp, 192 ; 000000c0H
0003c 3b ec cmp ebp, esp
0003e e8 00 00 00 00 call __RTC_CheckEsp
00043 8b e5 mov esp, ebp
00045 5d pop ebp
00046 c3 ret 0
_main ENDP
1 #include <iostream>
2
3 int main() {
4 std::cout << "Hello";
5 return 0;
6 }

Python

Python

Robot Media Laboratory

- 인터프리터 언어
- 객체지향적 동적 타이핑 대화형 언어
- 동적 타이핑의 일종인 덕 타이핑을 사용하는 언어
- 다양한 프로그래밍 패러다임을 지원하는 언어

❖ 덕 타이핑:객체의 변수 및 메소드의 집합이 객체의 타입을 결정하는 것
❖ 프로그래밍 패러다임:프로그래밍의 방법론

Python-표준 입출력-print()

Robot Media Laboratory

print(출력문자열)

출력 문자열 특성:

- 문자열 속에 ' 기호가 있는 경우에는 “ ”로, “ 기호가 있는 경우에는 ‘ ’를 사용하면 편리
- 콤마(,)로 문자열을 나열할 경우 공백(기본값)이 자동으로 추가
- 더하기(+) 기호 및 공백없이 문자열 연결 가능
- 긴 문자열은 ₩ 기호로 사용하여 여러 줄에 작성 가능

Python-표준 입출력-print()

Robot Media Laboratory

- 개행, 공백, 특수기호를 포함하여 입력한 그대로 화면에 출력하기 위해서는 ‘‘‘ ’’’ 또는 “”” ””” 기호를 사용
- end 인자를 사용하면 print함수 마지막 효과를 변경 (기본값은 개행)
- Sep 인자를 사용하면 콤마로 구분된 문자열을 다르게 결합 (기본값은 공백)
- File 인자를 사용하면 출력 결과를 파일, 표준 에러 처리로 전달
- 이스케이프 문자(Escape character)는 ₩문자를 사용하여 출력

Python-표준 입출력-print()

Robot Media Laboratory

- 문자열 객체의 다양한 메소드(함수)를 활용하여 출력 모양을 변경 가능
- 정수, 실수, 불 자료형을 문자열로 변경할 필요없이 바로 출력 가능
  (명시적으로 변경하기 위해서는 str()함수를 사용하면 됨)
- 수식과 리스트 같은 복잡한 내용도 쉽게 출력 가능
- 3.6버전 이상부터는 f-string기능 존재

Python-표준 입출력-input()

Robot Media Laboratory

input(입력 문자열)

- 입력 문자열은 출력 문자열의 특성과 동일
- 입력된 DATA는 <str>자료형으로 입력
- .split(입력구분자)를 이용하여 복수입력 가능(기본 공백)
- map(자료형,input(입력문자열).split(입력구분자))로 복수 형 변환 가능

Python-자료형

Robot Media Laboratory

종류	설명	문법 예
str	문자열: 이뮤터블 방식의 일련의 유니코드 코드포인트.	"Wikipedia" """Spanning multiple lines"""
bytearray	뮤터블(mutable) 방식의 일련의 바이트.	bytearray(b"Some ASCII") bytearray([119, 105, 107, 105])
bytes	이뮤터블(immutable) 방식의 일련의 바이트.	b"Some ASCII" bytes([119, 105, 107, 105])
list	뮤터블(mutable) 방식의 리스트. 혼합 형태를 포함할 수 있다.	[4.0, 'string', True]
tuple	이뮤터블(immutable) 방식. 혼합 형태를 포함할 수 있다.	(4.0, 'string', True)
set, frozenset	순서가 정해지지 않은 집합. 중복 허용 안 함. frozenset은 이뮤터블(immutable)이다.	{4.0, 'string', True} frozenset([4.0, 'string', True])
dict	뮤터블(mutable) 방식의 연관 배열의 키와 값 쌍.	{'key1': 1.0, 3: False}
int	이뮤터블(immutable) 방식의 정수로서 크기는 무제한.	42
float	이뮤터블(immutable) 방식의 부동소수점 수 (시스템 정의 정밀도).	3.1415927
complex	이뮤터블(immutable) 방식의 복소수. (실수와 허수)	3+2.7j
bool	이뮤터블(immutable) 방식의 진리값.	True False

❖ 뮤러블:가변객체(변수) 이뮤러블:불변객체(상수)

Python-변수

Robot Media Laboratory

- **데이터를 저장해 놓는 일종의 저장 공간**
- **데이터를 저장할 수 있도록 이름을 할당 받은 메모리 공간**
- **언제든지 다시 접근하거나 그 값을 변경 가능**
- **변수 작성 규칙**
  - 알파벳, 숫자, 언더스코어(_)로 구성
  - 알파벳은 대/소문자 구분
  - 한글 사용 가능
  - 변수명의 시작은 숫자로 할 수 없음
  - 공백이나 특수 기호는 포함 할 수 없음
  - Python 예약어 사용 불가

Python-연산자

Robot Media Laboratory

순위	연산자 종류	연산자
1	큐플,리스트,딕셔너리	(), {}, []
2	첨자,슬라이싱,인수,속성등	c [index], c [index1, index2], f(x), obj.m
3	거듭제곱	**
4	양수,음수,비트부정	+, -, ~
5	곱,몫,정수 몫,나머지	*, /, //, %
6	더하기,빼기	+, -
7	시프트연산	<<, >>
8	비트 곱	&
9	비트 XOR	^
10	비트 합	|
11	멤버,아이디,관계	in, not in, is, is not, <,<=,>,>=,==,!=
12	논리 부정	Not
13	논리 곱	and
14	논리 합	or
15	삼항	참 if 조건 else
16	람다	lambda

Python-조건문-if-else

Robot Media Laboratory

- **if 조건식:**
  명령문
- **elif 조건식:**
  명령문
- **else:**
  명령문

Python-반복문

Robot Media Laboratory

- **while문**
  - 조건식이 특정 조건을 만족할 때까지 계속해서 명령문을 실행하는 반복문
  - while 조건식:
    명령문
  - 무한루프는 True를 조건식에 작용하여 사용한다
- **for문**
  - 튜플이나 리스트, 문자열 등을 원하는 횟수만큼 실행하는 반복문
  - for 변수 in 문자열(or 튜플 or 리스트):
    명령문
- **range()함수**
  - range(마지막정수), range(시작정수=0,마지막정수, 증감식=1)

- append(value) 리스트 끝에 값을 추가
- extend(iter) 리스트 끝에 list, tuple, dict의 값을 하나씩 추가
- insert(idx, value) 특정 인덱스 위치에 값을 추가
- pop([idx]) 마지막 인덱스의 값을 반환 후 삭제, 인덱스 번호를 지정 가능
- remove(value) 특정 값에 해당하는 것을 찾아 삭제
- clear() 모든 값을 삭제하여 빈 리스트만 남김
- count(value) 리스트에서 일치하는 값의 수를 반환
- index(value) 리스트에서 일치하는 값의 인덱스 번호를 반환
- reverse() 리스트의 모든 값을 뒤집어 나열
- sort([reverse=False]) 리스트의 값을 오름차순(False), 내림차순 (True) 정렬

선언

- **def 함수명(매개변수1=기본값,...):**
  실행 명령문
  return 반환값

호출

- **함수명(전달인수1,...)**
- **함수명(매개변수1=전달인수1,...)**

Python-함수-가변 매개변수

Robot Media Laboratory

선언

- **def 함수명(*매개변수1,...):**

  매개변수1(튜플화)

  return 반환값

호출

- **함수명(전달인수1(복수입력가능),...)**

Python-함수-딕셔너리 매개변수

Robot Media Laboratory

선언

- **def 함수명(**딕셔너리 매개변수,...):**

  딕셔너리 변수

  return 반환값

호출

- **함수명(key=전달인수1,...)**

선언

- **def 함수명(매개변수,...):**

  data1,data2

  return 반환1,반환2,...

호출

- **함수명(key=전달인수1,...)**

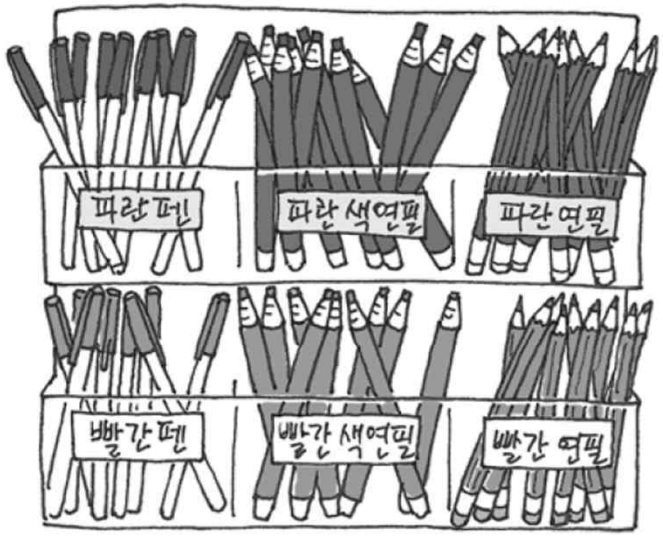
자료구조

자료구조

Robot Media Laboratory

- 자료를 효율적으로 사용하기 위해서 자료의 특성에 따라서 분류하여 구성하고 저장 및 처리하는 모든 작업

[나쁜 자료구조]



[좋은 자료구조]

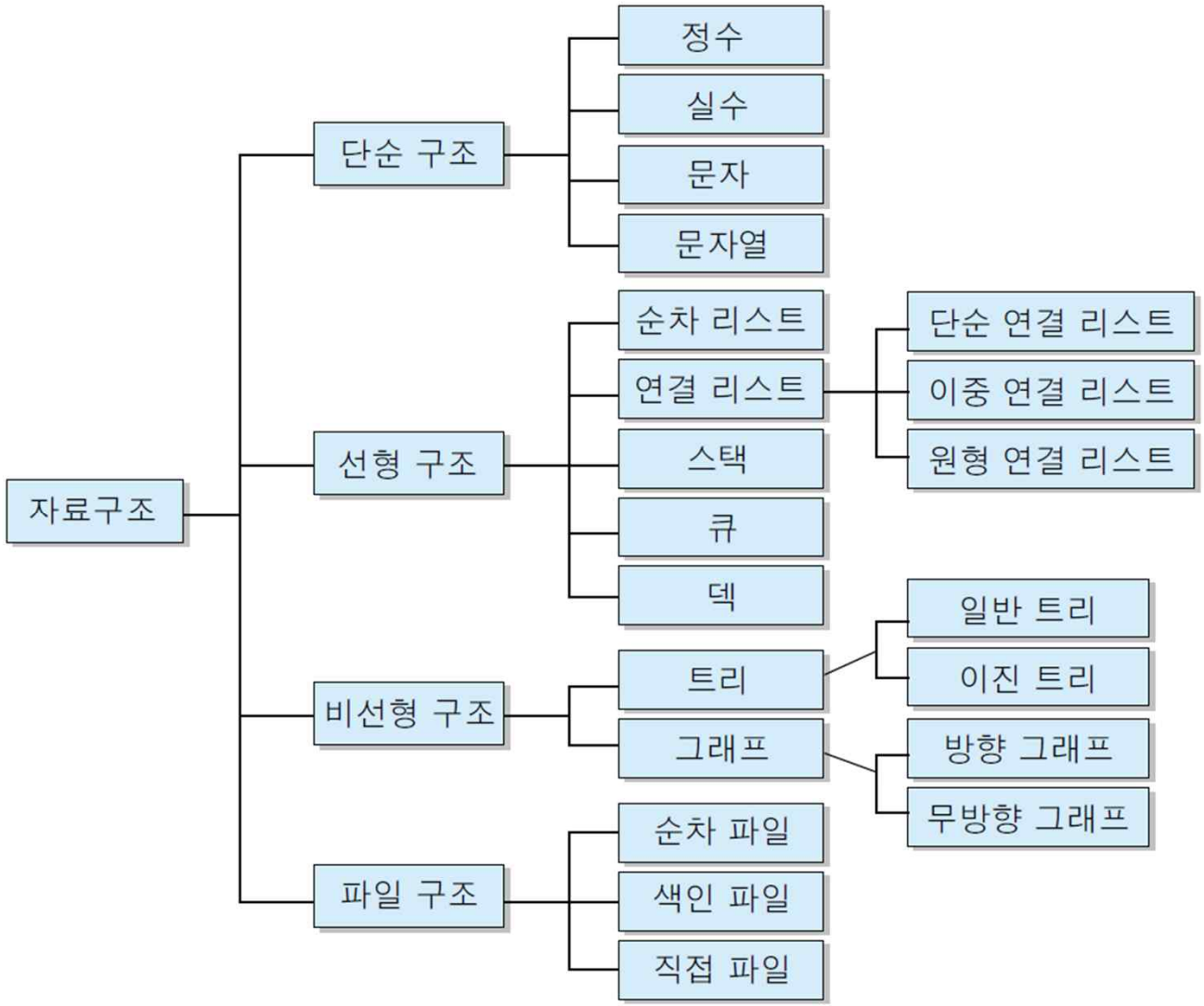
자료구조-형태에 따른 분류

Robot Media Laboratory

- **단순 구조**
  - 정수, 실수, 문자, 문자열, 등의 기본 자료형
- **선형 구조**
  - 자료들 간의 앞뒤 관계가 1:1의 선형 관계
  - 리스트, 연결 리스트, 스택, 큐, 덱 등
- **비선형 구조**
  - 자료들 간의 앞뒤 관계가 '1:다', 또는 '다:다'의 관계
  - 트리 그래프 등
- **파일 구조**
  - 레코드의 집합인 파일에 대한 구조
  - 순차파일, 색인 파일, 직접파일 등

