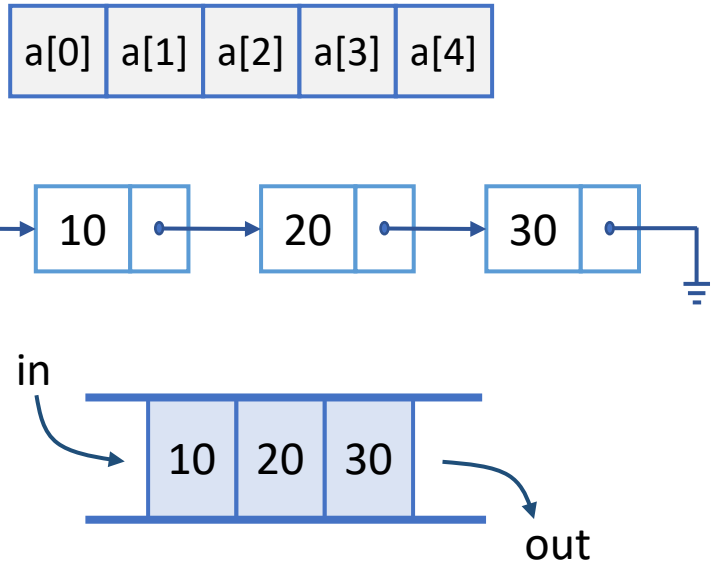
1. 자료 구조와 알고리즘은 처음이지?

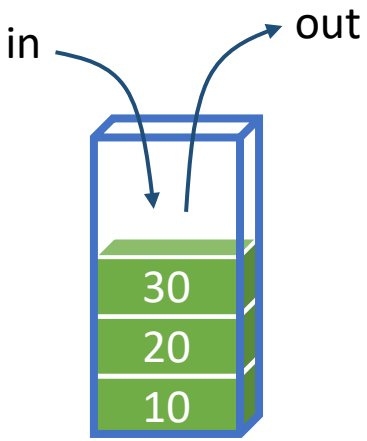
1) 강의 개요

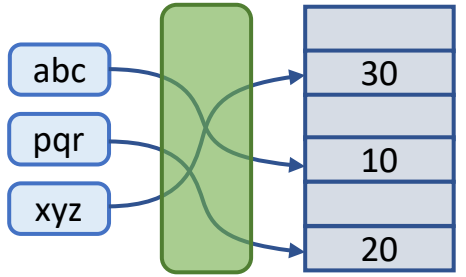
Presented by sunkyoo.hwang@gmail.com

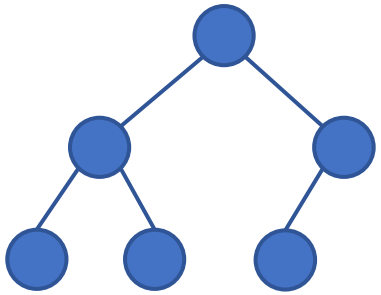
자료 구조와 알고리즘

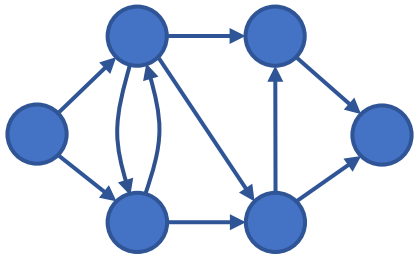
- 자료 구조(data structure)
  - 자료(data)를 효율적으로 이용하기 위한 자료의 저장 방식
    - 자료의 형식, 자료 사이의 관계, 자료를 이용하는 방식(연산, 함수)
    - 효율적: 실행 시간 + 메모리 사용량
    - 주어진 문제에 적합한 자료 구조를 선택함으로써 효율적인 알고리즘을 사용할 수 있음







