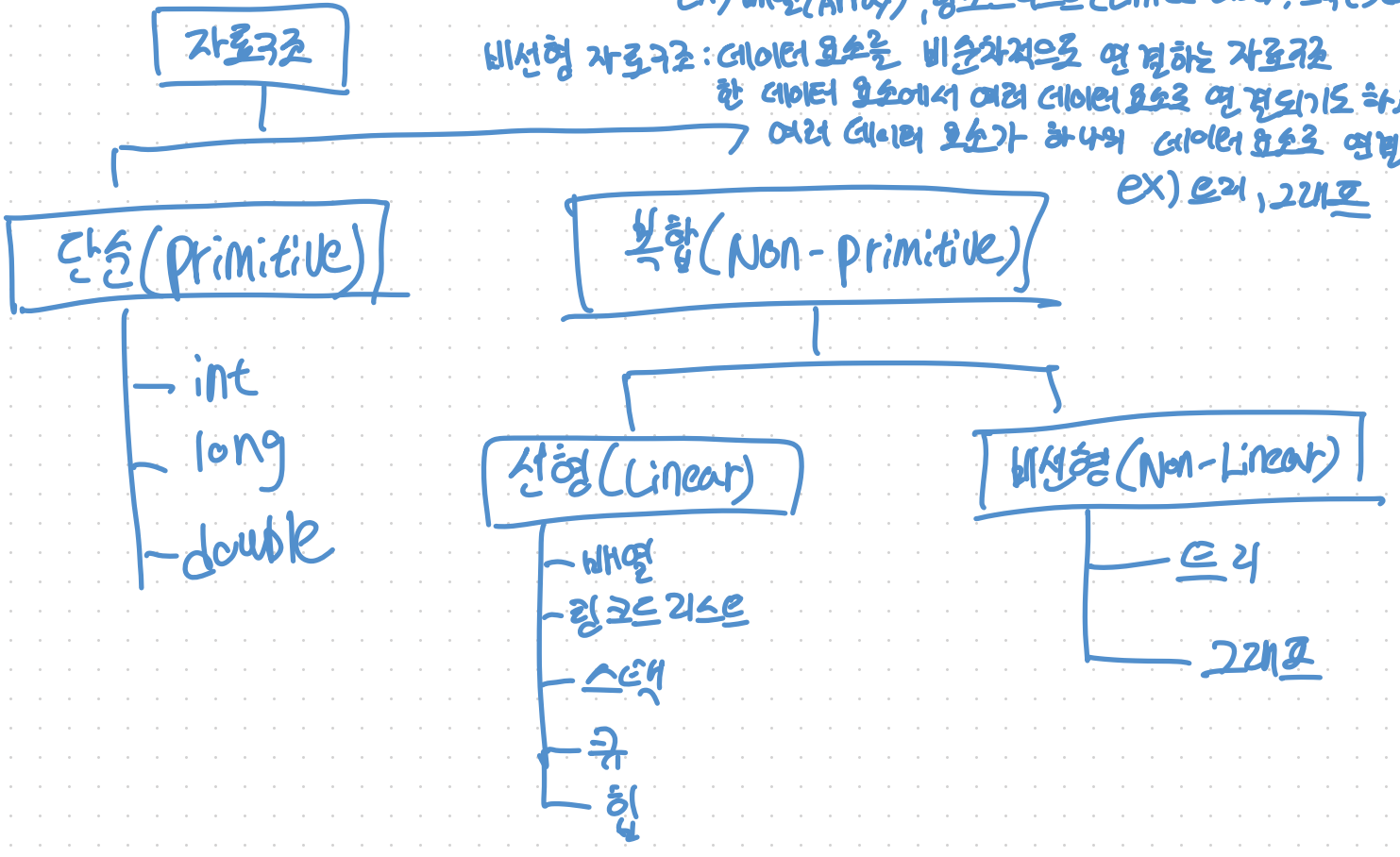
자료구조
Data Structure

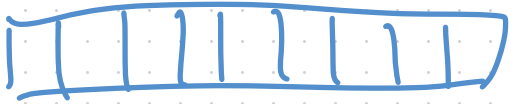
자료구조: 컴퓨터가 데이터를 효율적으로 다룰 수 있게 도와주는 데이터 보관 방법과 데이터에 관한 연산의 총체

선형 자료구조: 데이터 요소를 순차적으로 연결하는 자료구조, 구현 easy
ex) 배열(Array), 링크드리스트(Linked List), 스택(Stack), 큐(Queue)

비선형 자료구조: 데이터 요소를 비순차적으로 연결하는 자료구조
한 데이터 요소에서 여러 데이터 요소로 연결되기도 하고,
여러 데이터 요소가 하나의 데이터 요소로 연결되기도 한다.
ex) 트리, 그래프