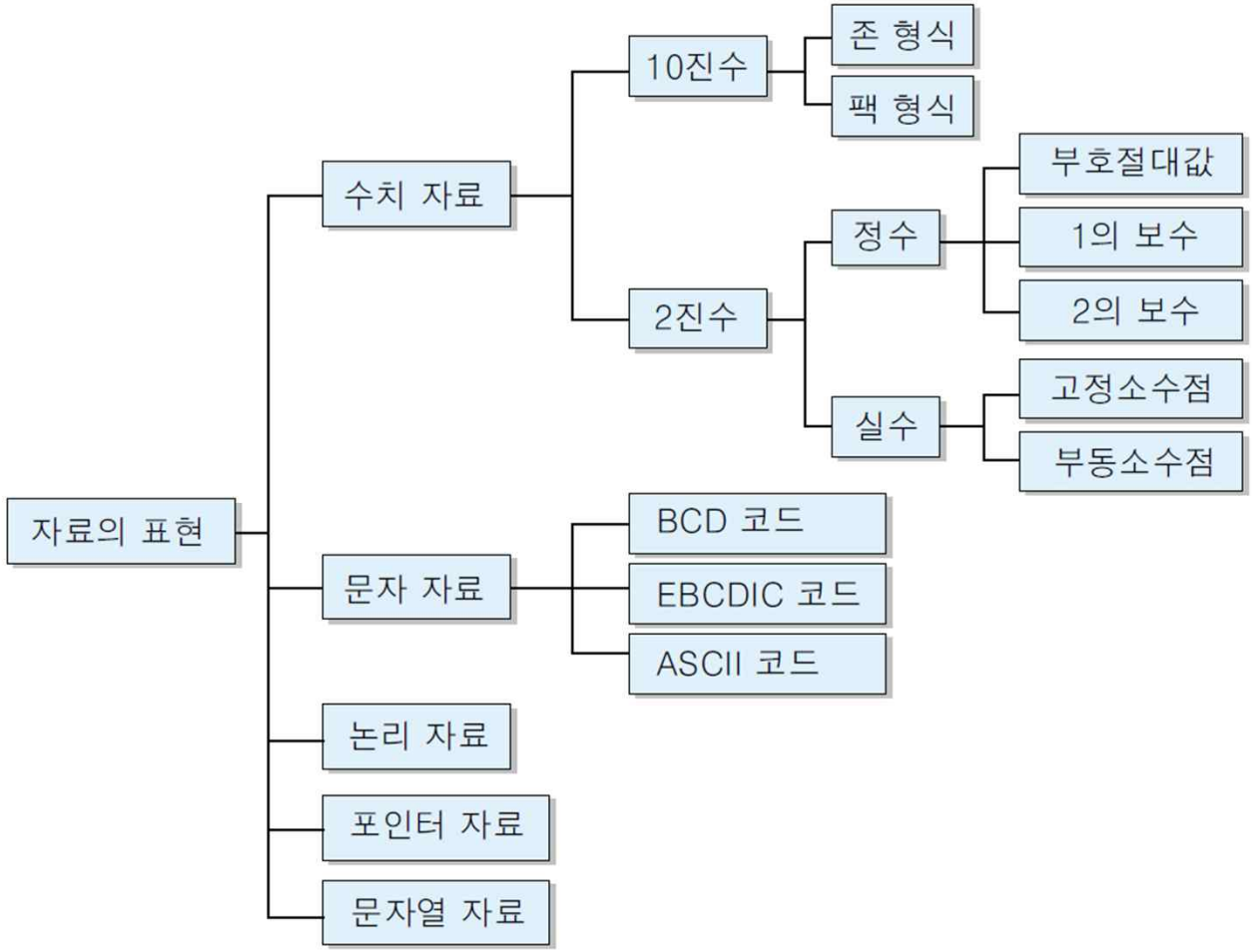
자료구조

Robot Media Laboratory



자료구조

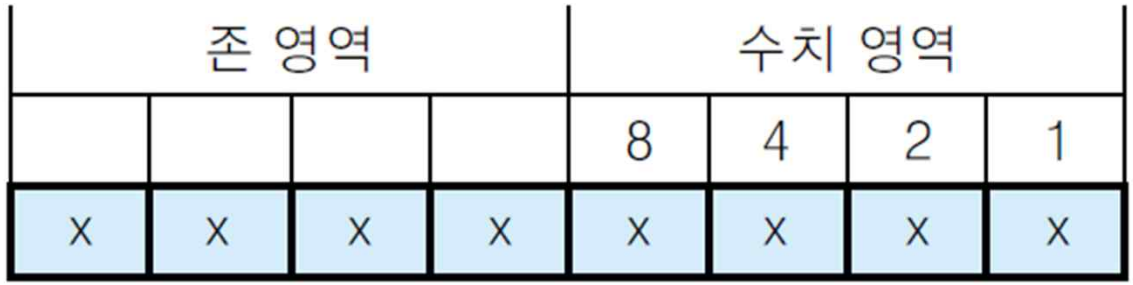
Robot Media Laboratory



존(Zone) 형식의 표현

- 10진수 한 자리를 표현하기 위해서 1바이트(8비트)를 사용하는 형식
- 존 영역
  - 상위 4비트
  - 1111로 표현
- 수치 영역
  - 하위 4비트
  - 표현하고자 하는 10진수 한 자리 값에 대한 2진수 값을 표시

- 존 형식의 구조

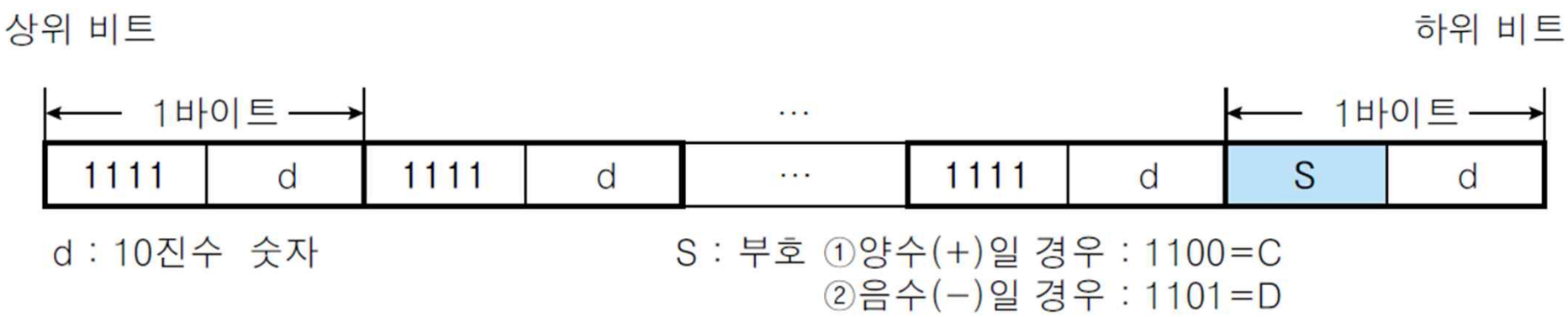


자료구조-10진수의 표현

Robot Media Laboratory

여러 자리의 10진수를 표현하는 방법

- 10진수의 자릿수만큼 존 형식을 연결하여 사용
- 마지막 자리의 존 영역에 부호를 표시
  - 양수(+) : 1100
  - 음수(-) : 1101

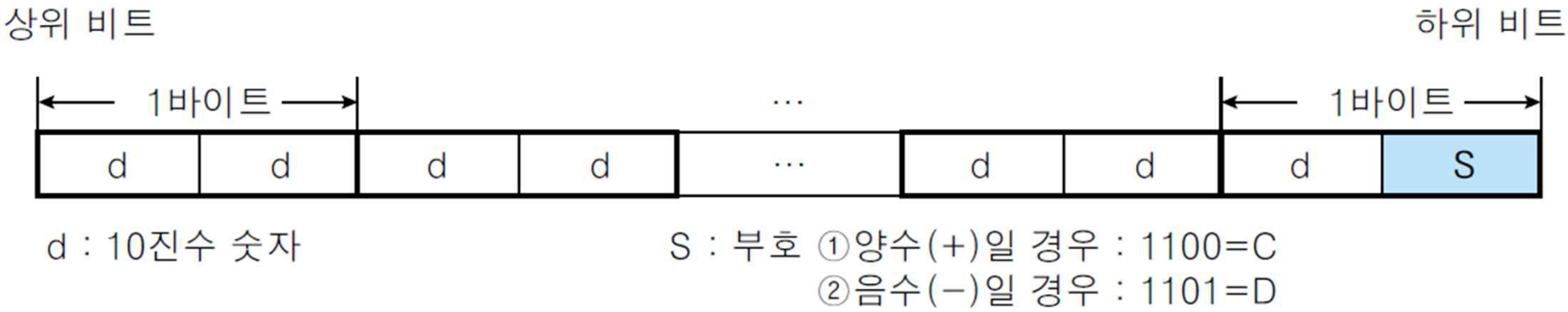


자료구조-10진수의 표현

Robot Media Laboratory

팩(Pack) 형식의 표현

- 10진수 한 자리를 표현하기 위해서 존 영역 없이 4비트를 사용하는 형식
- 최하위 4비트에 부호를 표시
  - 양수(+) : 1100



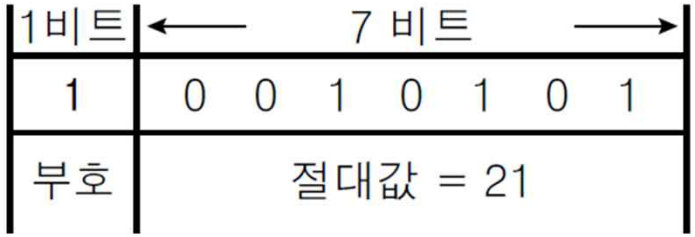
n비트의 부호 절대값 형식

- 최상위 1비트 : 부호 표시
  - 양수(+) : 0
  - 음수(-) : 1
- 나머지 n-1 비트 : 이진수 표시

- 1바이트를 사용하는 부호 절대값 형식의 예

① +21

② −21



1의 보수(1' Complement) 형식

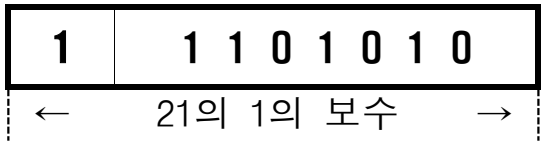
- 음수의 표현에서 부호 비트를 사용하는 대신 1의 보수를 사용하는 방법
- n비트의 2진수를 1의 보수로 만드는 방법
  - n비트를 모두 1로 만든 이진수에서 변환하고자 하는 이진수를 뺀다.
  - 예) 10진수 21을 1의 보수로 만들기(1바이트 사용)

```
  1 1 1 1 1 1 1 1
- 0 0 0 1 0 1 0 1   ☜ 21의 2진수 값
-----------------
  1 1 1 0 1 0 1 0   ☜ 21의 1의 보수
```

- 1바이트 를 사용하는 1의 보수 형식의 예

+21

-21



2의 보수(2' Complement) 형식

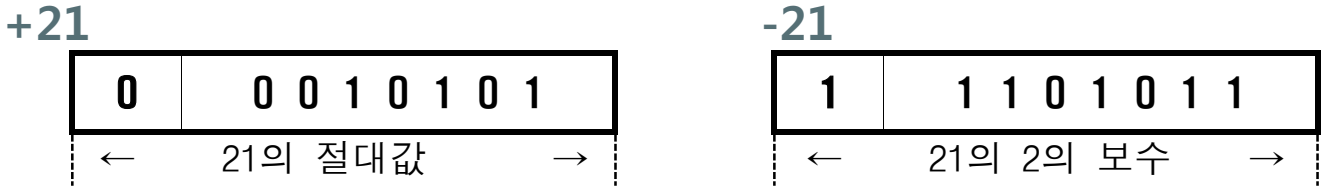
- 음수의 표현에서 부호 비트를 사용하는 대신 2의 보수를 사용하는 방법
- n비트의 2진수를 2의 보수로 만드는 방법
  - 1의 보수에 1을 더해준다.
  - 예) 10진수 21을 2의 보수로 만들기(1바이트 사용)

```
  1 1 1 1 1 1 1 1
- 0 0 0 1 0 1 0 1   ☜ 21의 2진수 값
-----------------
  1 1 1 0 1 0 1 0   ☜ 21의 1의 보수
+               1
-----------------
  1 1 1 0 1 0 1 1   ☜ 21의 2의 보수
```

자료구조-2진수의 정수 표현

Robot Media Laboratory

- **2의 보수(2' Complement) 형식**
  - 1바이트를 사용하는 2진 보수 형식의 예



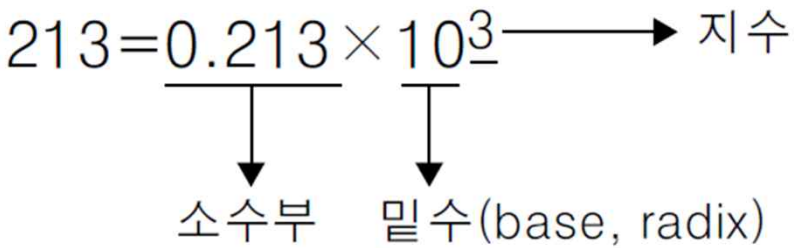
- **2진수 정수의 세 가지 표현 방법에서 양수의 표현은 같고 음수의 표현만 다르다.**

고정 소수점 표현

- 소수점이 항상 최상위 비트의 왼쪽 밖에 고정되어 있는 것으로 취급하는 방법
- 고정 소수점 표현의 **00010101**은 **0.00010101**의 실수 값을 의미

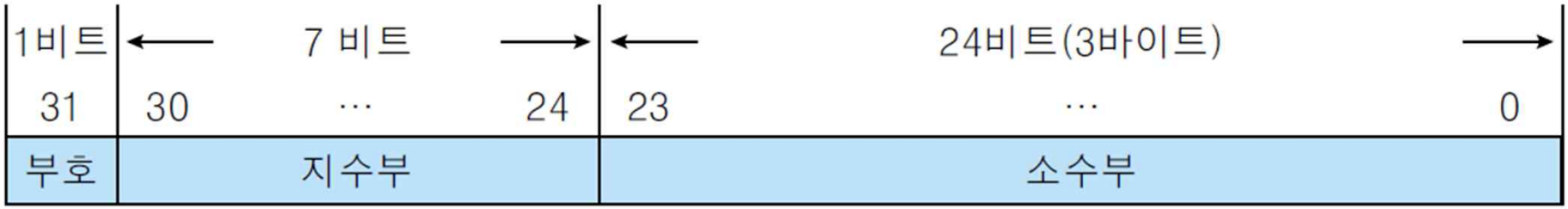
부동 소수점 형식의 표현

- 고정 소수점 형식에 비해서 표현 가능한 값의 범위가 넓다
- 실수를 구분하여 표현



부동 소수점 형식의 표현

- 4바이트를 사용하는 부동 소수점 형식



자료구조-문자자료의 표현

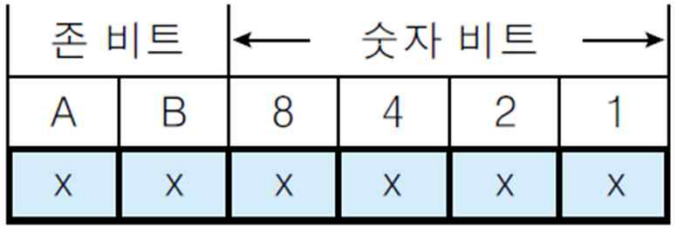
Robot Media Laboratory

- 문자에 대한 이진수 코드를 정의하여 사용
- 문자에 대한 이진수 코드표
  - BCD 코드
  - EBCDIC 코드
  - ASCII 코드

자료구조-BCD 코드

Robot Media Laboratory

- **6비트를 사용하여 문자 표현**
  - 상위 2비트 : 존 비트
  - 하위 4비트 : 2진수 비트
  - 존 비트와 2진수 비트를 조합하여 10진수 0~9와 영어 대문자, 특수 문자를 표현



존 비트		← 숫자 비트 →			
A	B	8	4	2	1
X	X	X	X	X	X

존 비트 AB의 값
- 00 : 0, 19(1010, 00011001)
- 01 : 문자 AI(00011001)
- 10 : 문자 R(00011001)
- 11 : 문자 S(00101001)

