

Data Structures & Programming

Lecture 02. Data Structure & Algorithm

HyungGi Jo
Div. of Electronics
Jeonbuk National University

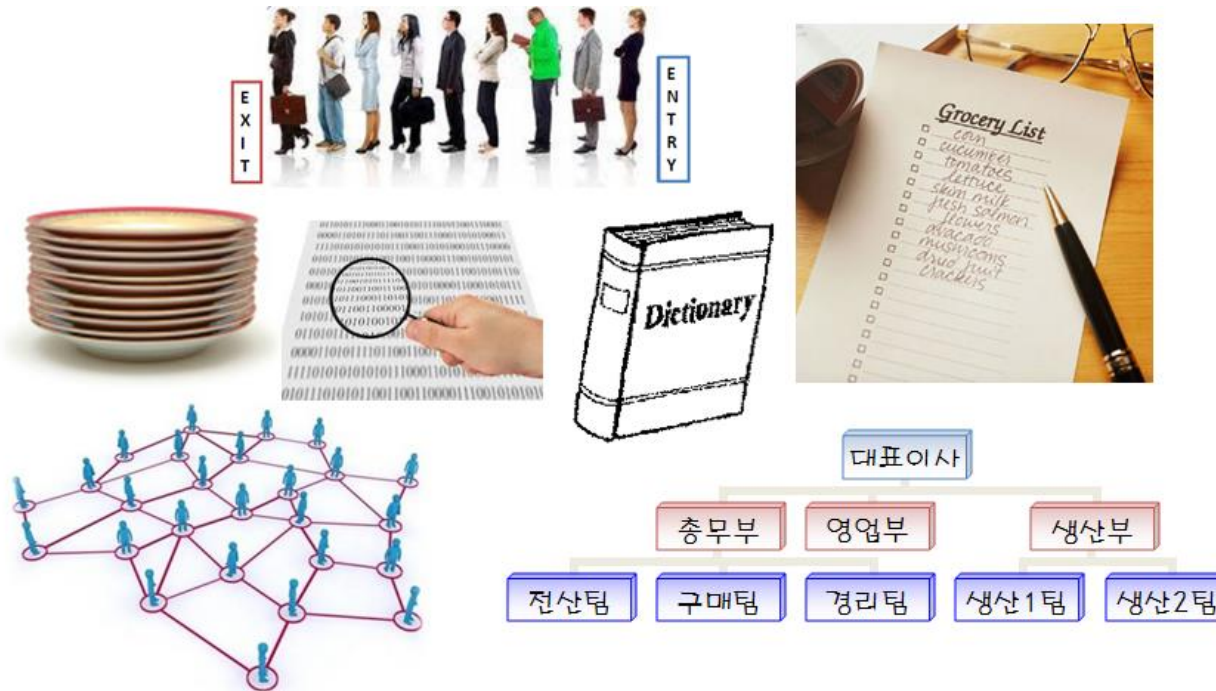
Contents

- The goal of this lecture
 - Data Structure? Algorithm? 에 대한 이해
 - Abstraction(추상화)란?
 - Abstraction Data Type (ADT) 구현하기 위한 방법

Problem-solving

- 우리가 생각하고 행동하는 것들
 - Problem-solving and Develop a solution
- 일상 생활에서 자료를 정리하고 조직화하는 이유
 - 사물을 편리하고 효율적으로 사용하기 위함
 - 다양한 자료를 효율적인 규칙에 따라 정리한 예