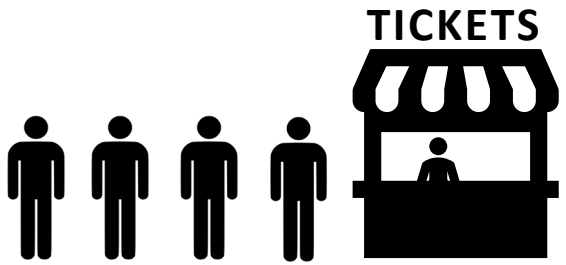




Presented by sunkyoo.hwang@gmail.com

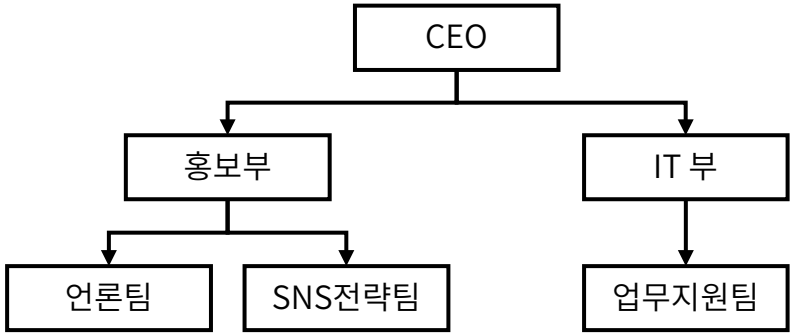
자료 구조와 알고리즘

- 일상 생활에서의 자료 구조

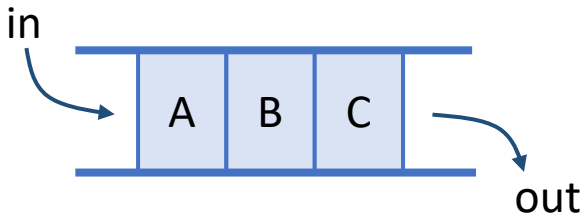


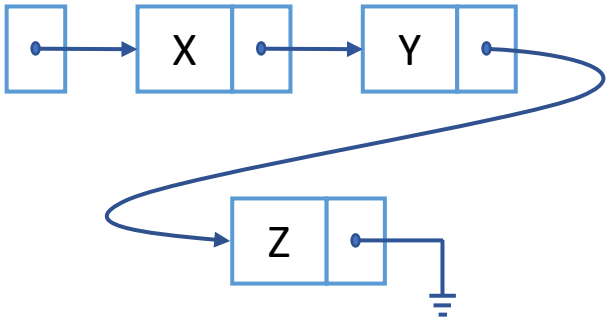
매표소의 대기열

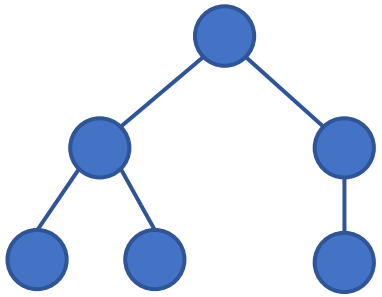
TO DO 리스트



회사 조직도







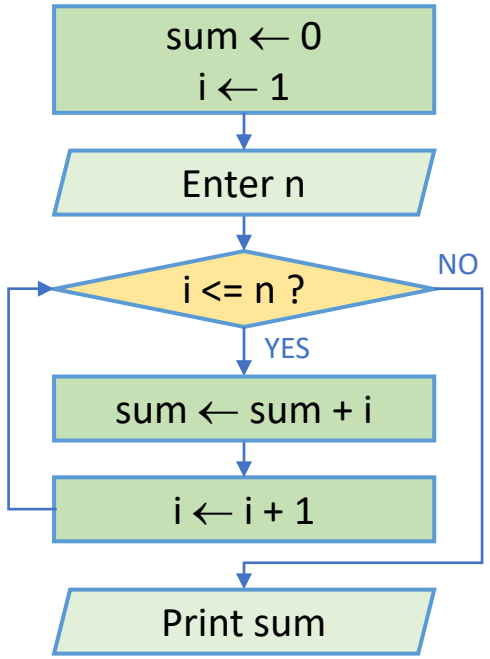
Presented by sunkyoo.hwang@gmail.com

자료 구조와 알고리즘

- 알고리즘(algorithm)
  - 컴퓨터로 문제를 해결하기 위한 일련의 절차나 방법
  - 입력, 출력 / 명확성, 유한성, 유효성
  - 자연어, 순서도(flowchart), 의사코드(pseudo-code), 프로그래밍 언어 코드 등

summation(n)

- 변수 sum을 선언하고 0으로 초기화
- 변수 i를 0부터 n까지 증가시키면서 sum에 i를 더함
- sum 값을 반환



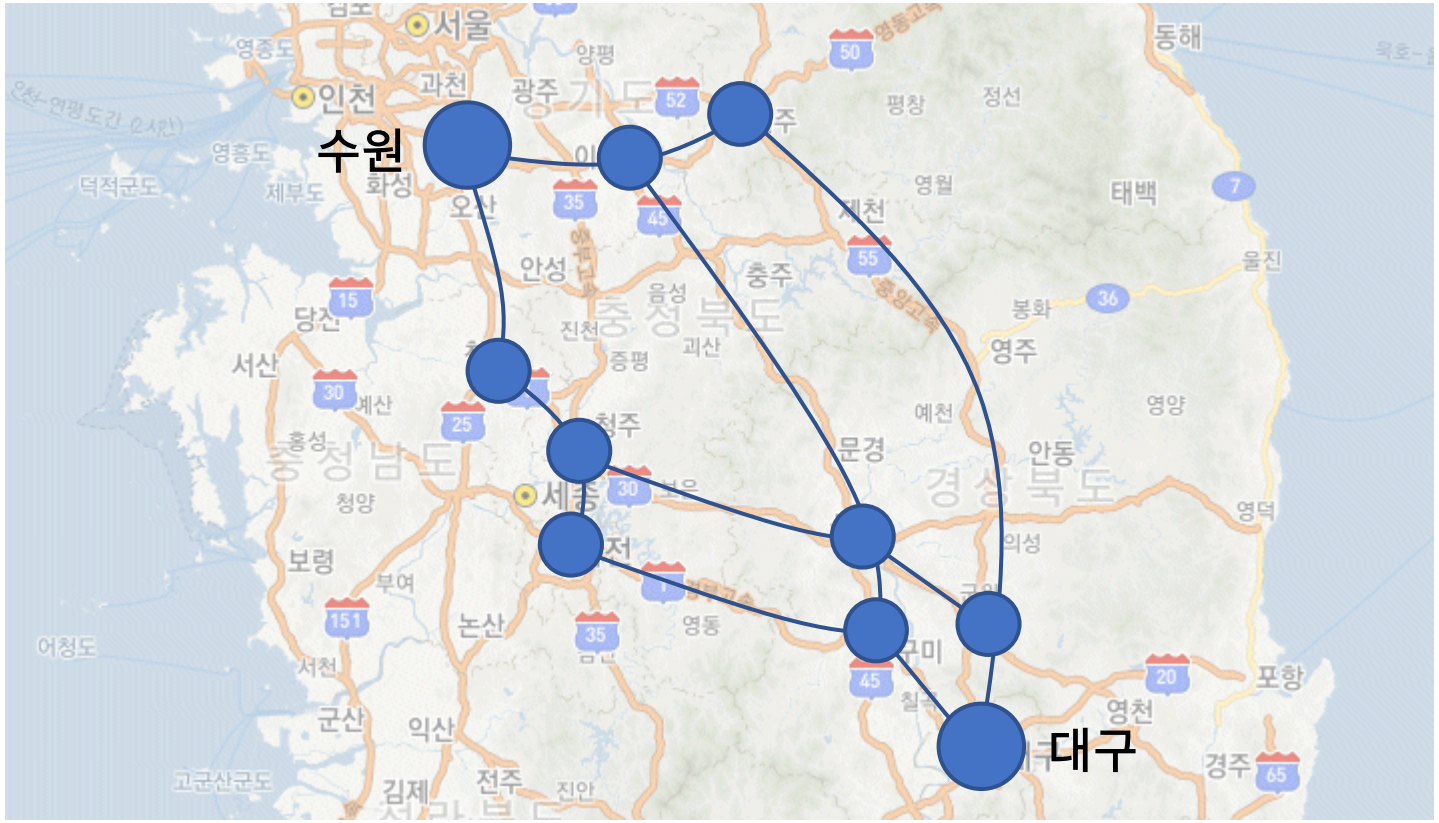
summation(n)

```
sum = 0
for i = 1 to n:
    sum = sum + i
    i = i + 1

return sum
```