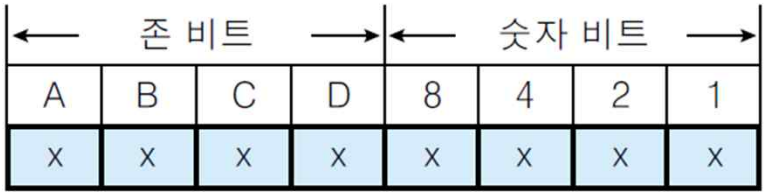
자료구조-EBCDIC 코드

Robot Media Laboratory

8비트를 사용하여 문자 표현

- 상위 4비트 : 존 비트
- 하위 4비트 : 2진수 비트
- 존 비트와 2진수 비트를 조합하여 10진수 0~9와 영어 대문자/소문자와 특수문자를 표현

EBCDIC 코드의 구성



존 비트 AB의 값
- 00 : 여분
- 01 : 특수 문자
- 10 : 영어 소문자
- 11 : 영어 대문자

존 비트 CD의 값
- 00 : 문자 AI(00011001)
- 01 : 문자 R(00011001)
- 10 : 문자 S(00101001)
- 11 : 09(00001001)

자료구조-EBCDIC 코드

Robot Media Laboratory

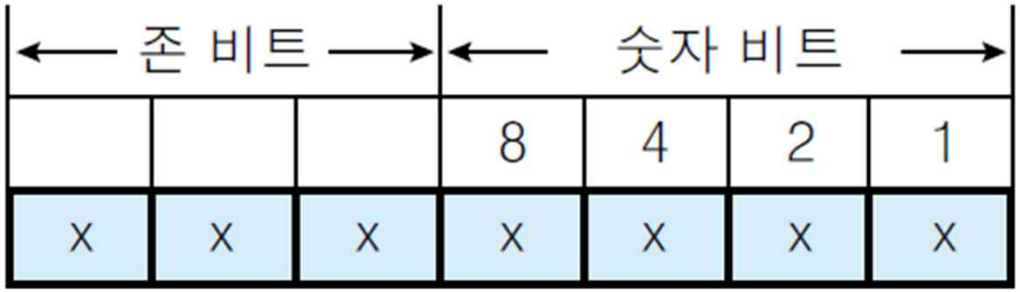
상위 / 하위	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE	DS		SP	&	-						{	}	₩(\)	0
0001	SOH	DC1	SOS				/		a	j	~		A	J		1
0010	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	ETX	TM							c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	ETB	UC					f	o	w		F	O	W	6
0111	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000	GE	CAN							h	q	y		H	Q	Y	8
1001	RLF	EM							i	r	z		I	R	Z	9
1010	SMM	CC	SM		¢	!		:								
1011	VT	CU1	CU2	CU3	.	$	,	#								
1100	FF	IFS		DC4	<	*	%	@								
1101	CR	IGS	ENQ	NAK	(	)	_	'								
1110	SO	IRS	ACK		+	;	>	=								
1111	SI	IUS	BEL	SUB	|	¬	?	"								

자료구조-ASCII 코드

Robot Media Laboratory

- **7비트를 사용하여 문자 표현**
  - 상위 3비트 : 존 비트
  - 하위 4비트 : 2진수 비트
  - 존 비트와 2진수 비트를 조합하여 10진수 0~9와 영어 대문자/소문자, 특수문자를 표현

- **ASCII 코드의 구성**



자료구조-ASCII 코드

Robot Media Laboratory

하위 \ 상위	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	″	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	END	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	′	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	₩(\)	l	|
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	·	>	N	∧	n	~
1111	SI	US	/	?	O	_	o	DEL

※ 코드의 의미

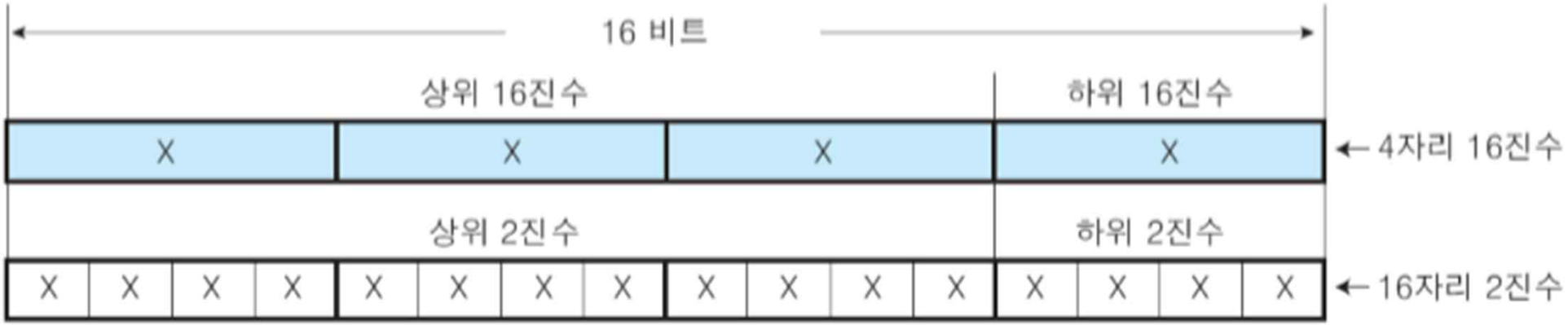
GS	Group Separator	RS	Record Separator	US	Unit Separator

ROBOT MEDIA LABORATORY

자료구조-유니 코드

- **16비트를 사용하여 문자 표현**
  - 16비트의 코드값을 4자리의 16진수로 표시

- **유니 코드의 구성**



자료구조-유니 코드

Robot Media Laboratory

상위16진수 / 하위16진수	000	001	002	003	004	005	006	007
0	NUL	DEL	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|

자료구조-유니 코드

D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	〉	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

- 영문자 A에 대한 유니 코드 ☞ 0041 ➔ 0000 0000 0100 0001

자료구조-유니 코드

Robot Media Laboratory

상위16진수 / 하위16진수	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7
0	가	감	갠	갰	갈	갹	걨	거
1	각	갑	갡	갱	갉	갥	갤	걱
2	갂	값	갢	갲	갊	갦	갴	걲
3	갃	갓	갣	갳	갋	갧	갵	걳
4	간	갔	갤	갴	갌	개	갶	건
5	갅	강	갥	갵	갍	객	갷	걵
6	갆	갖	갦	갶	갎	갞	갸	걶
7	갇	갗	갧	갷	갏	갟	갹	걷
8	갈	갘	갨	갸	감	갠	갰	걸
9	갉	같	갩	갹	갑	갡	갱	걹
A	갊	갚	갪	갺	값	갢	갲	걺
B	갋	갛	갫	갻	갓	갣	갳	걻
C	갌	개	갬	간	갔	갤	갴	걼
D	갍	객	갭	갽	강	갥	갵	걽
E	갎	갞	갮	갾	갖	갦	갶	걾
F	갏	갟	갯	갿	갗	갧	갷	걿

ROBOT MEDIA LABORATORY

자료구조-유니 코드

- 한글 '가'에 대한 유니 코드 ☞ AC0 0 ➔ 1010 1100 0000 0000

자료구조-논리자료

Robot Media Laboratory

- **논리값을 표현하기 위한 자료 형식**
- **논리값**
  - 참(True)와 거짓(False), 1과 0
- **1바이트를 사용하여 논리자료를 표현하는 방법**
  - 방법 1)
    - 참 : 최하위 비트를 1로 표시 **00000001**
    - 거짓 : 전체 비트를 0으로 표시. **00000000**
  - 방법2)
    - 참 : 전체 비트를 1로 표시. **11111111**
    - 거짓 : 전체 비트를 0으로 표시. **00000000**
  - 방법3)
    - 참 : 하나 이상의 비트를 1로 표시 **00000001 or 00000100 .....**
    - 거짓 : 전체 비트를 0으로 표시. **00000000**

소프트웨어 생명주기

Robot Media Laboratory

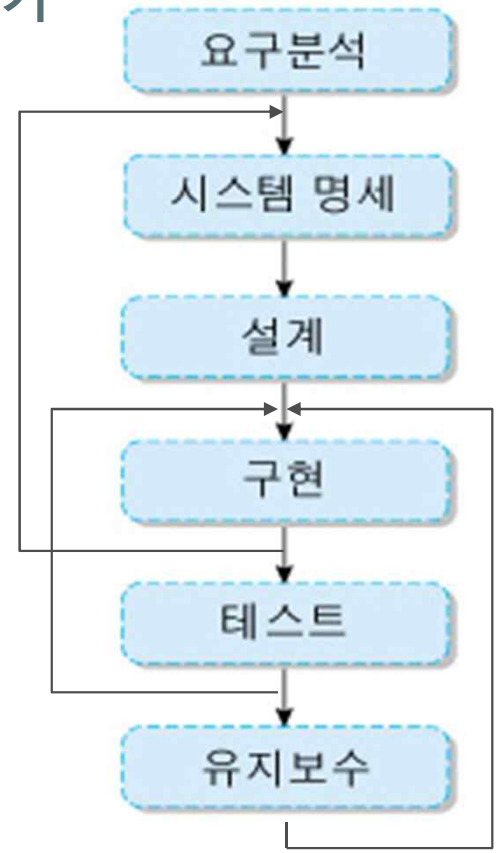
- **성공적인 소프트웨어 개발이란?**
  - 얼마나 정확하고 효율적으로 소프트웨어를 개발하고 사용 및 관리가 이루어지는가?
    - 개발할 소프트웨어에 대한 정확한 이해
    - 사용할 자료와 자료 간의 연산관계를 분석하여 최적의 자료구조 정의

- **소프트웨어 생명주기(Software Life Cycle)**
  - 소프트웨어를 체계적으로 개발하고 관리하기 위해서 개발 과정을 단계별로 나누어 구분한 것
  - 일반적으로 6단계로 구분

소프트웨어 생명주기

Robot Media Laboratory

일반적인 소프트웨어의 생명주기



소프트웨어 생명주기-요구분석단계

Robot Media Laboratory

- 문제 분석 단계
- 개발할 소프트웨어의 기능과 제약조건, 목표 등을 소프트웨어 사용자와 함께 명확히 정의하는 단계
- 개발할 소프트웨어의 성격을 정확히 이해하고 개발 방법과 필요한 개발 자원 및 예산 측정
- 요구명세서 작성

소프트웨어 생명주기-시스템 명세

Robot Media Laboratory

- 시스템이 무엇을 수행해야 하는가를 정의하는 단계
- 입력 자료, 처리 내용, 생성되는 출력이 무엇인지를 정의
- 시스템 기능 명세서 작성

ROBOT MEDIA LABORATORY

소프트웨어 생명주기-설계단계

Robot Media Laboratory

- **시스템 명세 단계에서 정의한 기능을 실제로 수행하기 위한 방법을 논리적으로 결정하는 단계**

- **시스템 구조 설계**
  - 시스템을 구성하는 내부 프로그램이나 모듈 간의 관계와 구조 설계

- **프로그램 설계**
  - 프로그램 내의 각 모듈에서의 처리 절차나 알고리즘을 설계

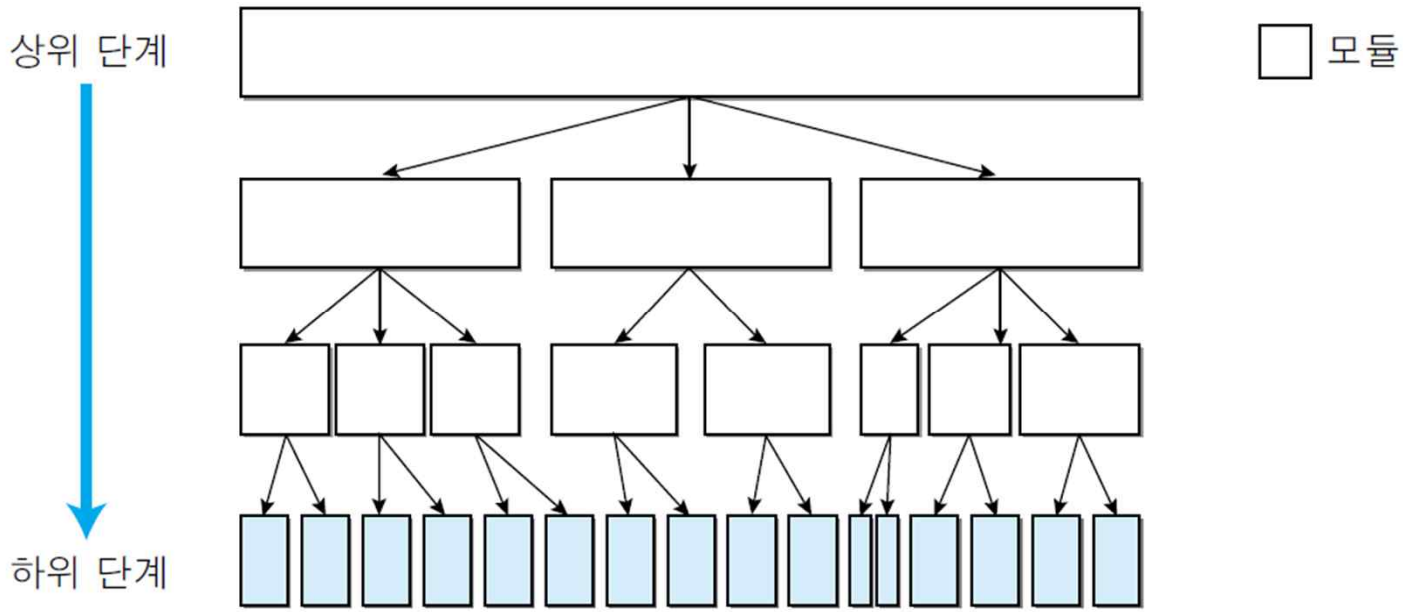
- **사용자 인터페이스 설계**
  - 사용자가 시스템을 사용하기 위해 보여지는 부분 설계