자료의 정리/조직화

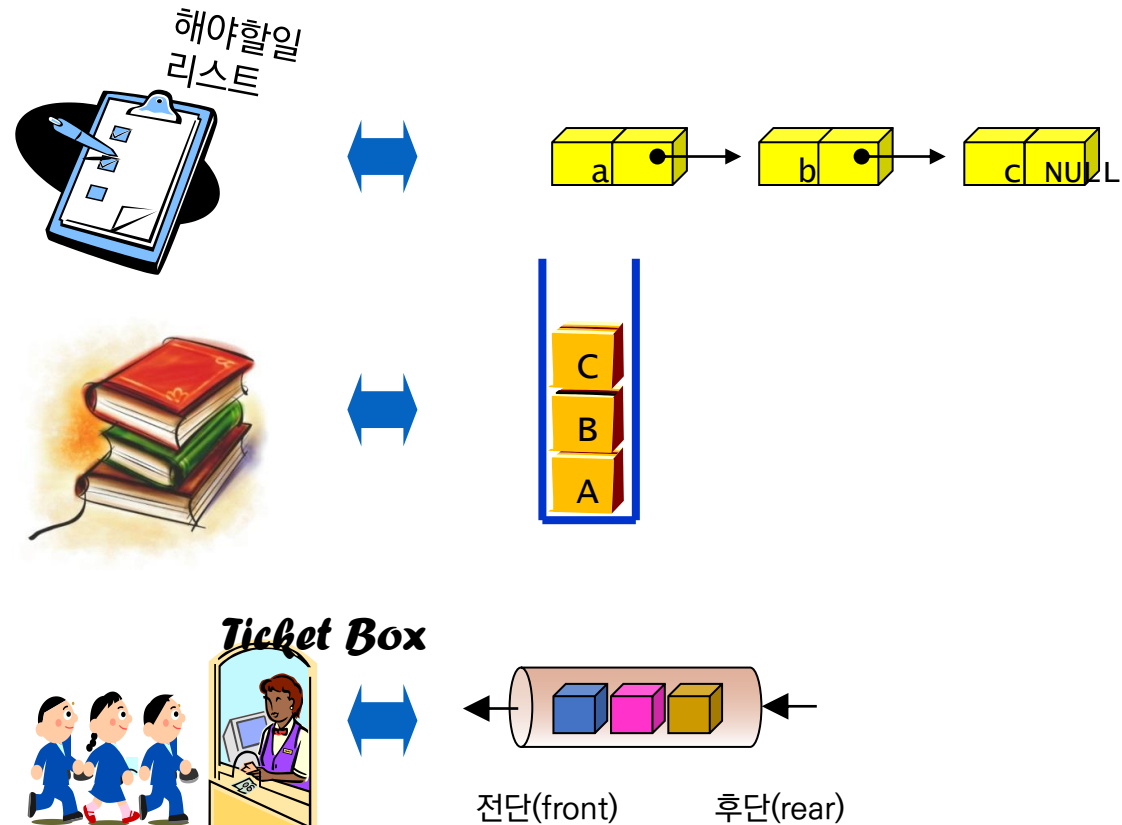


Problem-solving

➤ 다양한 일상 생활 속 문제들

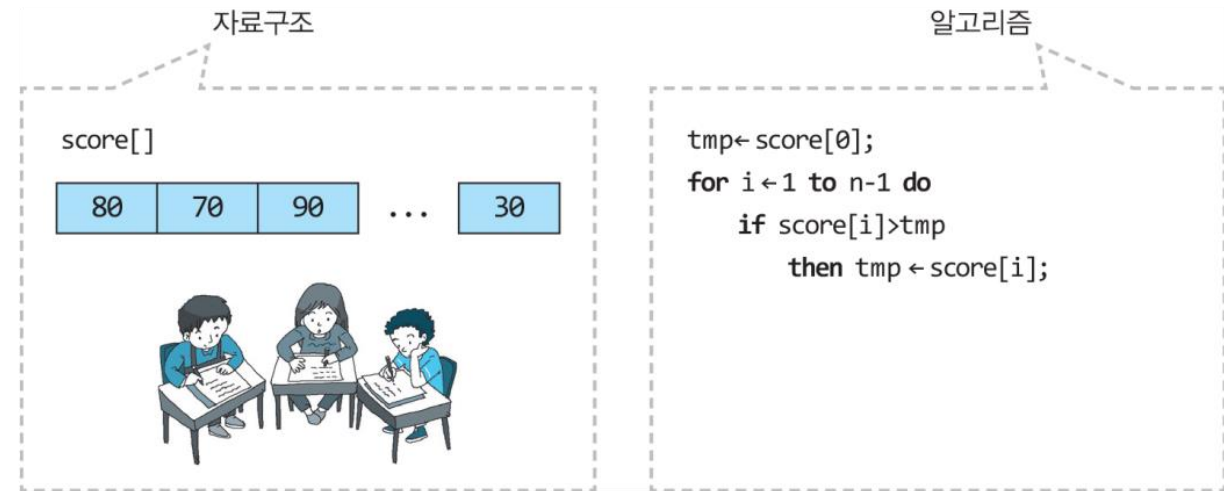
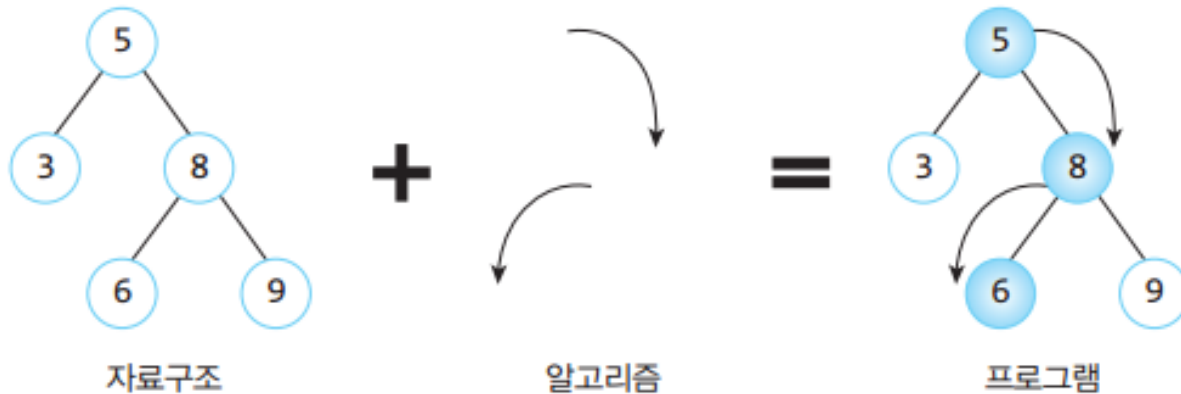
- 사전에서 단어 찾기
- TV에서 볼만한 채널 찾기
- 학교에 가는 최적의 방법 찾기
- 가격 순으로 정렬하기

일상생활에서의 예	자료구조
물건을 쌓아 두는 것	스택
영화관 매표소의 줄	큐
할일 리스트	리스트
영어사전	사전, 탐색구조
지도	그래프
조직도	트리



Computer Science

- 컴퓨터를 사용하여 문제를 해결(Problem-Solving) 하는 방법을 연구
- Programming을 통해 문제의 Solution을 구현 => **Algorithm**
- Program = **Data Structure** + **Algorithm**



(예) 최대값 탐색 프로그램 = **배열** + **순차탐색**

Computer Science

➤ Algorithm?

- 컴퓨터로 문제를 풀기 위한 단계적인 절차
- 입력(input)을 출력(output)으로 변환해주는 Computational steps
- 같은 문제를 서로 다른 방법으로 풀어 나가면서 어떤 방식이 효율적인지를 확인

➤ Algorithm 표현 방법

- 영어나 한국어와 같은 자연어
- 흐름도(flow chart)
- 유사 코드(pseudo-code)
- 특정한 프로그래밍 언어 (C언어, C++, java 등)

Computer Science

➤ 자연어

- 인간이 읽기가 쉽다.
- 단어들을 정확하게 정의하지 않으면 의미 전달이 모호해질 우려가 있다.

배열에서 최대값 찾기 알고리즘

ArrayMax(A, n)