Presented by sunkyoo.hwang@gmail.com

자료 구조와 알고리즘

- 수원에서 대구까지 가장 빠른 경로는?



그래프 자료 구조

+

최단 거리 알고리즘

Presented by sunkyoo.hwang@gmail.com

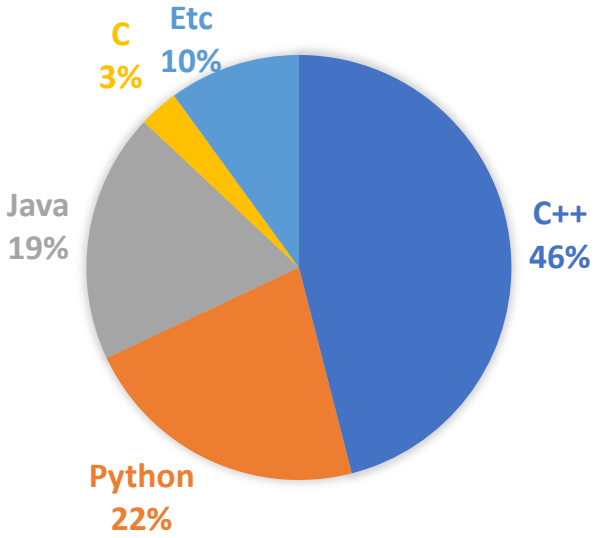
코딩 테스트를 위한 프로그래밍 언어

- 현업에서 사용되고 있는 주요 프로그래밍 언어
  - Java, JavaScript, Python, C, C++, SQL, C#, PHP, etc.

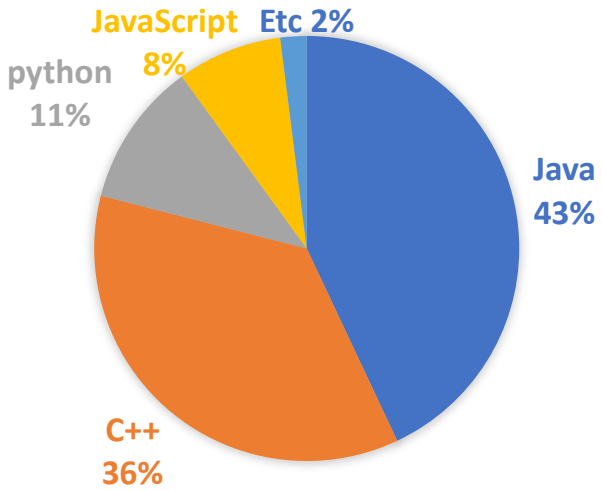
Image from https://programmers.co.kr/pages/dev-survey-2020#developer-tech-stack-lang