소프트웨어 생명주기-설계단계

Robot Media Laboratory

설계 방법

- 하향식 설계 방법
  - 상위단계에서 하위단계로 설계해가면서 점차 구체적으로 설계하는 방법
  - 분할 정복 방식의 설계 방법

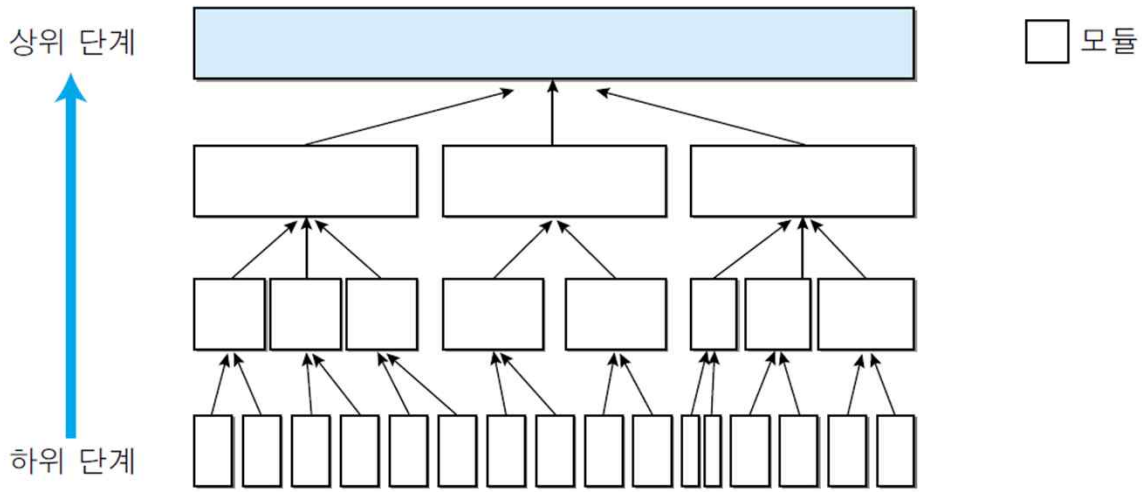


소프트웨어 생명주기-설계단계

Robot Media Laboratory

설계 방법

- 상향식 설계 방법
  - 하위단계의 작은 단위의 문제를 먼저 해결하고 이를 이용하여 상위단계의 큰 단위의 문제를 해결하는 방법



- 객체지향 설계 방법
  - 하위단위의 문제해결 도구를 객체로 만들어 재사용하는 방법으로 전체 문제를 해결하는 방법

소프트웨어 생명주기-구현 단계

Robot Media Laboratory

- **설계 단계에서 논리적으로 결정한 문제 해결 방법(알고리즘)을 프로그래밍언어를 사용하여 실제 프로그램을 작성하는 단계**

- **프로그래밍 기법**
  - 구조화 프로그래밍
    - 지정문과 조건문, 반복문만을 사용하여 프로그램을 작성
    - 순차구조, 선택구조, 반복구조의 세가지 제어구조로 표현
    - 구조가 명확하여 정확성 검증과 테스트 및 유지보수 용이
  - 모듈러 프로그래밍
    - 프로그램을 여러 개의 작은 모듈로 나누어 계층 관계로 구성하는 프로그래밍 기법
    - 모듈별로 개발과 테스트 및 유지보수 가능
    - 모듈의 재사용 가능

소프트웨어 생명주기-테스트 단계

Robot Media Laboratory

- 개발한 시스템이 요구사항을 만족하는지, 실행결과가 예상한 결과와 정확하게 맞는지를 검사하고 평가하는 일련의 과정
- 숨어있는 오류를 최대한 찾아내어 시스템의 완성도를 높이는 단계

- **1단계 - 단위 테스트(Unit Test)**
  - 시스템의 최소 구성요소가 되는 모듈에 대해서 개별적으로 시행

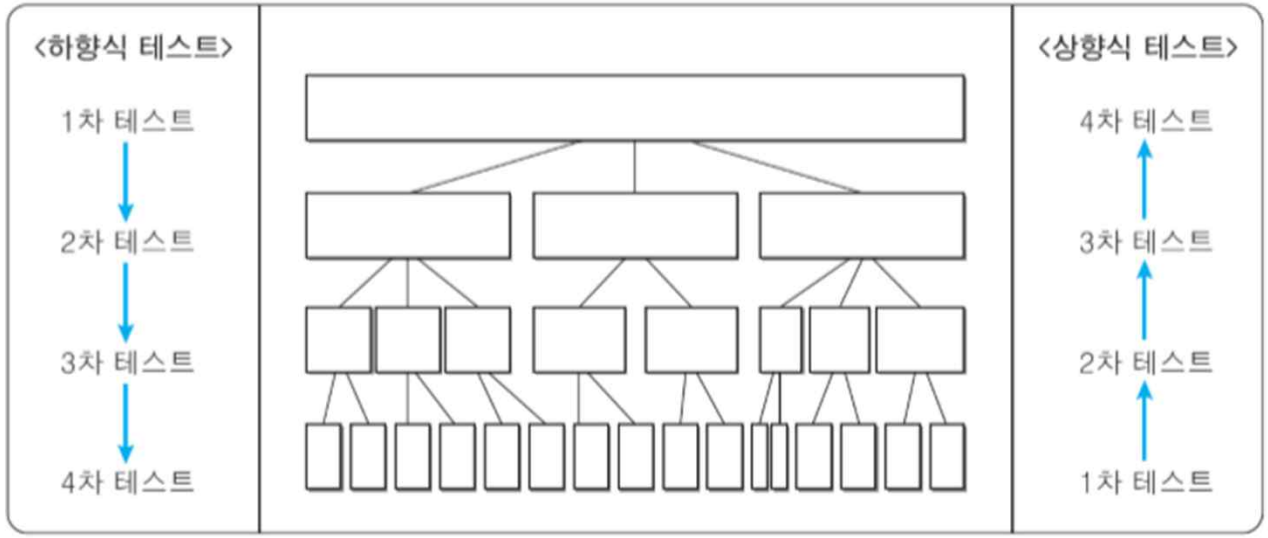
- **2단계 – 통합테스트(Integration test)**
  - 단위 테스트를 통과한 모듈을 연결하여 전체 시스템으로 완성하여 통합적으로 시행하는 테스트
  - 구성요소 연결을 점진적으로 확장하면서 테스트 시행
    - 하향식 테스트
    - 상향식 테스트

ROBOT MEDIA LABORATORY

소프트웨어 생명주기-테스트 단계

Robot Media Laboratory

- **2단계 – 통합테스트(Integration test)**
  - 하향식/상향식 점진적 테스트



- **3단계 – 인수 테스트**
  - 완성된 시스템을 인수하기 위해서 실제 자료를 사용한 최종 테스트

소프트웨어 생명주기-유지보수 단계

Robot Media Laboratory

- **시스템이 인수되고 설치된 후 일어나는 모든 활동**
  - 소프트웨어 생명주기에서 가장 긴 기간
- **유지보수의 유형**
  - 수정형 유지보수
    - 사용 중에 발견한 프로그램의 오류 수정 작업
  - 적응형 유지보수
    - 시스템과 관련한 환경적 변화에 적응하기 위한 재조정 작업
  - 완전형 유지보수
    - 시스템의 성능을 향상시키기 위한 개선 작업
  - 예방형 유지보수
    - 앞으로 발생할지 모를 변경 사항을 수용하기 위한 대비 작업

소프트웨어 생명주기
Robot Media Laboratory

개발된 소프트웨어 품질 평가

- 정확성
  - 요구되는 기능들을 정확하게 수행하는 정도를 평가
- 유지 보수성
  - 효율적 유지 보수의 정도를 평가
- 무결성
  - 바이러스 등의 외부 공격에 대한 보안성 평가
- 사용성
  - 사용자가 쉽고 편리하게 사용할 수 있는가에 대한 평가

뇌의 추상화 기능

- 기억할 대상의 구별되는 특징만을 단순화하여 기억하는 기능

- **크고 복잡한 문제를 단순화시켜 쉽게 해결하기 위한 방법**
- **자료 추상화(Data Abstraction)**
  - 처리할 자료, 연산, 자료형에 대한 추상화 표현
  - 자료 : 프로그램의 처리 대상이 되는 모든 것을 의미

  - 연산
    - 어떤 일을 처리하는 과정. 연산자에 의해 수행
    - 예) 더하기 연산은 +연산자에 의해 수행

  - 자료형
    - 처리할 자료의 집합과 자료에 대해 수행할 연산자의 집합
    - 예) 정수 자료형

  자료 : 정수의 집합. {…, -1, 0, 1, …}
  연산자 : 정수에 대한 연산자 집합. {+, -, x, ÷, mod}

추상화와 구체화

- 자료와 연산에 있어서의 추상화와 구체화의 관계

	자료	연산
추상화	추상 자료형	알고리즘 정의
구체화	자료형	프로그램 구현

- 추상데이터타입은 자료구조를 프로그램으로 구현할때 데이터를 저장할 구조를 생성,저장 데이터 처리하는 연산을 정의 하기위한 관계를 정형화 시킨 개념

(추상 데이터타입) (자료구조)

데이터 + 관련연산 --[구현]--> 데이터 + 구체화된 관련연산

- 자료구조는 추상데이터타입을 실제프로그램으로 구현한것

알고리즘

- 문제해결방법을 추상화하여 단계적 절차를 논리적으로 기술해 놓은 명세서

알고리즘의 조건

- 입력(input) : 알고리즘 수행에 필요한 자료가 외부에서 입력으로 제공될 수 있어야 한다.
- 출력(output) : 알고리즘 수행 후 하나 이상의 결과를 출력해야 한다.
- 명확성(definiteness) : 수행할 작업의 내용과 순서를 나타내는 알고리즘의 명령어들은 명확하게 명세되어야 한다.
- 유한성(finiteness) : 알고리즘은 수행 뒤에 반드시 종료되어야 한다.
- 효과성(effectiveness) : 알고리즘의 모든 명령어들은 기본적이며 실행이 가능해야 한다.

알고리즘
Robot Media Laboratory

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

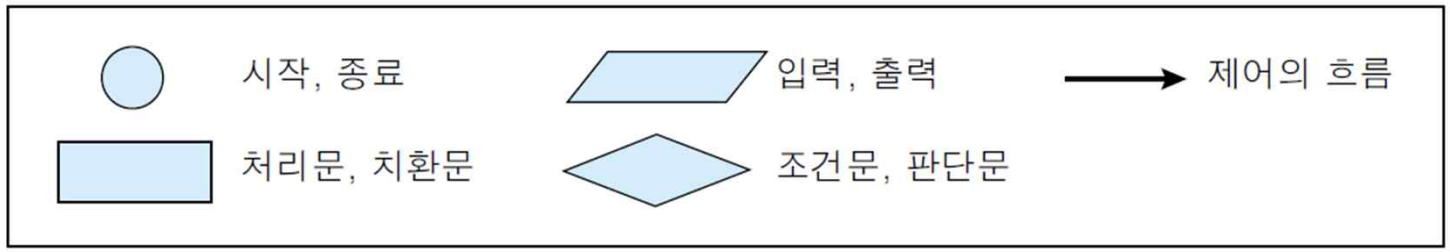
연산

알고리즘의 표현 방법

- 자연어를 이용한 서술적 표현 방법
- 순서도(Flow chart)를 이용한 도식화 표현 방법
- 프로그래밍 언어를 이용한 구체화 방법
- 가상코드(Pseudo-code)를 이용한 추상화 방법

순서도를 이용한 알고리즘의 표현

- 순서도에서 사용하는 기호



- 장점
  - 알고리즘의 흐름 파악이 용이함
- 단점
  - 복잡한 알고리즘의 표현이 어려움

가상코드를 이용한 알고리즘의 표현

- 가상코드, 즉 알고리즘 기술언어(ADL, Algorithm Description Language)를 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

가상코드의 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

변수 ← 값 ;

  - 사용형식

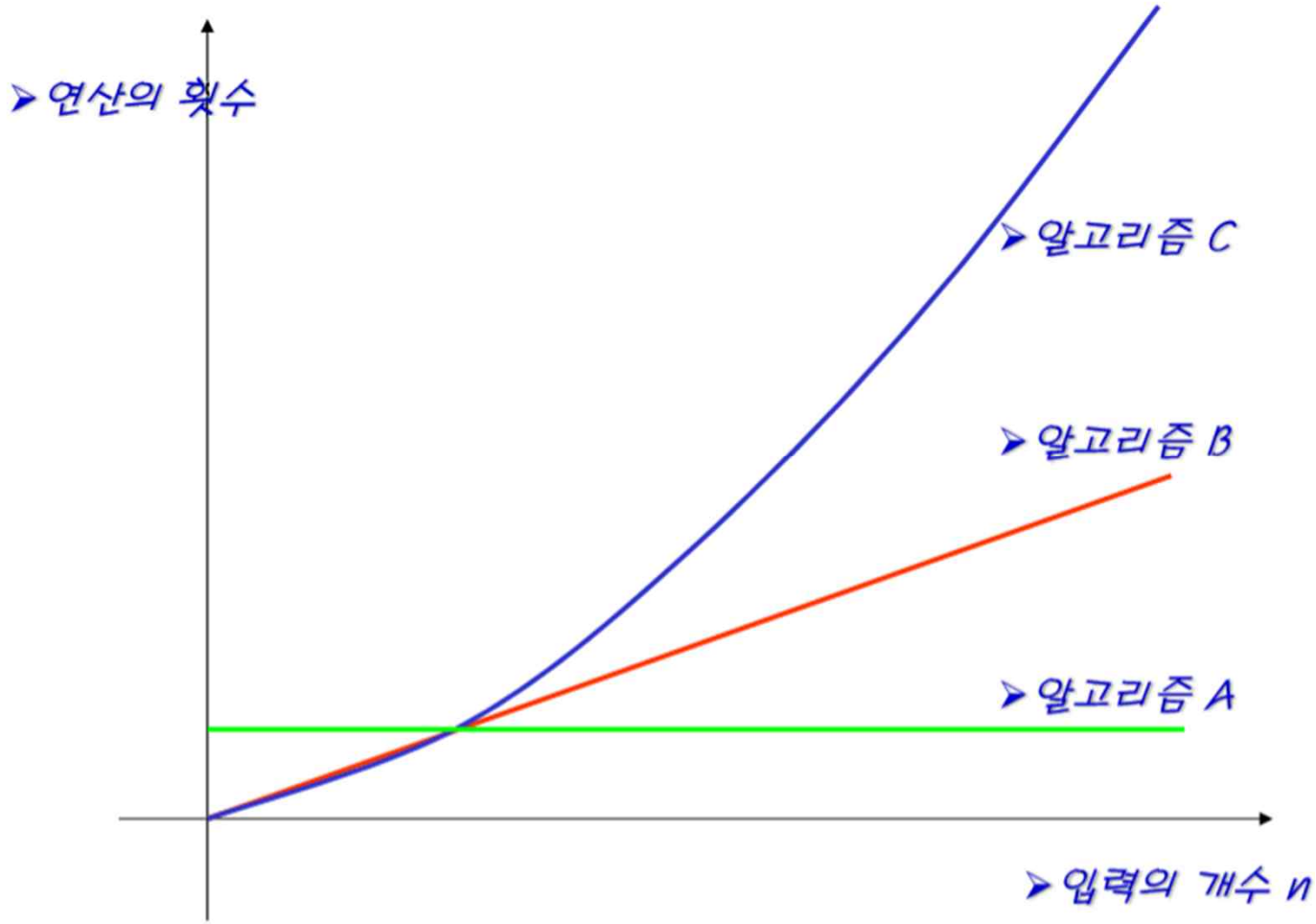
  - 지정연산자(←)의 오른쪽에 있는 값(또는 식의 계산 결과 값이나 변수의 값)을 지정연산자(←)의 왼쪽에 있는 변수에 저장

- **알고리즘 성능 분석 방법**
- **공간 복잡도**
  - 알고리즘을 프로그램으로 실행하여 완료하기까지 필요한 총 저장 공간의 양
  - 공간 복잡도 = 고정 공간 + 가변 공간