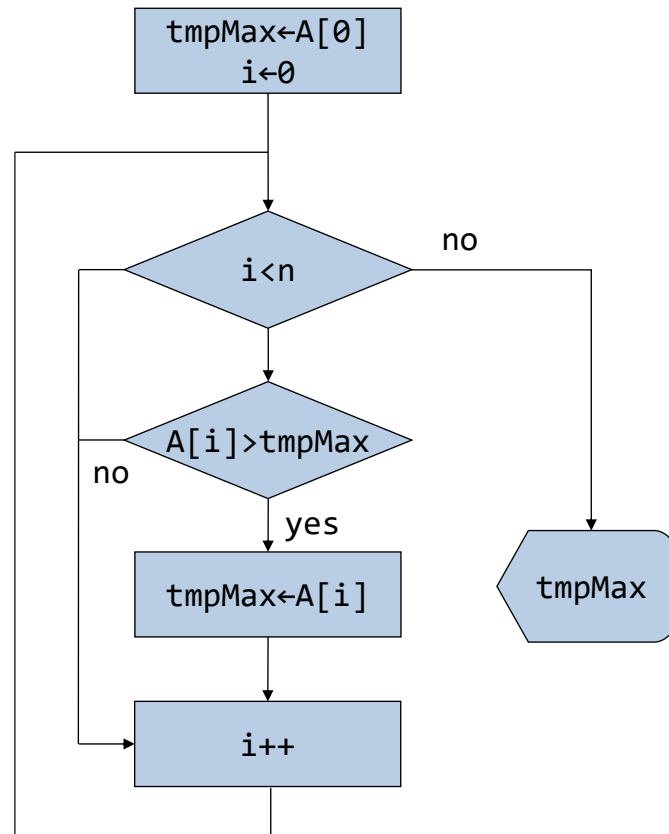
1. 배열 A의 첫 번째 요소를 변수 tmp에 복사
2. 배열 A의 다음 요소들을 차례대로 tmp와 비교하면 더 크면 tmp로 복사
3. 배열 A의 모든 요소를 비교했으면 tmp를 반환

Computer Science

➤ 흐름도

- 직관적이고 이해하기 쉬운 알고리즘 기술 방법
- 그러나 복잡한 알고리즘의 경우, 상당히 복잡해짐.

배열에서 최대값 찾기 알고리즘



Computer Science

➤ 유사코드(Pseudo code)

- 알고리즘의 고수준 기술 방법
- 자연어보다는 더 구조적인 표현 방법
- 프로그래밍 언어보다는 덜 구체적인 표현방법
- 알고리즘 기술에 가장 많이 사용
- 프로그램을 구현할 때의 여러 가지 문제들을 감출 수 있음
- 알고리즘의 핵심적인 내용에만 집중할 수 있음

배열에서 최대값 찾기 알고리즘

ArrayMax(A,n)

```
tmp ← A[0];  
for i ← 1 to n-1 do  
    if tmp < A[i]  
    then tmp ← A[i];  
return tmp;
```