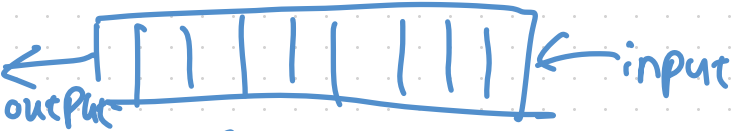
Linear



Array

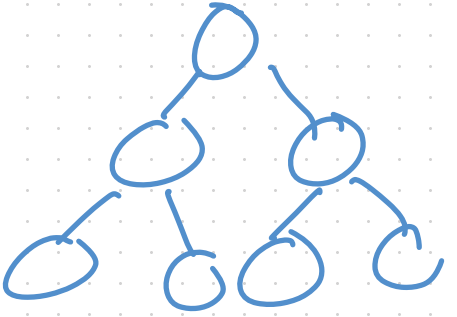
Linked List

Stack

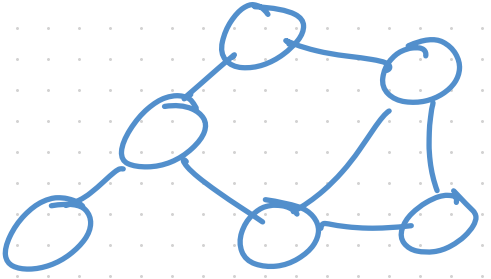


Queue

Non-Linear



Tree



Graph

추상 데이터 형식(ADT, Abstract Data Type)

: 자료구조의 동작 방법을 표현하는 데이터 형식

정리하면 ADT는 자료구조가 갖춰야 할 일련의 연산. → 언어로 표현 → 함수

2.2

ADT	자료구조
리스트	링크드 리스트/더블 링크드 리스트/ 환형 리스트
스택	배열 기반 스택/링크드 리스트 기반 스택
큐	환형 큐 / 링크드 큐
트리	이진트리 / LCRS트리/레드블랙트리
그래프	방향성 그래프 / 무방향성 그래프
힙	배열기반 힙/링크드 리스트기반 힙

자료구조를 공부해야 하는 이유

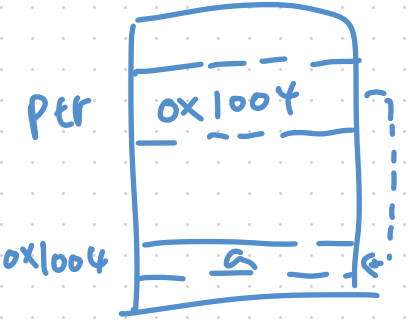
1. 개발하고자 하는 애플리케이션에 따라 적절한 자료구조를 선택할 수 있다.
   때로는 처리 속도를 중요시 해야할 수도 있고, 때로는 메모리 효율이 중요할 수도 있기 때문에
   자료구조 지식이 있으면 적절한 자료구조를 선택할 수 있다.

2. 자료구조는 알고리즘이 데이터를 효율적으로 사용할 수 있도록 돕기 때문이다.
   자료구조를 모르면 알고리즘을 공부하고 사용하기에 어려움이 따른다.

포인터

```
int a = 123;
int* ptr = &a;
```

&a == a의 주소값



구조체

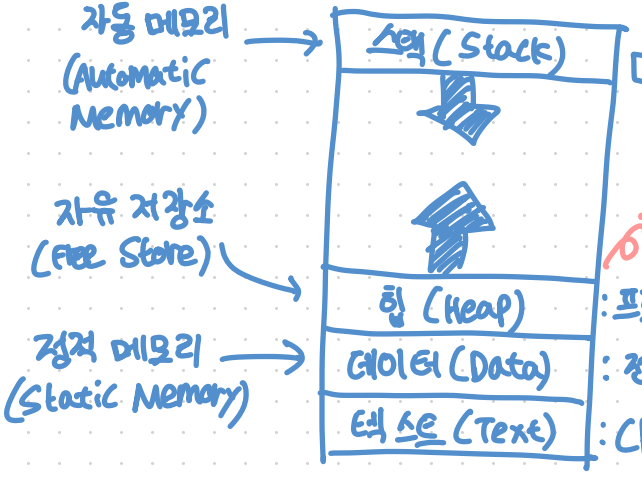
```
typedef struct tagPoint
{
    int x;
    int y;
} Point;

Point MyPoint = {30, 40};
Point* ptr = &MyPoint
```

typedef을 사용하면 구조체 변수를 만들 때마다 struct을 사용하지 않아도 된다.

메모리 레이아웃

C언어 코드
↓ 컴파일
.exe 실행파일
↓ 실행
OS가 실행파일을 바탕으로 프로세스 생성 후 이 프로세스에 스택, 힙, 데이터, 텍스트 영역으로 이루어진 메모리를 할당함.



스택 (Stack): FILO 방식으로 메모리에서 변수를 제거함 (LIFO) (지역변수)
malloc, calloc으로 메모리를 할당할 때 사용되는 영역, 반드시 free()를 통해 해제해야 함.
힙 (Heap): 프로그래머가 직접 메모리를 관리하는 메모리 영역
데이터 (Data): 전역 변수나 정적 변수 등이 저장됨
텍스트 (Text): CPU가 실행할 코드 적재