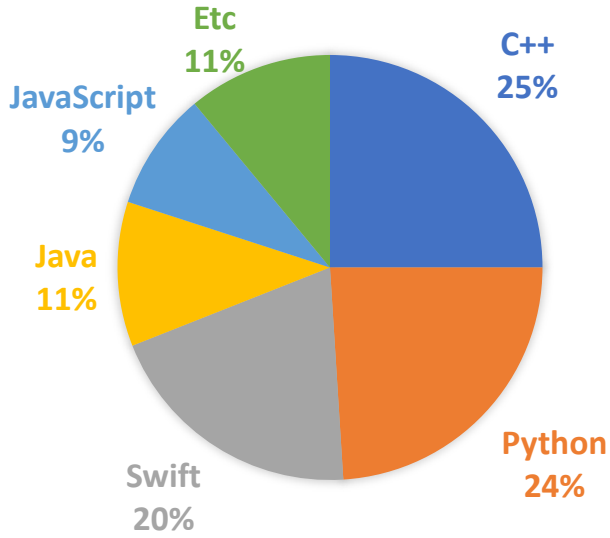
코딩 테스트를 위한 프로그래밍 언어

- 코딩 테스트에서 주로 사용되는 프로그래밍 언어
  - 널리 사용되는 자료 구조와 알고리즘을 제공하는 언어를 사용해야 효율적



2017년 구글 코드잼



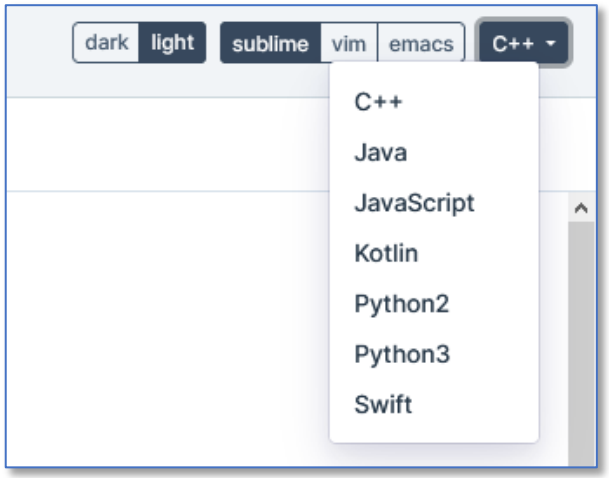


2017년 카카오 신입 개발자 코딩 테스트 (응시율, 합격율)

Ref: https://www.go-hero.net/jam/17/languages, https://tech.kakao.com/2017/09/27/kakao-blind-recruitment-round-1/

코딩 테스트를 위한 프로그래밍 언어

- C++ 언어의 장점
  - 전통적인 프로그래밍 언어
    - 방대한 온라인 리소스
    - 거의 모든 코딩 플랫폼에서 지원
  - 표준 라이브러리에서 다양한 자료 구조와 알고리즘을 지원
  - 실행 속도가 빠름
  - 메모리 관리에서 자유도가 높음



2020년 카카오 신입 개발자
코딩 테스트 지원 언어

- C++ 코딩테스트 주의 사항
  - 적절한 표준 라이브러리 선택 능력
  - C++11, 14, 17 등 모던 C++ 문법 숙지
  - 코딩 컨벤션 (코딩 스타일)
  - 메모리 관리 (메모리 릭 주의)

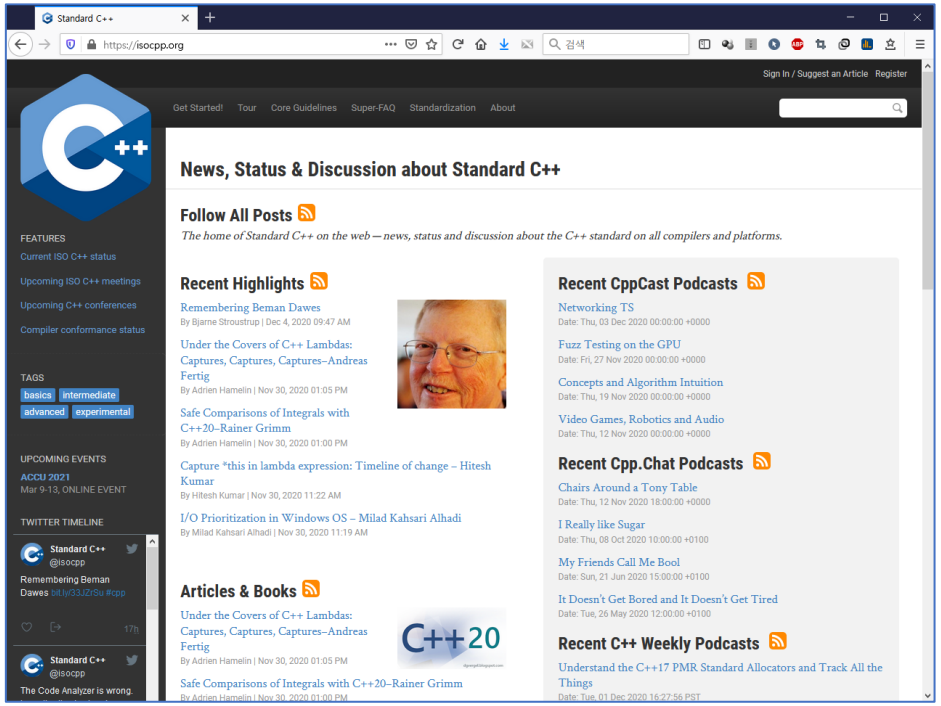
언어	컴파일러	컴파일 커맨드
C	Clang 10.0	clang -O2 -o FILENAME
C++	Clang++ 10.0 (C++17)	clang++ -std=c++17 -O2 -Wno-unused-result -o FILENAME