Computer Science

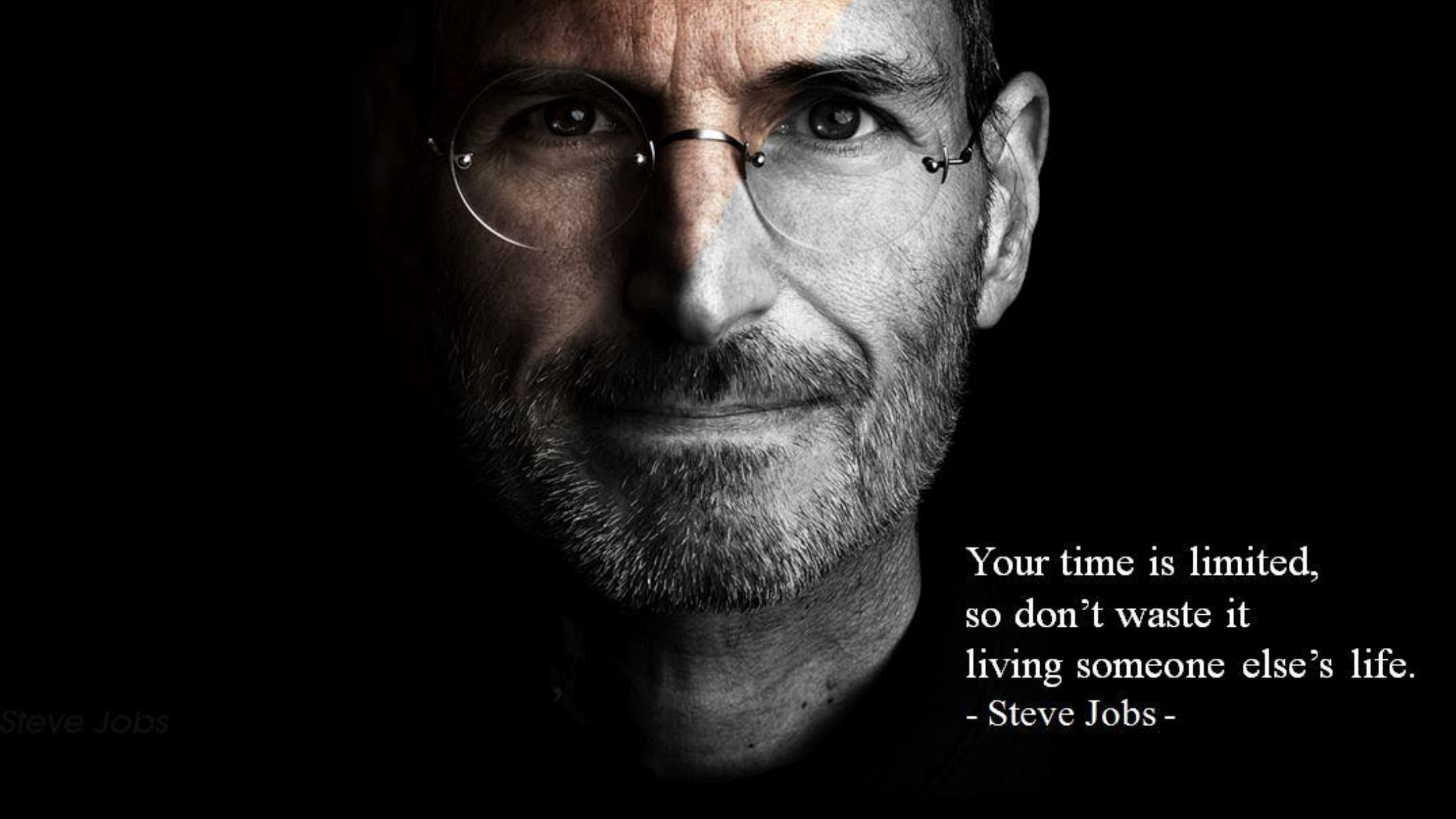
➤ 특정 프로그래밍 언어

- 알고리즘의 가장 정확한 기술 가능
- 실제 구현시의 많은 구체적인 사항들이 알고리즘의 핵심적인 내용들의 이해를 방해할 수 있음
- 예) C언어로 표기된 알고리즘

배열에서 최대값 찾기 알고리즘

```
#define MAX_ELEMENTS 100
int score[MAX_ELEMENTS];

int find_max_score(int n) {
    int i, tmp;
    tmp=score[0];
    for(i=1;i<n;i++){
        if( score[i] > tmp ){
            tmp = score[i];
        }
    }
    return tmp;
}
```