- **시간 복잡도**
  - 알고리즘을 프로그램으로 실행하여 완료하기까지의 총 소요시간
  - 시간 복잡도 = 컴파일 시간 + 실행 시간
    - 컴파일 시간 : 프로그램마다 거의 고정적인 시간 소요
    - 실행 시간 : 컴퓨터의 성능에 따라 달라질 수 있으므로 실제 실행시간 보다는 명령문의 실행 빈도수에 따라 계산
  - 실행 빈도수의 계산
    - 지정문, 조건문, 반복문 내의 제어문과 반환문은 실행시간 차이가 거의 없으므로 하나의 단위시간을 갖는 기본 명령문으로 취급

알고리즘

Robot Media Laboratory

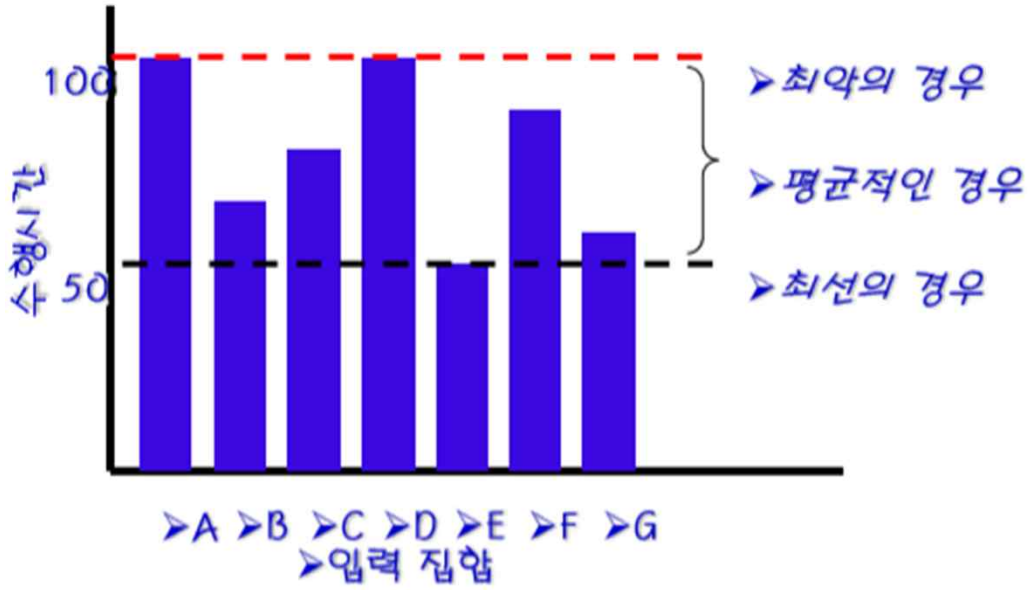


ROBOT MEDIA LABORATORY

알고리즘
Robot Media Laboratory

- 알고리즘의 수행시간은 입력 자료 집합에 따라 다르다

- Best case: 수행시간이 가장 빠른 경우
- Average case: 수행시간이 평균적인 경우
- Worst case: 수행시간이 가장 늦은 경우



- **탐색 상황 하나: 운이 좋은 경우**
  - 배열의 맨 앞에서 대상을 찾는 경우
  - 만족스러운 상황이므로 성능평가의 주 관심이 아니다!
- **최상의 경우(best case)**

- **탐색 상황 둘: 운이 좋지 않은 경우**
  - 배열의 끝에서 찾거나 대상이 저장되지 않은 경우
  - 만족스럽지 못한 상황이므로 성능평가의 주 관심이다!
- **최악의 경우(worst case).**

- **평균이란 가장 현실적인 경우에 해당**
  - 일반적으로 등장하는 상황에 대한 경우의 수
  - 최상의 경우와 달리 알고리즘 평가에 도움
  - 하지만 계산하기가 어렵다
  - 객관적 평가가 쉽지 않다

- **평균적인 경우의 복잡도 계산이 어려운 이유**
  - 평균적인 경우 의 연출이 어렵다.
  - 평균적인 경우 임을 증명하기 어렵다.
  - 평균적인 경우 는 상황에 따라 달라진다.
  - 반면 최악의 경우는 늘 동일하다.

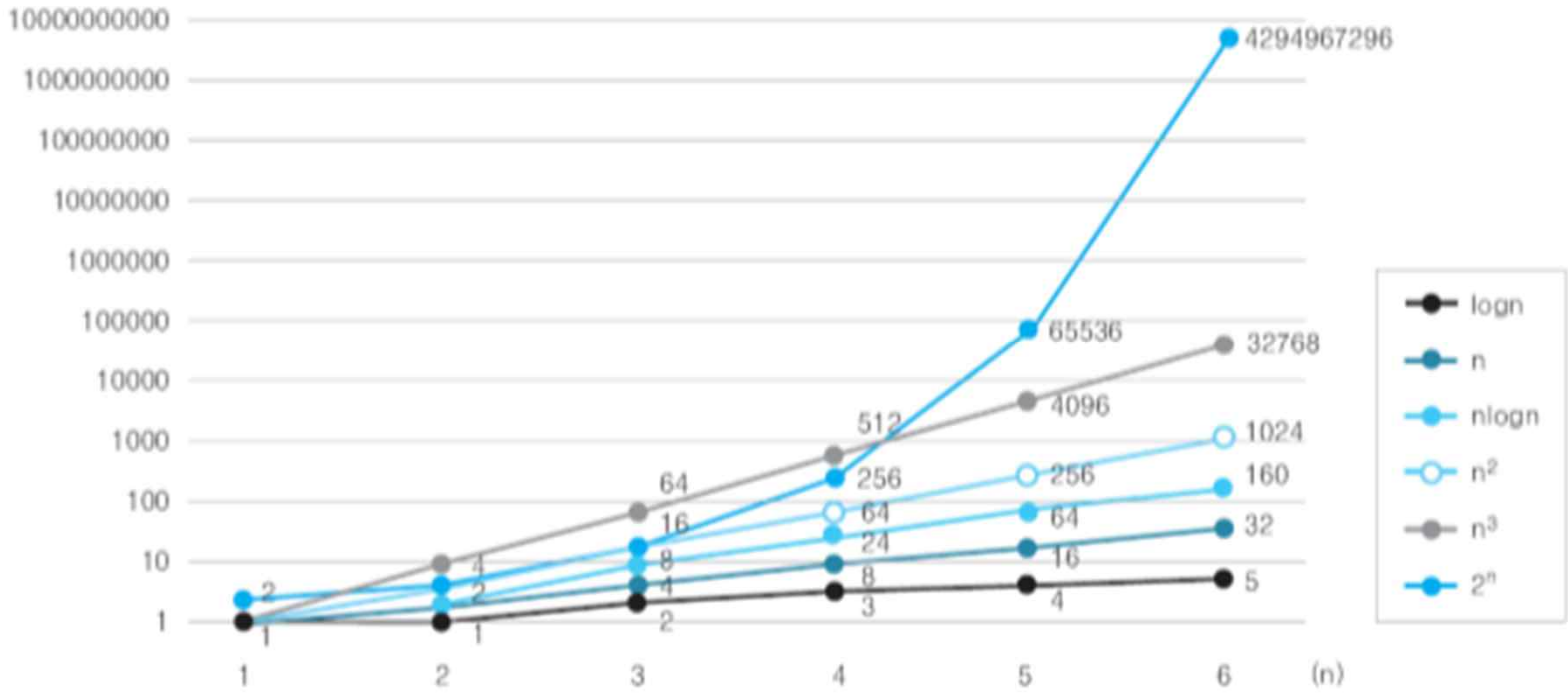
- **알고리즘 성능 분석 방법**
- **시간 복잡도 표기법**
  - 빅-오(Big-Oh) 표기법 사용
  - 빅-오(Big-Oh) 표기법 순서
    ① 실행 빈도수를 구하여 **실행시간 함수** 찾기
    ② 실행시간 함수의 값에 가장 큰 영향을 주는 **n에 대한 항을 선택**하여
    ③ **계수는 생략**하고 ***O*** (Big-Oh)의 오른쪽 괄호 안에 표시

  - 피보나치 수열의 시간 복잡도 = *O* (n)
    ① 실행시간 함수 : **4n+2**
    ② n에 대한 항을 선택 : **4n**
    ③ 계수 4는 생략하고 ***O*** (Big-Oh)의 오른쪽 괄호 안에 표시 : ***O* (n)**

- 각 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교

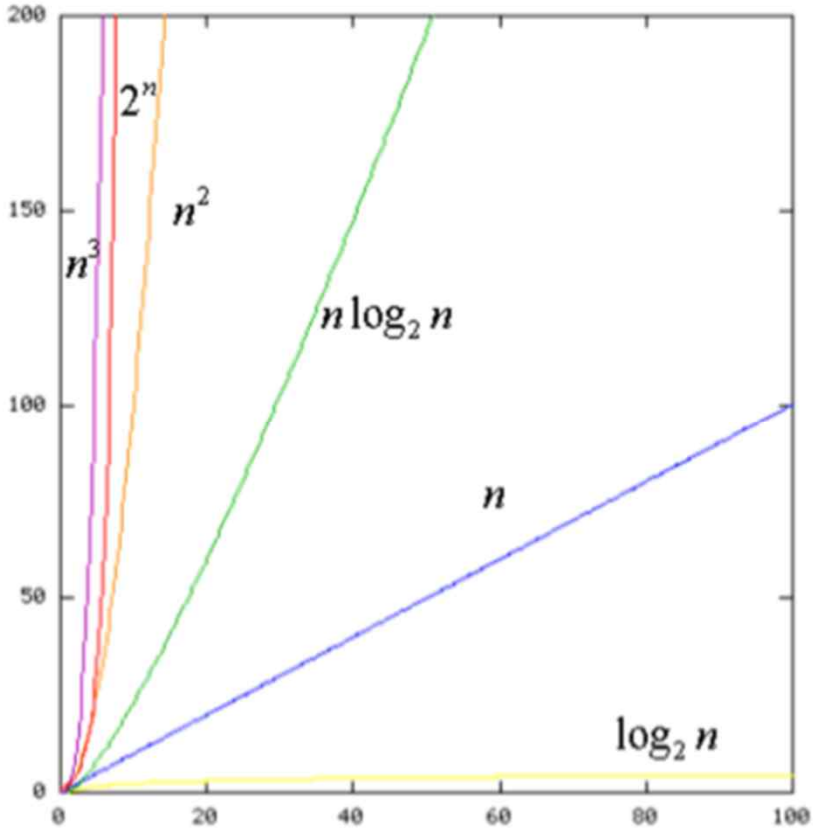
$\log n$	<	n	<	$n\log n$	<	n^2	<	n^3	<	2^n
0		1		0		1		1		2
1		2		2		4		8		4
2		4		8		16		64		16
3		8		24		64		512		256
4		16		64		256		4096		65536
5		32		160		1024		32768		4294967296

- 각 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교



- 각 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교

O(1) : 상수형
O(logn) : 로그형
O(n) : 선형
O(nlogn) : 로그선형
O(n^2) : 2차형
O(n^3) : 3차형
O(n^k) : k차형
O(2^n) : 지수형
O(n!) : 팩토리얼형



- 빅O(n)표기법의 예제

클래스

Robot Media Laboratory

- 객체지향
- 과학, 군사적 모의실험(simulation)을 위해 컴퓨터를 이용한 가상세계를 구현하려는 노력으로 부터 객체지향이론이 시작
- 기존의 프로그래밍언어와 크게 다르지 않다.
- 코드의 재사용성이 높다
- 코드의 관리가 쉬워졌다.
- 신뢰성이 높은 프로그램의 개발을 가능하게 한다.

클래스
Robot Media Laboratory

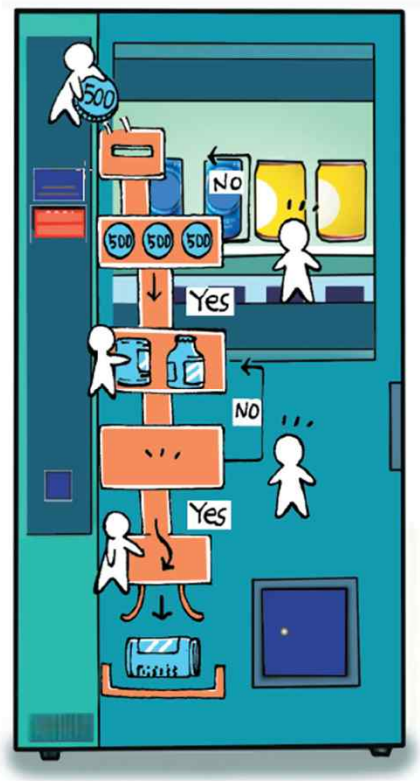
- 객체지향
- 프로그램을 구성하는 요소는 객체
- 객체가 상호작용하도록 프로그래밍
- 클래스, 객체 인스턴스를 알아야 한다.
- 객체지향의 4성질 캡슐화, 상속성, 다형성, 추상화

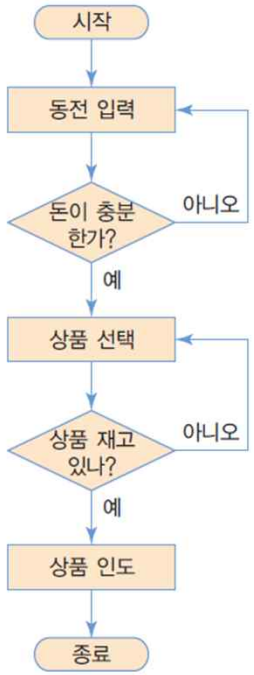
클래스
Robot Media Laboratory

- **소프트웨어 생산성 향상**
  - 소프트웨어의 생명 주기 단축 문제 해결 필요
  - 기 작성된 코드의 재사용 필요
- **실세계에 대한 쉬운 모델링**
  - 과거의 소프트웨어
    - 수학 계산이나 통계 처리에 편리한 절차 지향 언어가 적합
  - 현대의 소프트웨어
    - 물체 혹은 객체의 상호 작용에 대한 묘사가 필요
    - 실세계는 객체로 구성된 세계
    - 객체를 중심으로 하는 객체 지향 언어 적합