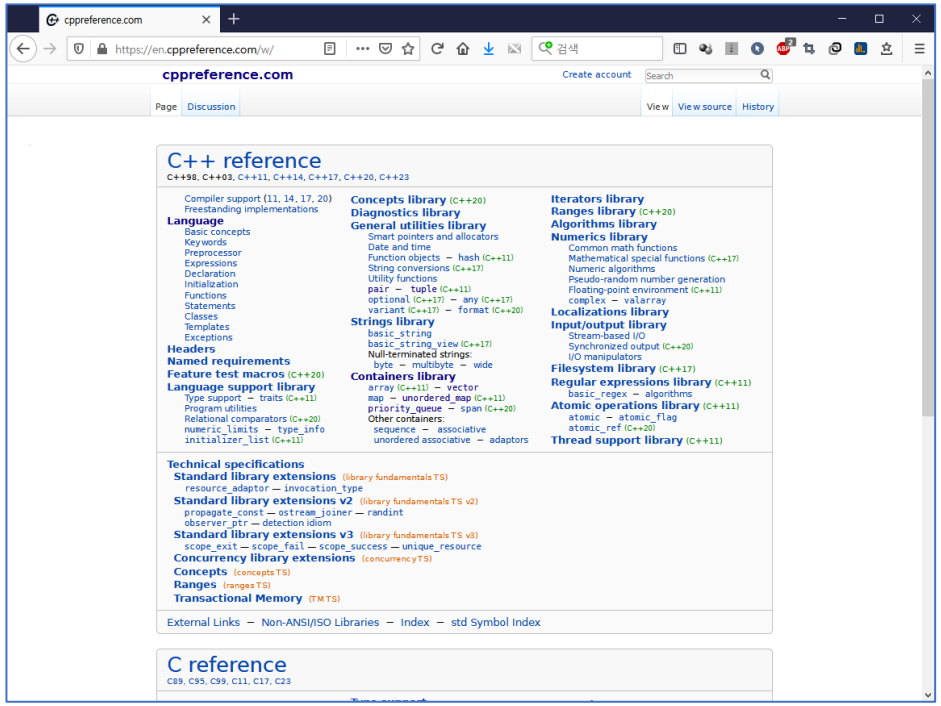
Images from https://programmers.co.kr/

Presented by sunkyoo.hwang@gmail.com

코딩 테스트를 위한 프로그래밍 언어

- C++ 참고 사이트
  - C++ 표준: https://isocpp.org/
  - C++ 레퍼런스: https://cppreference.com/
    https://www.cplusplus.com/reference/





수강 대상과 강의 목표

- 수강 대상

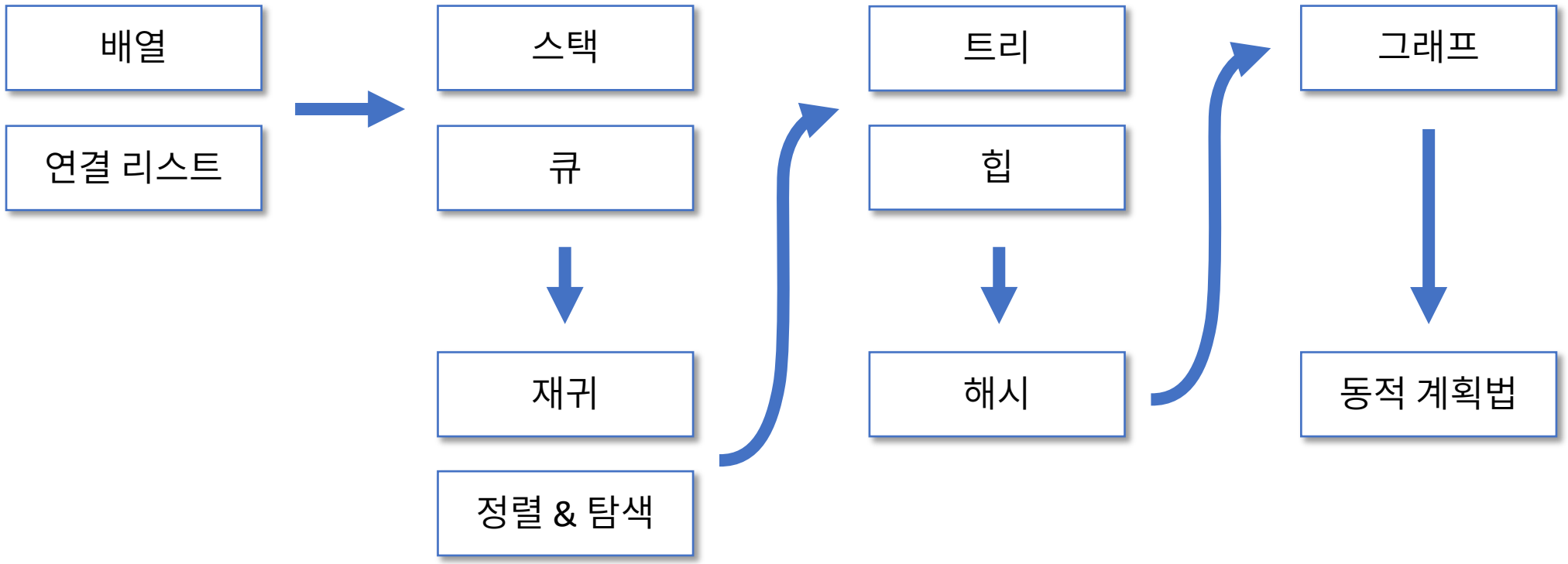
  - C++ 기초 문법을 충분히 이해하고 있는 학생 및 개발자
  - C++ 기초 문법: C 언어 문법, 클래스, 템플릿, STL(표준 템플릿 라이브러리)과 반복자 개념 등