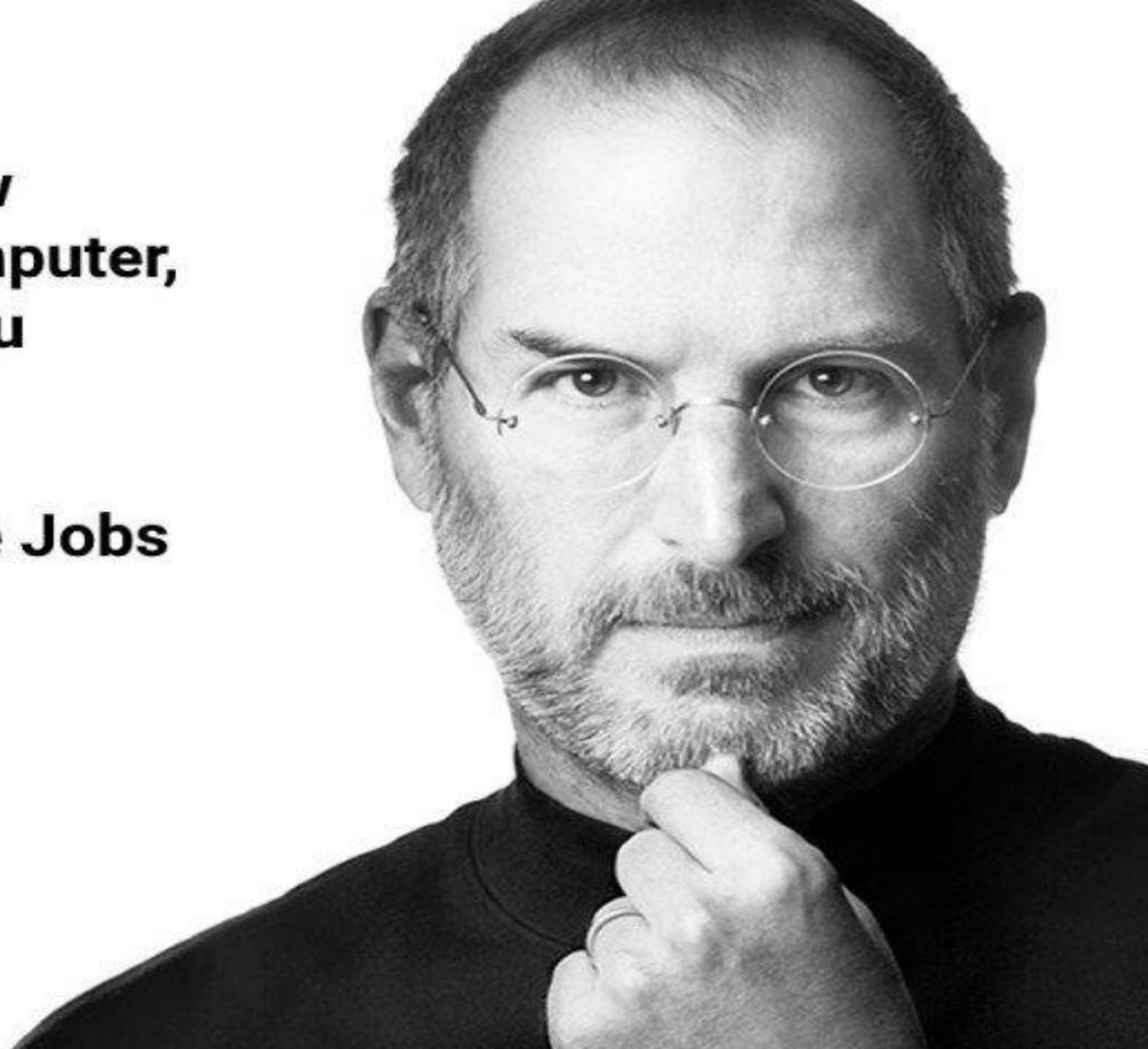


Your time is limited,
so don't waste it
living someone else's life.
- Steve Jobs -

Steve Jobs

**"Everyone should know
how to program a computer,
because it teaches you
how to think."**

Steve Jobs



Programming

- Programming은 Algorithm을 선택하고 컴퓨터의 언어로 표현하는 기술
- Computer Science는 프로그래밍을 배우는 것이 아니고,
 - Algorithm을 이용한 문제 해결 방법을 배우는 것
 - 그리고 Abstraction(추상화)에 대해서 배우는 것

Abstraction

➤ Abstraction(추상화):

- 어떤 시스템의 간략화 된 기술 또는 명세
- 시스템의 정말 핵심적인 구조나 동작에만 집중

➤ 추상 자료형(ADT: Abstract Data Type)

- 어떤 자료나 연산이 제공되는가 만을 정의
- 어떻게 구현되는지는 정의하지 않음
- 시스템의 핵심적인 구조나 동작에만 집중

자료형(data type)

데이터의 집합과 연산의 집합

예) int 데이터 타입

데이터: $\{ \dots, -2, -1, 0, 1, 2, \dots \}$

연산: $+, -, /, *, \%$

Abstraction

➤ Example – 자동차

Logical Perspective: 사용자(운전자) 관점

차에서 제공하는 기능(Interface)들을 사용
차 문 열기
탑승
시동 걸기
사이드 브레이크
엑셀, 브레이크
핸들 이용해서 조향

Physical Perspective: 기술자 관점

어떻게 차가 여러 기능들을 제공하는지에 대해 자세하게 알아야 함
어떻게 차 문이 열리고
어떻게 엔진이 점화하며
어떻게 기어 변속이 되는지..