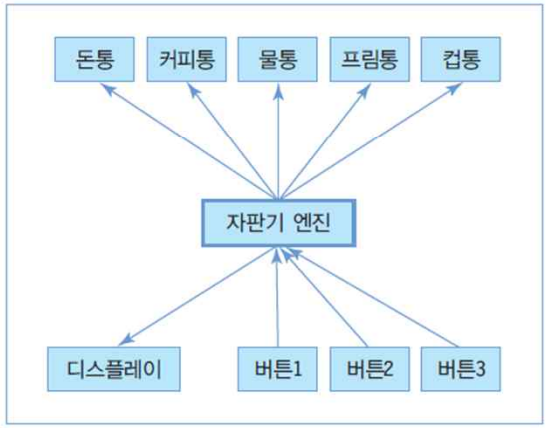
클래스

Robot Media Laboratory





(a) 절차 지향적 프로그래밍으로 구현할 때의 흐름도



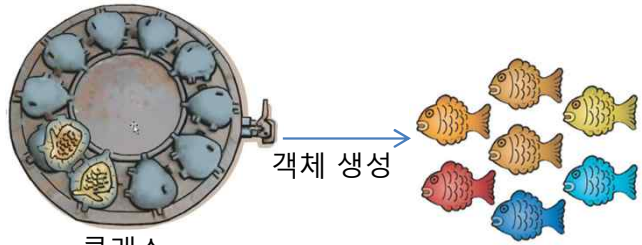
(b) 객체 지향적 프로그래밍으로 구현할 때의 객체 관계도

- 실행하고자 하는 절차대로 일련의 명령어 나열.
- 흐름도를 설계하고 흐름도에 따라 프로그램 작성

- 객체들을 정의하고, 객체들의 상호 관계, 상호 작용으로 구현

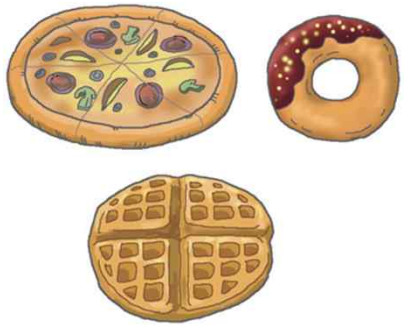
클래스
Robot Media Laboratory

- **캡슐화(Encapsulation)**
  - 데이터를 캡슐로 싸서 외부의 접근으로 부터 보호
  - Python에서 클래스class 로 캡슐 표현
- **클래스와 객체**
  - 클래스 – 객체를 만드는 틀
  - 객체 – 클래스라는 틀에서 생겨난 실체
  - 객체(object), 실체(instance)는 같은 뜻



클래스
(객체를 정의하는 틀)

객체들 - 실체



원 객체들(실체)

클래스

Robot Media Laboratory

- 캡슐화: 중요한 데이터를 보존,보호하는 것
- 일반적으로 연관 있는 변수와 함수를 클래스로 묶는 작업
- 클래스 만드는 작업과 비슷하다고 여길 수도 있다.
- 은닉성이란게 있어서 클래스에 담는 내용 중 중요한 데이터나 기능을 외부에서 접근하지 못하게 할 수 있다.
- 만일의 상황(타인이 외부에서 조작)을 대비해서 외부에서 특정 속성이나 메서드를 시용자가 사용할 수 없도록 숨겨놓은 것

클래스
Robot Media Laboratory

- 캡슐화
- 객체의 필드(속성), 메소드를 하나로 묶고, 실제 구현 내용을 외부에 감추는 것을 말한다.

- 외부 객체는 객체 내부의 구조를 얻지 못하며 객체가 노출해서 제공하는 필드와 메소드만 이용할 수 있다.

- 필드와 메소드를 캡슐화 하여 보호하는 이유는 외부의 잘못된 사용으로 인해 객체가 손상되지 않도록 하는데 있다.

- 캡슐화 된 멤버를 노출시킬 것인지 숨길 것인지를 결정하기 위해 접근 제한자(Access Modifier)를 사용한다.

클래스
Robot Media Laboratory

- 상속
- 자식(하위,파생) 클래스가 부모(상위) 클래스의 멤버를 물려받는 것
- 자식이 부모를 선택해서 물려 받는 것
- 상속 대상 : 부모의 필드와 메소드

클래스

Robot Media Laboratory

- 부모 클래스를 재사용해서 자식 클래스를 빨리 개발할 수 있다.
- 반복된 코드의 중복을 줄여준다.

- 유지 보수의 편리성을 제공해 준다.

(부모 클래스를 한 번만 수정함으로써 자식클래스를 수정할 필요가 없음)

- 객체의 다형성을 구현할 수 있다.

클래스
Robot Media Laboratory

- 다형성
- 같은 타입이지만 실행 결과가 다양한 객체를 대입(이용)할 수 있는 성질
- 객체를 부품화시킬 수 있습니다.
- 부모클레스에서 물려받은 가상 함수를 자식 클래스 내에서 오버라이딩 되어 사용되는 것

다형성(Polymorphism)

- 하나의 기능이 경우에 따라 다르게 보이거나 다르게 작동하는 현상
- 연산자 중복, 함수 중복, 함수 재정의(overriding)

- 추상화


양평역
영등포구청역
오목교역
문래역
도림천역
영등포역
양천구청역
신도림역
46
구로역
구일역
구로구
대림역


양평역
영등포구
오목교역
영등포구청역
문래역
도림천역
영등포역
신도림역
46
구로역
구로구
대림역


신도림
구일
구로
남구로
가리봉

클래스

Robot Media Laboratory

- 추상화
- 공통의 속성이나 기능을 묶어 이름을 붙이는 것
- 객체 지향적 관점에서 클래스를 정의하는 것
- 물고기, 사자, 토끼, 뱀등을 동물 또는 생물이라고 묶는 것을 추상화라고 한다.
- 객체지향에서 추상화라는 개념은 '객체에서 공통된 속성과 행위를 추출하는 것'을 의미한다.

클래스

Robot Media Laboratory

- 객체
- 자신의 속성을 갖고있고 다른 것과 식별이 가능한 것.
- 배열과 비슷하다
- 타언어에서 연관배열, 맵, 딕셔너리등의 데이터 타입과 같다 볼수있다.

클래스

Robot Media Laboratory

- 객체
- 데이터를 가지고 있습니다. - 데이터는 객체의 상태를 기술하는 정보를 저장합니다.
- 행위의 집합을 가지고 있습니다. - 이 행위들은 메세지를 받았을 때 객체가 어떻게 해야하는지 알고 있는 것 입니다.
- 개체를 구분하는 아이덴티티를 가지고 있습니다. - 어떠한 객체를 다른 객체와 구분하는 것을 가능케 합니다.

클래스

Robot Media Laboratory

- 객체
- 사전적 정의로 실제 존재하는 것
- 객체지향 이론에서는 사물과 같은 유형적인 것 뿐만 아니라, 개념이나 논리같은 무형적인 것들도 객체로 간주한다.
- 프로그래밍에서의 객체는 클래스에 정의된 내용대로 메모리에 생성된 것

클래스
Robot Media Laboratory

- 붕어빵기계: 클래스
- 붕어빵: 객체
- 붕어빵속 앙금: 필드

클래스

Robot Media Laboratory

- 클래스
- 객체를 정의 해놓은 것
- 객체의 설계도,틀
- 객체를 생성하는데 사용
- 객체는 클래스에 정의된 대로 생성

class 클래스명:

- **clss 클래스명:**

```
def__init__(self):                          <=생성자
        self.인스턴스변수1=0
def setdata(self,입력매개변수1,...):          <=초기화(생성자로 대체가능)
        self.인스턴스변수1=입력매개변수1
        ...
def 인스턴스 메소드(self,...):                <=내부의 메소드 생성가능
        ...
        return 반환값
...
```

class 자동차:

- **자동차 클래스가 생성되었다고 해서 자동차가 만들어진 것은 아니다.**

```
class Test:

t1 = Test()
t2 = Test()
```

- 생성자를 이용하여 메모리에 객체를 만들라는 명령.
- 메모리에 만들어진 객체를 인스턴스(instance)라고도 한다.
- 객체를 참조하는 변수 t1 , t2
- Test라는 객체가 2개가 만들어지고, 각각의 객체를 참조하는 t1과 t2 변수가 선언

클래스
Robot Media Laboratory

● 객체 ≒ 인스턴스

객체는 인스턴스를 포함하는 일반적인 의미

● 인스턴스화

클래스로부터 인스턴스를 생성하는것

클래스

Robot Media Laboratory

```
class Animal

cat = Animal()
```

클래스에 의해서 만들어진 객체를 인스턴스.

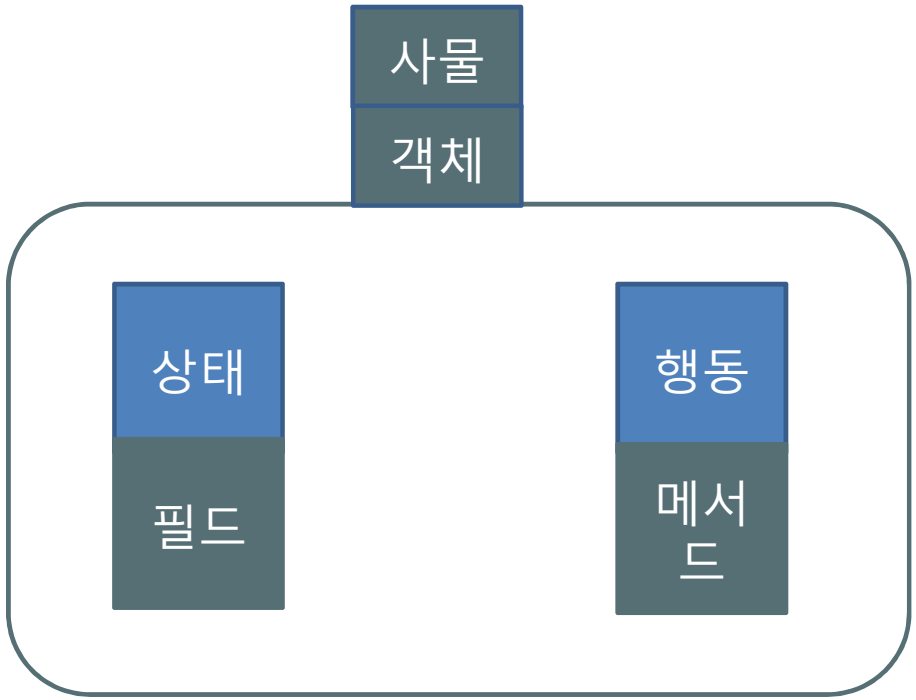
cat은 객체, cat이라는 객체는 Animal의 인스턴스(instance)

인스턴스 = 특정 객체(cat)가 어떤 클래스(Animal)의 객체인지를 관계 위주로 설명할 때 사용
즉, cat은 인스턴스" 보다는 cat은 객체
cat은 Animal의 객체 보다는 cat은 Animal의 인스턴스

ROBOT MEDIA LABORATORY

클래스

Robot Media Laboratory



대상의 속성= 필드

Test 클래스 인스턴스화

```
t1 = Test()
t2 = Test()
```

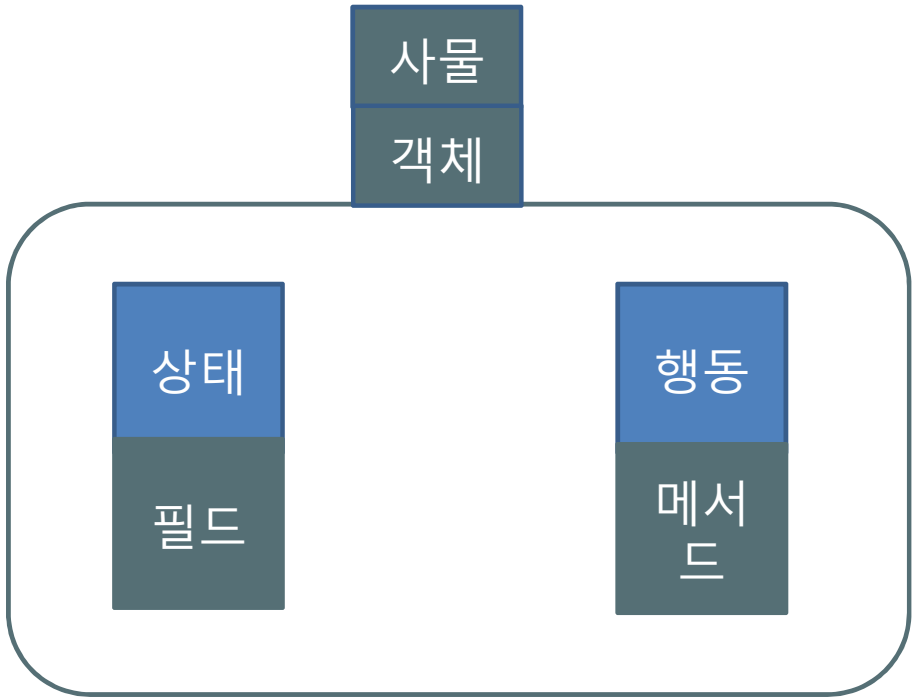
객체별로 name과 number라는 속성을 가진다.

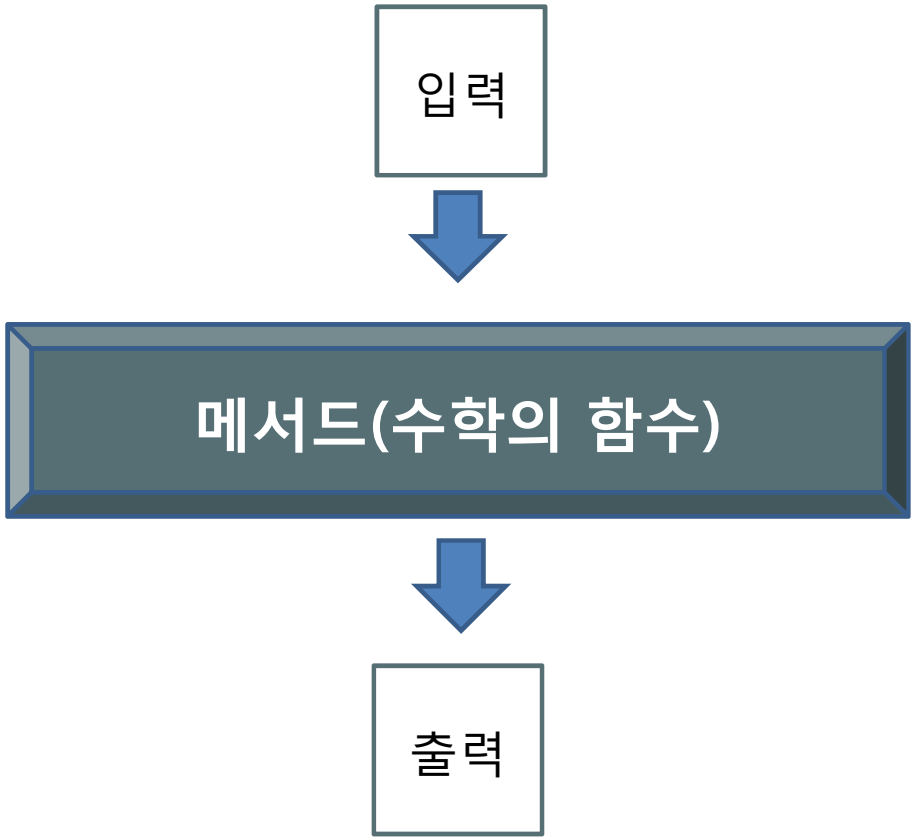
```
t1.name = "소방차"
t1.number = 1234
t2.name = "구급차"
t2.number = 1004

print(t1.name)
print(t1.number)
name = t2.name
Print(name)
```