  - C++ 언어를 이용하여 코딩 테스트를 준비하는 취업준비생
  - 자료 구조와 알고리즘을 C++ 언어 특성에 맞게 다시 배우고 싶으신 분
  - 현업 C++ 개발자 중에서 좀 더 실력 있는 개발자가 되고 싶은 분

- 강의 목표

  - 자료 구조와 알고리즘의 기본 원리 이해
  - C++ 언어를 이용한 기본적인 자료 구조와 알고리즘 구현 방법
  - STL에서 제공하는 자료 구조와 알고리즘의 효율성을 이해하고, 실제 코딩 테스트 문제 풀이에 적합한 기능을 선택할 수 있는 능력

강의 커리큘럼



1. 자료 구조와 알고리즘은 처음이지?

2) C++ 프로그램 개발 환경 설정

Presented by sunkyoo.hwang@gmail.com

다양한 C++ 프로그램 개발 환경

Visual Studio

Code::Blocks

Dev C++

Xcode

GCC

VIM

GCC

VS Code

MinGW-w64

VS Code

Presented by sunkyoo.hwang@gmail.com

MinGW-w64와 VS Code를 이용한 C++ 개발 환경 설정

1. VS Code 설치
   - https://code.visualstudio.com/ 에서 설치 실행 파일 다운로드 받은 후 설치 진행
2. C++ 개발을 위한 VS Code 확장 설치
   - C/C++ 확장
   - (Optional) Korean Language Pack for Visual Studio Code 확장
3. MinGW-w64 설치
   - https://sourceforge.net/projects/mingw-w64/files/ 에서 필요한 파일을 다운로드 받은 후 설치 진행
   - MinGW-w64가 설치된 폴더를 사용자 환경변수 PATH에 추가
4. C++ 예제 프로그램 작성 & 빌드 & 실행
   - C++ 소스 코드를 저장할 폴더를 열고, 새로운 C++ 소스 파일을 작성
   - g++.exe 를 사용하여 빌드하고 실행하도록 설정

Presented by sunkyoo.hwang@gmail.com

MinGW-w64와 VS Code를 이용한 C++ 개발 환경 설정

1. VS Code 설치
   - https://code.visualstudio.com/ 에서 설치 실행 파일 다운로드 받은 후 설치 진행
2. C++ 개발을 위한 VS Code 확장 설치
   - C/C++ 확장
   - (Optional) Korean Language Pack for Visual Studio Code 확장
3. MinGW-w64 설치
   - https://sourceforge.net/projects/mingw-w64/files/ 에서 필요한 파일을 다운로드 받은 후 설치 진행
   - MinGW-w64가 설치된 폴더를 사용자 환경변수 PATH에 추가
4. C++ 예제 프로그램 작성 & 빌드 & 실행
   - C++ 소스 코드를 저장할 폴더를 열고, 새로운 C++ 소스 파일을 작성
   - g++.exe 를 사용하여 빌드하고 실행하도록 설정

Presented by sunkyoo.hwang@gmail.com

MinGW-w64와 VS Code를 이용한 C++ 개발 환경 설정

1. VS Code 설치
   - https://code.visualstudio.com/ 에서 설치 실행 파일 다운로드 받은 후 설치 진행
2. C++ 개발을 위한 VS Code 확장 설치
   - C/C++ 확장
   - (Optional) Korean Language Pack for Visual Studio Code 확장
3. MinGW-w64 설치
   - https://sourceforge.net/projects/mingw-w64/files/ 에서 필요한 파일을 다운로드 받은 후 설치 진행
   - MinGW-w64가 설치된 폴더를 사용자 환경변수 PATH에 추가
4. C++ 예제 프로그램 작성 & 빌드 & 실행
   - C++ 소스 코드를 저장할 폴더를 열고, 새로운 C++ 소스 파일을 작성
   - g++.exe 를 사용하여 빌드하고 실행하도록 설정

Presented by sunkyoo.hwang@gmail.com

MinGW-w64와 VS Code를 이용한 C++ 개발 환경 설정

1. VS Code 설치
   - https://code.visualstudio.com/ 에서 설치 실행 파일 다운로드 받은 후 설치 진행
2. C++ 개발을 위한 VS Code 확장 설치
   - C/C++ 확장
   - (Optional) Korean Language Pack for Visual Studio Code 확장
3. MinGW-w64 설치
   - https://sourceforge.net/projects/mingw-w64/files/ 에서 필요한 파일을 다운로드 받은 후 설치 진행
   - MinGW-w64가 설치된 폴더를 사용자 환경변수 PATH에 추가
4. C++ 예제 프로그램 작성 & 빌드 & 실행
   - C++ 소스 코드를 저장할 폴더를 열고, 새로운 C++ 소스 파일을 작성
   - g++.exe 를 사용하여 빌드하고 실행하도록 설정