Abstraction

➤ Example – Computer

Logical Perspective: 사용자 관점

컴퓨터에서 제공해주는 기능(Interface)들을 사용
문서 작성
문서 저장
웹 서핑
메신저 사용

Physical Perspective: 기술자 관점

어떻게 컴퓨터가 여러 기능들을 제공하는지에 대해
자세하게 알아야 함
어떻게 웹 반응을 하고
어떻게 문서를 체계적으로 저장하며
어떻게 다음 곡으로 넘기는지

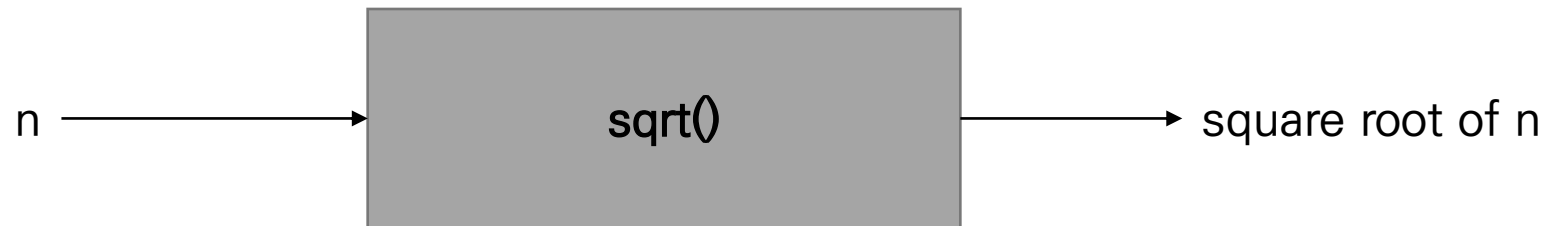
Abstraction

➤ Example – python *math* module

```
import math
math.sqrt(16)

[2] ✓ 0.4s
... 4.0
```

- 실제 sqrt() 내부에서 어떤 연산이 일어나는지 관심 X
- 입력 형태, 함수명, 반환 값에 대한 정보만 표현 : Interface



Abstraction

➤ Abstraction

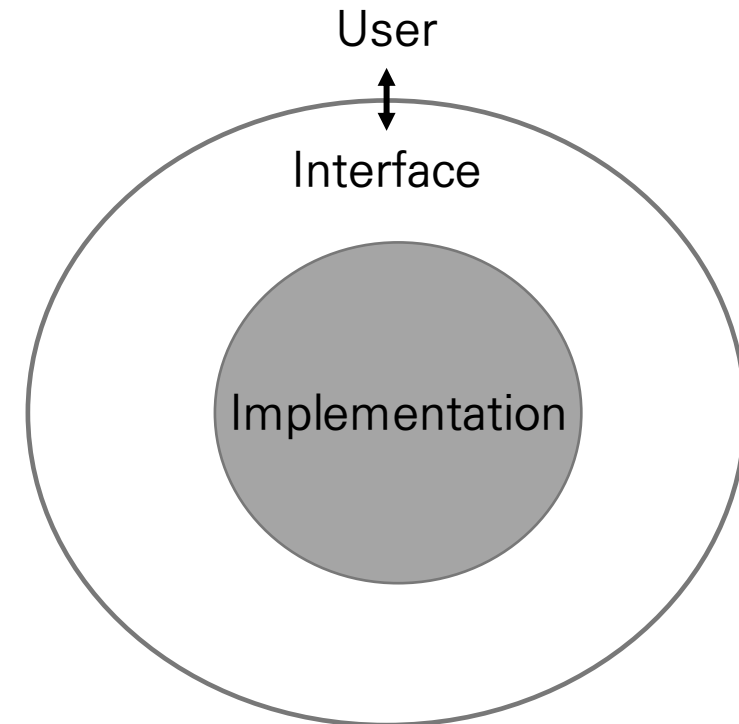
- 어떤 자료나 연산이 제공되는가(interface) 만을 정의, 어떻게 구현되는가는 정의하지 않음
- *Big Picture*

➤ Abstraction data type (ADT)

- 추상적으로 정의한 자료형 (Logical description)
- Encapsulation (information hiding)

➤ Data Structure

- 데이터를 저장하는 방식
- Data Structure = 구현된 ADT (Physical Perspective)