클래스

Robot Media Laboratory




입력
메서드(수학의 함수)
출력

입력값: 매개변수,인자　　결과값: 리턴값

매개변수(Parameter): 전달된 인자를 받아들이는 변수를 의미

인자(Argument):어떤 함수를 호출시에 전달되는 값을 의미

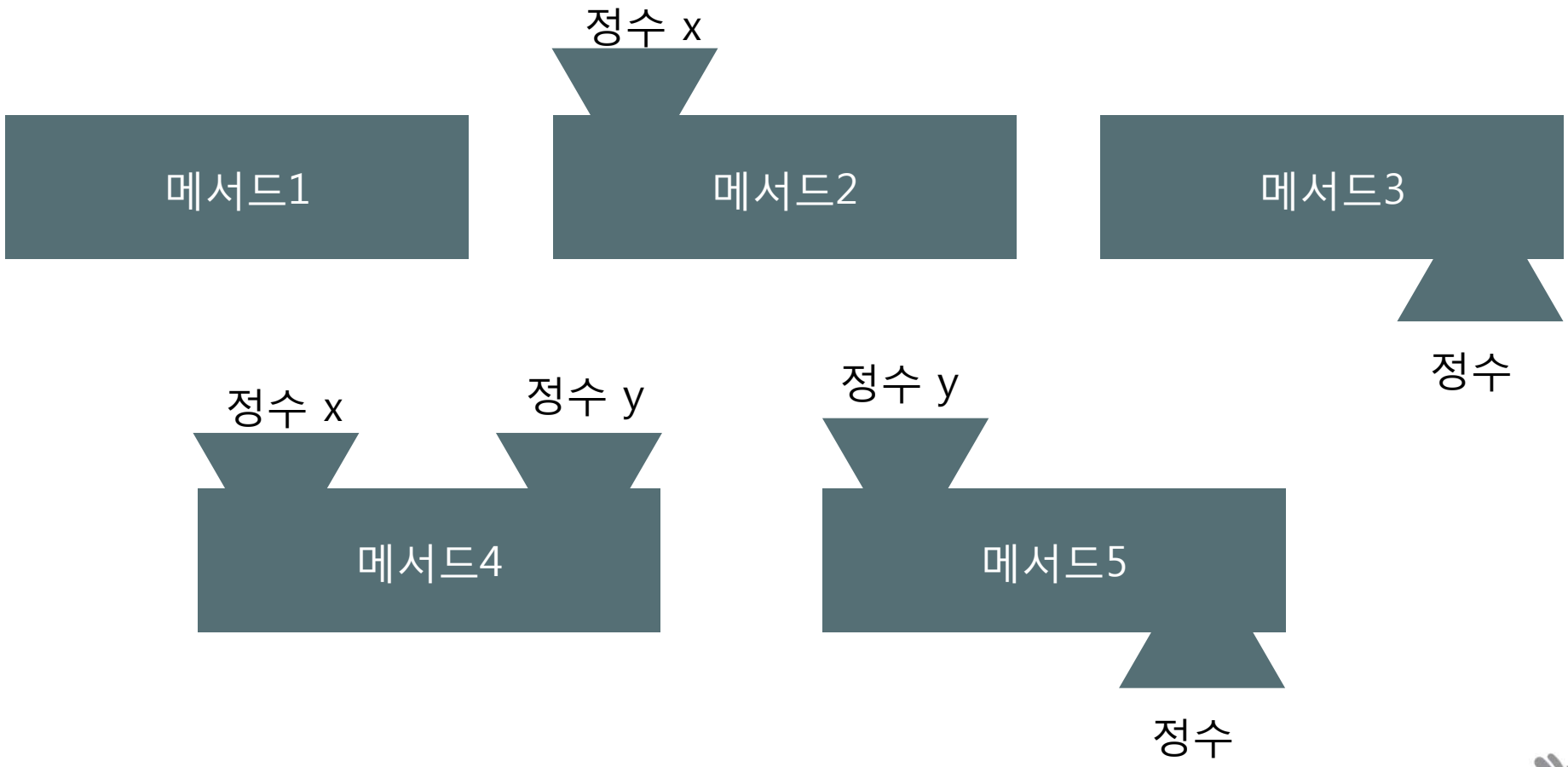
메소드란 클래스가 가지고 있는 기능

클래스 안에 선언

def 메서드이름(매개변수들):
    실행문

메서드

Robot Media Laboratory



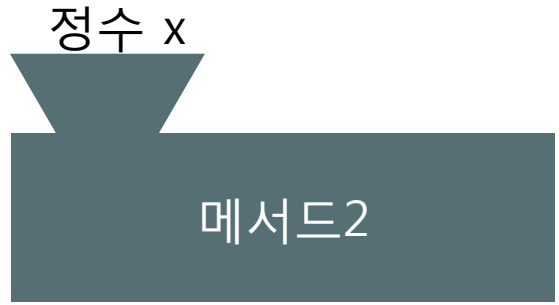
메서드1

매개변수도 없고 리턴하는 것도 없는 형태의 메소드

```
def method1(self):
        print("method1이 실행됩니다.")
```

메서드

Robot Media Laboratory



정수를 받아들인 후, 리턴하지 않는 메소드

```
def method2(self,x):
        print(x + " 를 이용하는 method2입니다.")
```

메서드3

정수

아무것도 받아들이지 않고, 정수를 반환하는 메소드

```
def method3(self):
        print("method3이 실행됩니다.")
        return 10
```

*리턴하는 값 앞에 return 이라는 키워드를 사용한다.

메서드

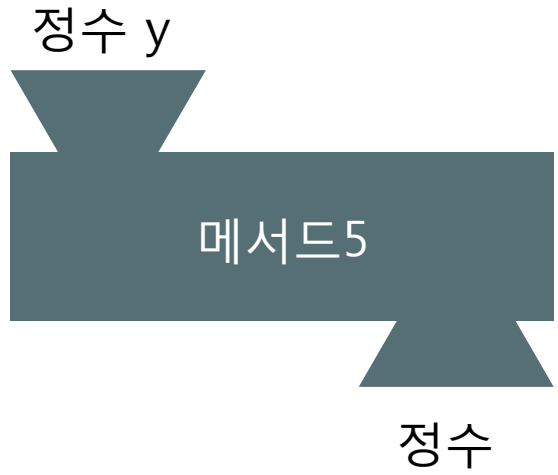
Robot Media Laboratory

정수를 2개 매개변수로 받고, 아무것도 반환하지 않는 메소드

```
def method4(self,x, y):
 print (x + "," + y + " 를 이용하는 method4입니다.")
```

메서드

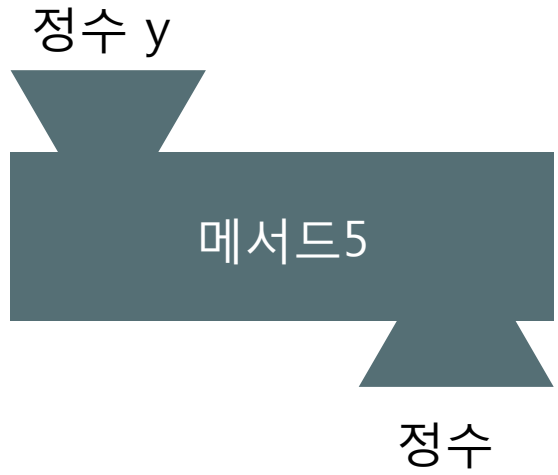
Robot Media Laboratory



정수형 매개변수 1개를 받고 정수 1개를 반환하는 메소드를 작성하시오

메서드

Robot Media Laboratory

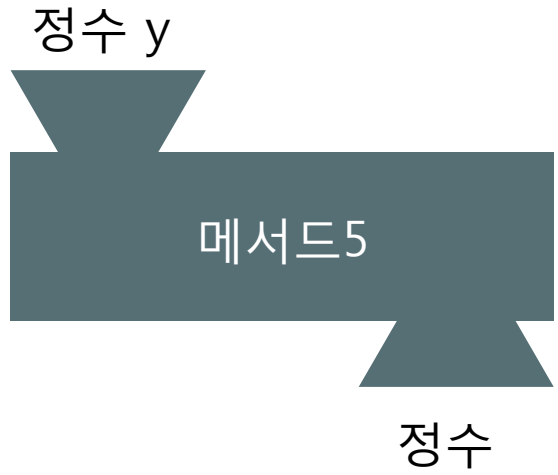


정수를 한개 받아들인 후, 정수를 반환하는 메소드

```
def method5(self,y):
        print(y + " 를 이용하는 method5입니다.")
    return 5
```

메서드

Robot Media Laboratory



정수를 한개 받아들인 후, 정수를 반환하는 메소드

```
def method5(self,y):
        print(y + " 를 이용하는 method5입니다.")
    return 5
```

자료 구조
-리스트-

리스트(List)

- 자료를 나열한 목록

동창 이름 리스트	좋아하는 음식 리스트	오늘의 할 일 리스트
상원	김치찌개	운동
승희	닭볶음탕	자료구조 스터디
수영	된장찌개	과제 제출
철이	잡채	동아리 공연 연습
…	…	…

자료구조

Robot Media Laboratory

리스트(List)

- 데이터 추가
- 데이터 탐색
- 데이터 참조
- 데이터 추출
- 데이터 삭제
- 리스트 전체 출력

선형 리스트(Linear List)

- 순서 리스트(Ordered List)
- 자료들 간에 순서를 갖는 리스트

동창 이름 리스트		좋아하는 음식 리스트		오늘의 할 일 리스트	
1	상원	1	김치찌개	1	운동
2	승희	2	닭볶음탕	2	자료구조 스터디
3	수영	3	된장찌개	3	과제 제출
4	철이	4	잡채	4	동아리 공연 연습
…		…		…	

선형 리스트(Linear List)

- 리스트의 표현 형식

리스트 이름 = (원소1, 원소2, ..., 원소n)

- 선형 리스트에서 원소를 나열한 순서는 원소들의 순서가 됨.
  - 동창 = (상원, 승희, 수영, 철이)
- 공백 리스트
  - 원소가 하나도 없는 리스트
  - 빈괄호를 사용하여 표현

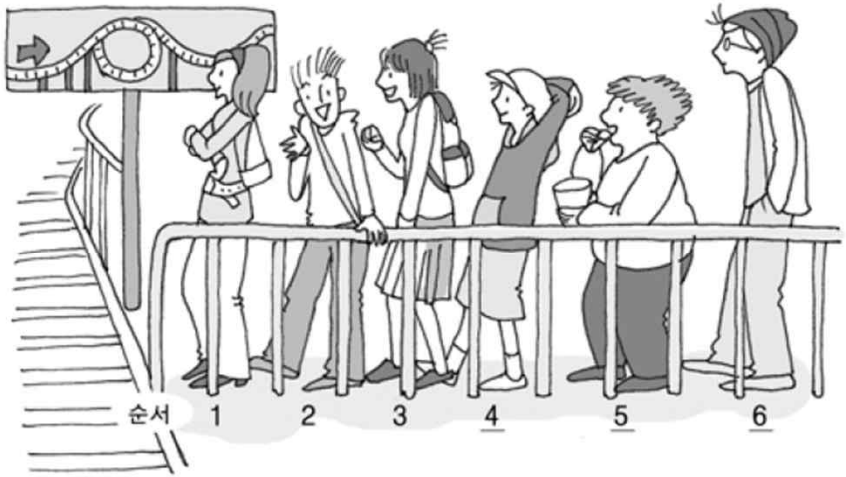
  공백리스트이름 = **( )**

선형 리스트의 저장

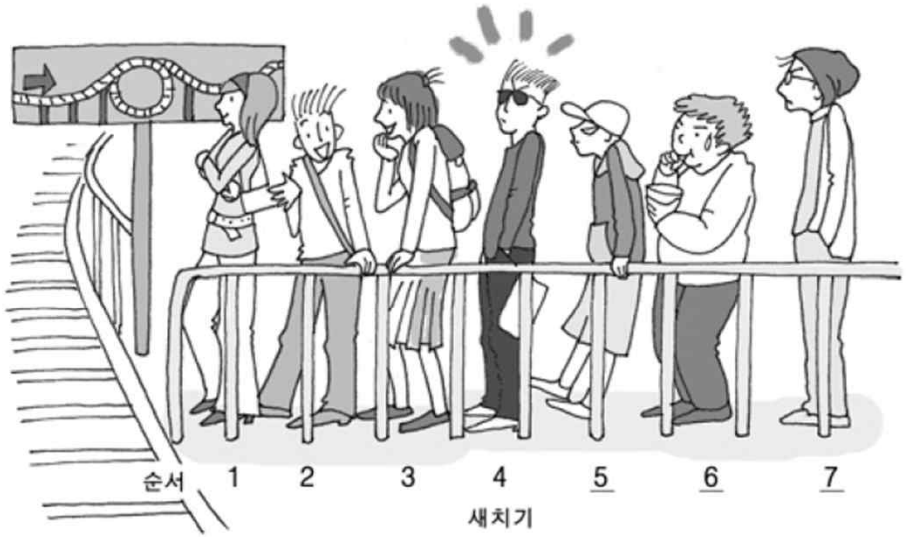
- 원소들의 논리적 순서와 같은 순서로 메모리에 저장
- 순차 자료구조
  - 원소들의 논리적 순서 = 원소들이 저장된 물리적 순서

선형 리스트에서 원소 삽입

- 선형리스트 중간에 원소가 삽입되면, 그 이후의 원소들은 한자리씩 자리를 뒤로 이동하여 물리적 순서를 논리적 순서와 일치시킨다.



(a) 새치기 전

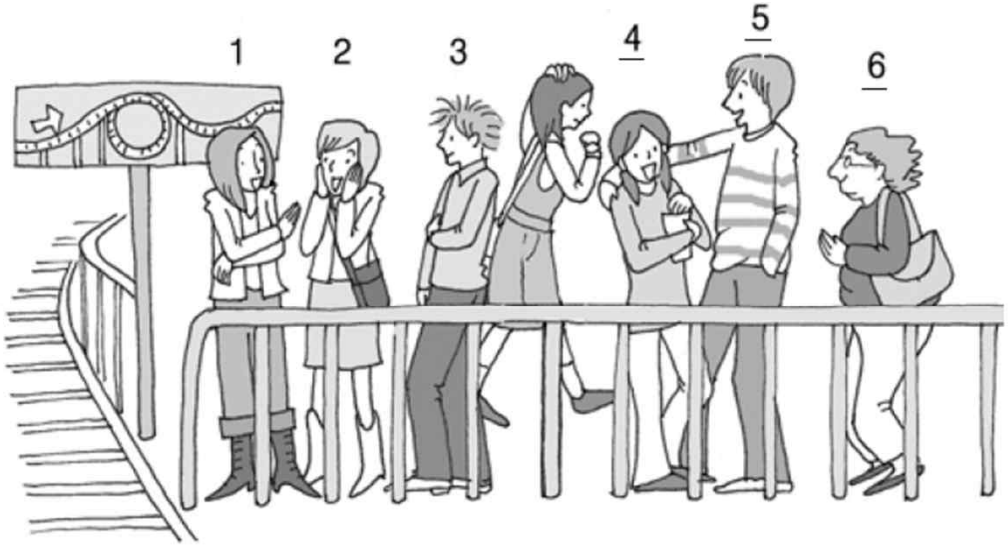


(b) 새치기 후

선형 리스트에서 원소 삭제

- 선형리스트 중간에서 원소가 삭제되면, 그 이후의 원소들은 한자리씩 자리를 앞으로 이동하여 물리적 순서를 논리적 순서와 일치시킨다.