Presented by sunkyoo.hwang@gmail.com

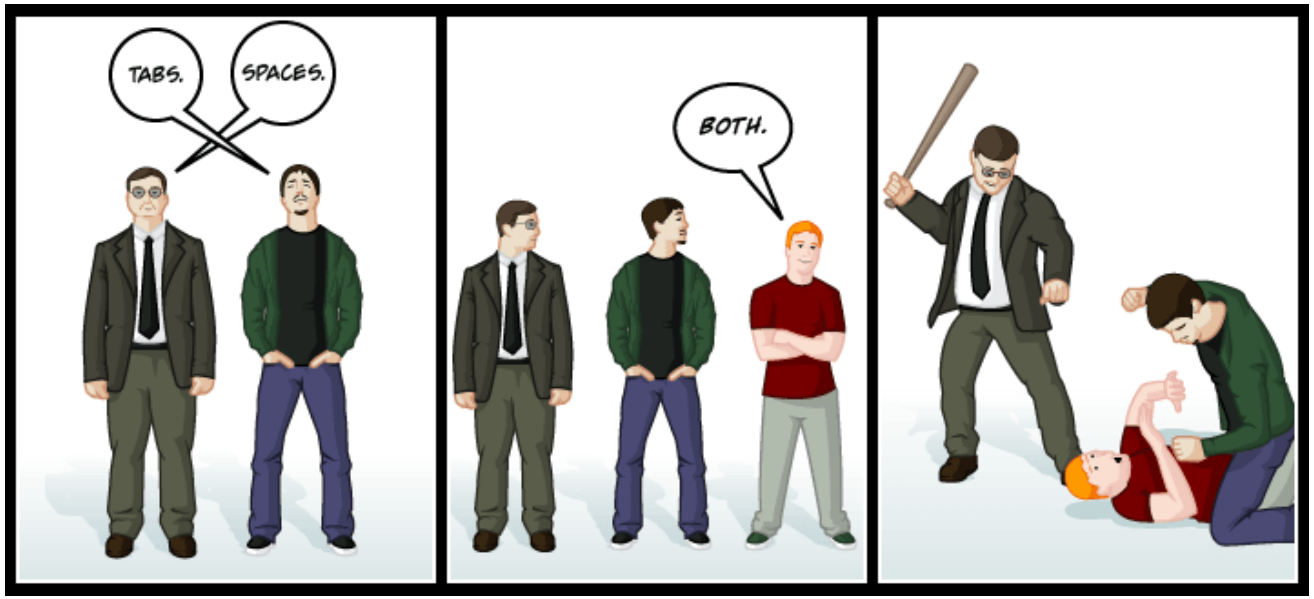
1. 자료 구조와 알고리즘은 처음이지?

3) 코딩 컨벤션

Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션

- 코딩 컨벤션(coding convention)
  - 가독성 있는 코드를 작성하기 위한 공통의 코드 작성 가이드라인. 코딩 스타일(coding style).
  - 여러 명의 개발자가 소스 코드를 공유하거나 함께 관리할 때 유용
  - 코딩을 잘하는지 판단하는 척도로도 사용



코딩 컨벤션

- 주요 C++ 코딩 스타일 가이드라인
  - ISOCPP C++ 가이드라인 https://github.com/isocpp/CppCoreGuidelines
  - 구글 C++ 가이드라인 https://google.github.io/styleguide/cppguide.html
  - PPP 스타일 가이드라인 http://www.stroustrup.com/Programming/PPP-style.pdf

```
std::string table_name;  // OK
std::string tableName;   // Bad


if(condition) {      // Bad
if ( condition ) {   // Bad
if (condition){      // Bad
if(condition){       // Doubly bad
```

```
void foo(bool b) {
  if (b) {
    baz(2);
  } else {
    baz(5);
  }
}
```

```
void foo(bool b)
{
  if (b)
  {
    baz(2);
  }
  else
  {
    baz(5);
  }
}
```

```
void foo(bool b)
{
  if (b) {
    baz(2);
  } else {
    baz(5);
  }
}
```

✓ 공통 항목은 반드시 준수하고, 다른 항목은 통일성 있게 작성!

코딩 컨벤션: 명명 규칙

- 명명 규칙(naming)
  - 타입, 변수, 함수 등의 이름은 의미를 알 수 있는 형태로 구성
  - 너무 단순하거나 또는 지나치게 장황한 이름은 자제

```
int a1, a2, a3, a, b, c; // BAD
int i, j, k; // OK for counting
int high_score_in_class_A; // BAD
```

- 타입 이름(클래스, 구조체, 타입 별칭, 열거형 등)
  - 타입 이름은 (가급적) 대문자로 시작
  - 여러 단어로 이루어진 이름의 경우, 각 단어는 대문자로 시작하고, (가급적) 밑줄은 사용하지 않음 (CamelCase)

```
class StudentInfo
{
private:
    std::string name;
    int age;
};
```

- 변수 이름 & 함수 이름
  - 변수와 함수 이름은 (가급적) 모두 소문자와 밑줄을 사용
  - 상수, 매크로 이름은 대문자와 밑줄로 구성

```
std::string table_name; // OK
std::string tableName;  // Bad

void print_tables(...) {}

#define MAX_STUDENTS 30
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 중괄호
    - 클래스 이름과 { 사이에 공백 삽입
    - if, for, switch, while 문에서 { 앞에 공백 삽입
    - 클래스 이름과 { 사이에 줄바꿈 가능
    - for, switch, if, while 문과 { 사이에 줄바꿈 가능