Data Structure

- Data Structure
 - 컴퓨터에서 자료를 정리하고 조직화하는 다양한 구조
- Linear Data Structure
 - 항목들을 순서적으로 나열하여 저장하는 창고
 - 항목 접근 방법에 따라 다시 세분화
 - List: 가장 자유로운 선형 자료구조
 - Stack, Queue, Deck: 항목의 접근이 맨 앞(전단)이 나 맨 뒤(후단)로 제한
- Non-linear Data Structure
 - 항목들이 보다 복잡한 연결 관계
 - Tree: 회사의 조직도나 컴퓨터의 폴더와 같은 계층 구조
 - Heap Tree: Priority Queue
 - Binary Search Tree, AVL Tree: 탐색을 위한 트리 구조
 - Graph: 가장 복잡한 연결 관계를 표현
- 다양한 문제를 해결하기 위한 기본 구조

Exercise

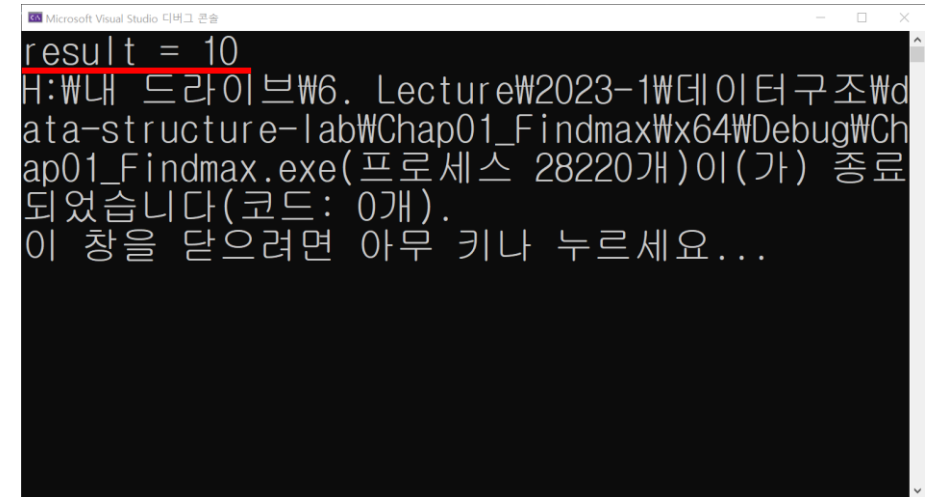
➤ 배열의 최대값을 찾는 C++ 프로그램 작성 & Github Push

1. 새 프로젝트 만들기 → 빈 프로젝트 → 프로젝트 이름: Chap01_Findmax
2. 소스 파일 오른쪽 클릭 → 새 항목 → C++ 파일 추가 (파일 이름: find_max.cpp)
3. 실행

```
#include <stdio>

int findArrayMax(int score[], int n)    // 자료구조: 배열 array, n은 배열 길이
{
    int tmp=score[0];
    for( int i=1 ; i<n ; i++ ){        // 알고리즘
        if( score[i] > tmp ){
            tmp = score[i];
        }
    }
    return tmp;
}

// 주 함수
void main()
{
    int score[5] = { 1, 10, 2, 4, 5 };
    int out;
    out = findArrayMax(score, 5);
    printf("result = %d", out);
}
```



```
Microsoft Visual Studio 디버그 콘솔
result = 10
H:\내 드라이브\6. Lecture\2023-1\데이터구조\data-structure-lab\Chap01_Findmax\64\Debug\Chap01_Findmax.exe(프로세스 28220개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Exercise

➤ Github Push

- 새로운 항목 추가 되었으니 Commit 후 Push 하자!
- 잘 연결되었는지 확인
 - `$ git remote -v`
- Commit & Push
 - `$ git add .`
 - `$ git commit -m "message"`
 - `$ git push origin master`

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a commit history table is displayed. The first commit, 'johg15 chap01 project added', is highlighted with a red box. This commit includes a folder 'Chap01_Findmax' (labeled 'chap01 project added') and a file 'ReadMe.txt' (labeled 'new line added'). Below the commit history, the content of 'ReadMe.txt' is shown, containing the text 'Hi hello' and 'Welcome to Data Structure Class'.

Commit Message	Commit Hash	Time	Commits
johg15 chap01 project added	011074e	15 minutes ago	3 commits
Chap01_Findmax		15 minutes ago	
ReadMe.txt		1 hour ago	

ReadMe.txt

```
Hi hello

Welcome to Data Structure Class
```