(a) 나가기 전



(b) 나간 후

선형 리스트의 구현

- 순차 구조의 배열을 사용
  - 배열 : <인덱스, 원소>의 순서쌍의 집합
  - 배열의 인덱스 : 배열 원소의 순서 표현

순차 자료구조의 문제점

- 삽입연산이나 삭제연산 후에 연속적인 물리 주소를 유지하기 위해서 원소들을 이동시키는 추가적인 작업과 시간 소요
  - 원소들의 이동 작업으로 인한 오버헤드는 원소의 개수가 많고 삽입・삭제 연산이 많이 발생하는 경우에 성능상의 문제 발생
- 순차 자료구조는 배열을 이용하여 구현하기 때문에 배열이 갖고 있는 메모리 사용의 비효율성 문제를 그대로 가짐
- 순차 자료구조에서의 연산 시간에 대한 문제와 저장 공간에 대한 문제를 개선한 자료 표현 방법 필요

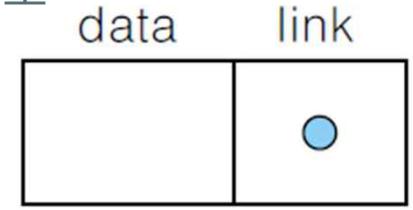
연결 자료구조(Linked Data Structure)

- 자료의 논리적인 순서와 물리적인 순서가 일치하지 않는 자료구조
  - 각 원소에 저장되어 있는 다음 원소의 주소에 의해 순서가 연결되는 방식
    - **물리적인 순서를 맞추기 위한 오버헤드가 발생하지 않음**
  - 여러 개의 작은 공간을 연결하여 하나의 전체 자료구조를 표현
    - **크기 변경이 유연하고 더 효율적으로 메모리를 사용**

- 연결 리스트
  - 리스트를 연결 자료구조로 표현한 구조
  - 연결하는 방식에 따라 단순 연결 리스트와 원형 연결 리스트, 이중 연결 리스트, 이중 원형 연결 리스트

연결 리스트의 노드

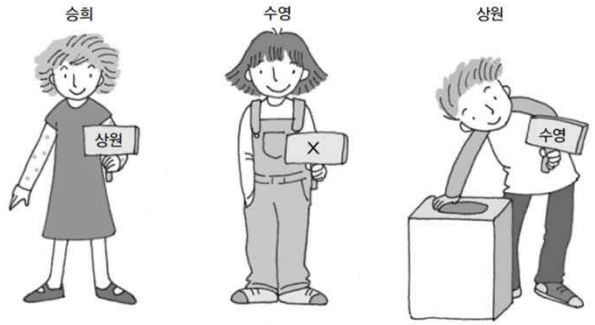
- 연결 자료구조에서 하나의 원소를 표현하기 위한 단위 구조
- <원소, 주소>의 구조



- 데이터 필드(data field)
  - 원소의 값을 저장
  - 저장할 원소의 형태에 따라서 하나 이상의 필드로 구성
- 링크 필드(link field)
  - 다음 노드의 주소를 저장
  - 포인터 변수를 사용하여 주소값을 저장

노드 연결 방법에 대한 이해 - 기차놀이

① 이름표 뽑기



② 자기가 뽑은 이름의 사람을 찾아서 연결하기 : 승희→상원 →수영

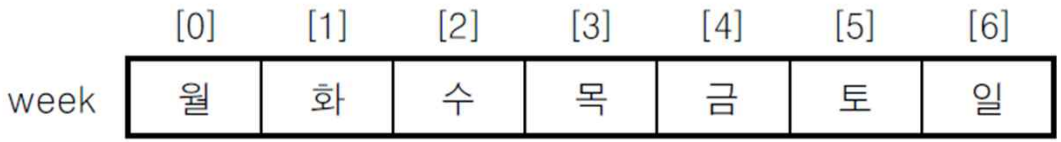
- X 표를 뽑은 사람은 마지막 기차
- 기차는 이름표를 들고 있는 방향으로 움직인다.

노드 연결 방법에 대한 이해 - 기차놀이

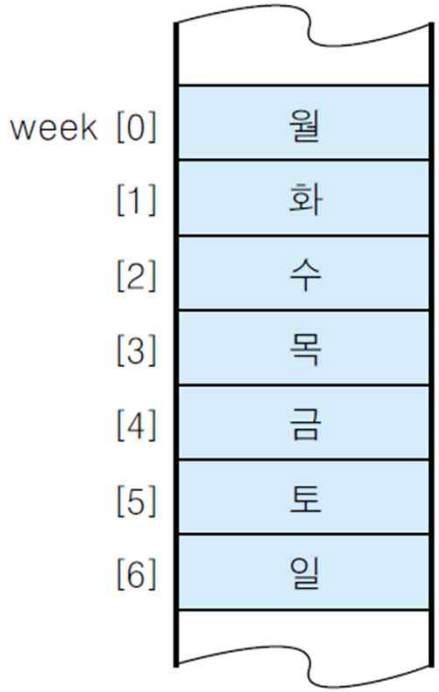
- 기차놀이와 연결 리스트
  - 기차놀이 하는 아이들 : 연결리스트의 노드
  - 이름표 : 노드의 링크 필드

선형 리스트와 연결 리스트의 비교

- 리스트 week=(월, 화, 수, 목, 금, 토, 일)
- week에 대한 선형 리스트



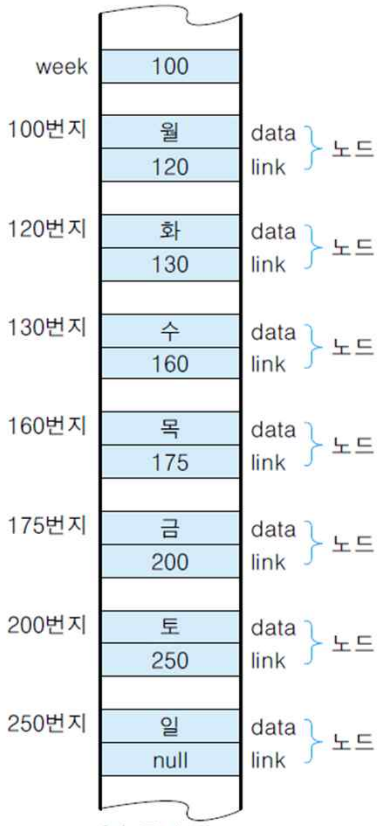
(a) 논리 구조



(b) 물리 구조


week
data
link
월
화
수
목
금
토
일
null

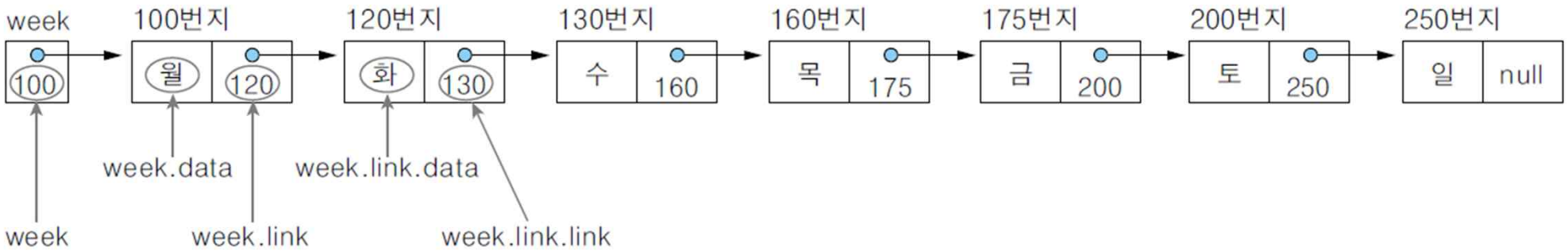
(a) 논리 구조


week
100
100번지
월
120
data
link
노드
120번지
화
130
130번지
수
160
160번지
목
175
175번지
금
200
200번지
토
250
250번지
일
null

(b) 물리 구조

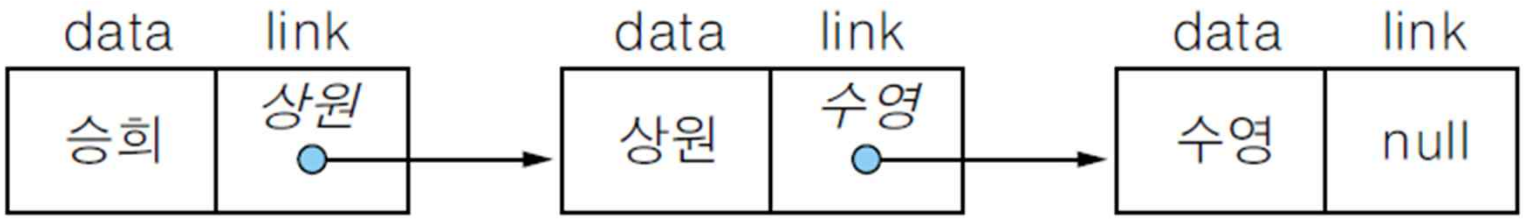
선형 리스트와 연결 리스트의 비교

- **리스트 이름** week - 연결 리스트의 시작을 가리키는 포인터변수
  - 포인터변수 week는 연결 리스트의 첫번째 노드를 가리키는 동시에 연결된 리스트 전체를 의미
- 연결 리스트의 마지막 노드의 링크필드 - 노드의 끝을 표시하기 위해서 **null**(널) 저장
- **공백 연결 리스트** - 포인터변수 week에 null을 저장 (널 포인터)



단순 연결 리스트(singly linked list)

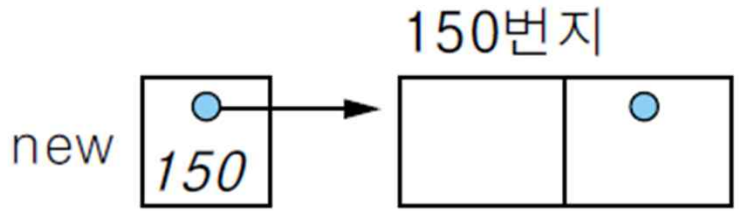
- 노드가 하나의 링크 필드에 의해서 다음 노드와 연결되는 구조를 가진 연결 리스트
- 연결 리스트, 선형 연결 리스트(linear linked list), 단순 연결 선형 리스트(singly linked linear list)
- 예)



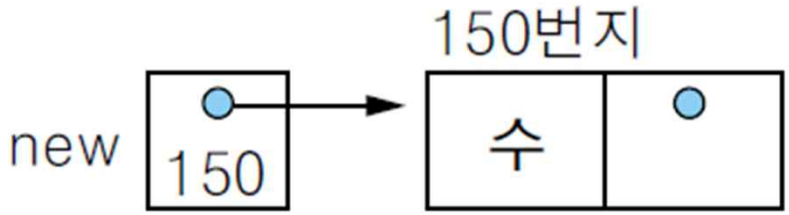
단순 연결 리스트의 삽입

- 리스트 week2=(월, 금, 일)에서 원소 "월"과 "금"사이에 새 원소"수" 삽입하기

① 삽입할 새 노드를 만들 공백노드를 메모리에서 가져와서 포인터변수 new가 가리키게 한다.

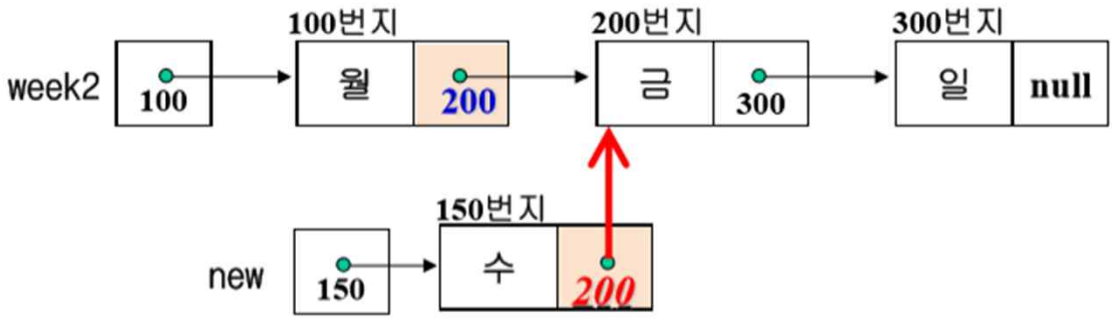


② new의 데이터 필드에 "수"를 저장한다.

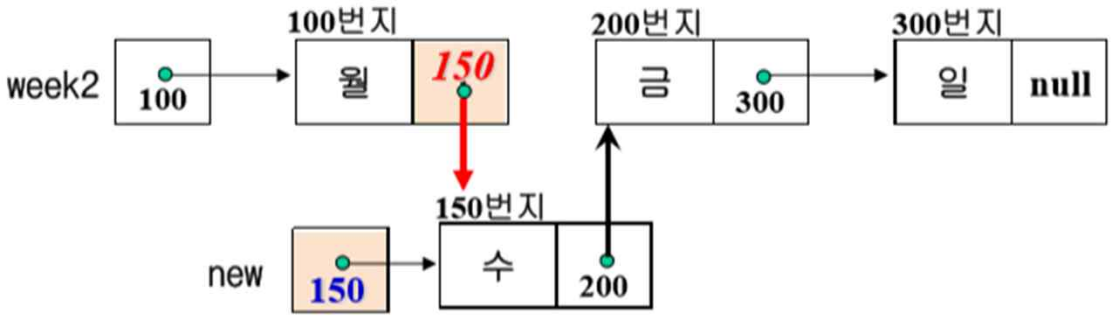


단순 연결 리스트의 삽입

③ new의 앞 노드, 즉 "월"노드의 링크 필드 값을 new의 링크 필드에 저장한다.



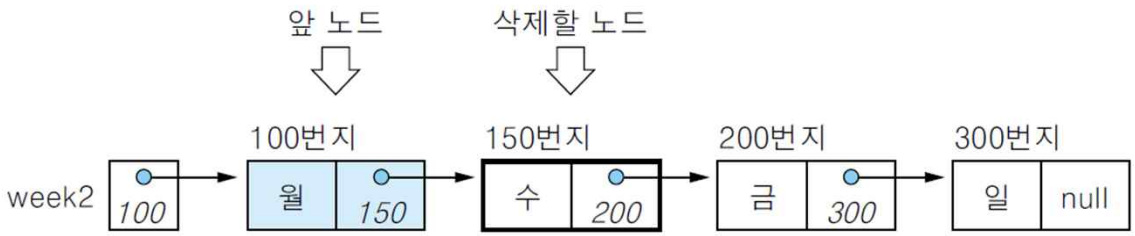
④ new의 값(new가 가리키고 있는 새 노드의 주소)을 "월"노드의 링크 필드에 저장한다



단순 연결 리스트의 삭제

- 리스트 week2=(월, 수, 금, 일)에서 원소 "수" 삭제하기

① 삭제할 원소의 앞 노드(선행자)를 찾는다.



② 삭제할 원소 "수"의 링크 필드 값을 앞 노드의 링크 필드에 저장한다.