```
class box{
public:
    int x,y,width,height;
};
int range_sum ( int a,int b )
{
    int sum=0;
    for( int i=a ; i<= b ; i++ ){
        sum+=i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 중괄호
    - 클래스 이름과 { 사이에 공백 삽입
    - if, for, switch, while 문에서 { 앞에 공백 삽입
    - 클래스 이름과 { 사이에 줄바꿈 가능
    - for, switch, if, while 문과 { 사이에 줄바꿈 가능

```
class box {
public:
    int x,y,width,height;
};
int range_sum ( int a,int b )
{
    int sum=0;
    for( int i=a ; i<= b ; i++ ) {
        sum+=i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 소괄호
    - 함수 이름과 (사이에 공백 제거
    - ✓ if, for, switch, while 키워드와 (사이에 공백 삽입
    - 함수 인자 또는 if, for, switch, while 문에서 여는 괄호 뒤, 닫는 괄호 앞에 공백 제거

```
class box {
public:
    int x,y,width,height;
};
int range_sum ( int a,int b )
{
    int sum=0;
    for( int i=a ; i<= b ; i++ ) {
        sum+=i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 소괄호
    - 함수 이름과 (사이에 공백 제거
    - ✓ if, for, switch, while 키워드와 (사이에 공백 삽입
    - 함수 인자 또는 if, for, switch, while 문에서 여는 괄호 뒤, 닫는 괄호 앞에 공백 제거

```
class box {
public:
    int x,y,width,height;
};
int range_sum(int a,int b)
{
    int sum=0;
    for (int i=a ; i<= b ; i++) {
        sum+=i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 쉼표, 세미콜론, 콜론, 연산자
    - 쉼표와 세미콜론 앞에는 공백 제거,
      뒤에는 공백 삽입
    - 범위 기반 for 문에서 콜론(:) 앞뒤에 공백 삽입
    - 대입 연산자, 이항 연산자 앞뒤에 공백 삽입

```
class box {
public:
    int x,y,width,height;
};
int range_sum(int a,int b)
{
    int sum=0;
    for (int i=a ; i<= b ; i++) {
        sum+=i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 쉼표, 세미콜론, 콜론, 연산자
    - 쉼표와 세미콜론 앞에는 공백 제거,
      뒤에는 공백 삽입
    - 범위 기반 for 문에서 콜론(:) 앞뒤에 공백 삽입
    - 대입 연산자, 이항 연산자 앞뒤에 공백 삽입

```
class box {
public:
    int x, y, width, height;
};
int range_sum(int a, int b)
{
    int sum = 0;
    for (int i = a; i <= b; i++) {
        sum += i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 빈 줄(vertical whitespace)
    - 함수, 클래스 정의 뒤에 빈 줄 삽입
    - if, for, switch, while 문이 끝난 후,
      논리적 구분을 위해 가급적 빈 줄 삽입

```
class box {
public:
    int x, y, width, height;
};
int range_sum(int a, int b)
{
    int sum = 0;
    for (int i = a; i <= b; i++) {
        sum += i;
    }
    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 공백

- 공백(whitespace)
  - 빈 줄(vertical whitespace)
    - 함수, 클래스 정의 뒤에 빈 줄 삽입
    - if, for, switch, while 문이 끝난 후,
      논리적 구분을 위해 가급적 빈 줄 삽입

```
class box {
public:
    int x, y, width, height;
};

int range_sum(int a, int b)
{
    int sum = 0;
    for (int i = a; i <= b; i++) {
        sum += i;
    }

    return sum;
}
```


Presented by sunkyoo.hwang@gmail.com

코딩 컨벤션: 기타

- 기타 코딩 컨벤션
  - 들여쓰기(indent)는 Tab 또는 Space 4개 권장
  - 변수는 선언과 동시에 초기화
  - (가급적) 전역 변수의 사용은 자제
  - "Magic constant"는 (가급적) 사용을 자제
    - 특정 숫자를 갑작스럽게 코드에 사용 금지
    - 의미를 알 수 있는 상수 변수 선언 권장
  - if, for, while 문에 하나의 문장만 있더라도 (가급적) 다음 줄에 작성
    - (가급적) 중괄호로 블록 지정

```
int month[12];
...
for (int i = 0; i < 12; ++i) // BAD
    cout << month[i] << endl;
```

```
const int MAX_MONTHES = 12;

int month[MAX_MONTHES] {31, 29, ..., 31};

for (int i = 0; i < MAX_MONTHES; ++i)
    cout << month[i] << endl;
```

```
for (int i = 0; i < MAX_MONTHES; ++i) {
    cout << month[i] << endl;
}
```


Presented by sunkyoo.hwang@gmail.com

VS Code에서 코딩 컨벤션 설정하기

- VS Code에서 코딩 스타일 지정하기(code formatting)
  - 전체 문서 코드 포맷 지정: Shift + Alt + F
  - 선택 영역 코드 포맷 지정: Ctrl + K, Ctrl + F

- VS Code 코딩 포맷 설정하기
  - 메뉴에서 [파일] → [기본 설정] → [설정] 선택
  - "C_cpp: Clang_format_fallback style" 항목 선택
  - 기본값은 Visual Studio이며 원하는 형태로 수정 가능
    - 참고 URL: https://clang.llvm.org/docs/ClangFormatStyleOptions.html
    - Example:

```
{ BasedOnStyle: LLVM, UseTab: Always, IndentWidth: 4, TabWidth: 4, BreakBeforeBraces: Linux,
AllowShortIfStatementsOnASingleLine: false, IndentCaseLabels: false, ColumnLimit: 0,
AccessModifierOffset: -4 }
```

1. 자료 구조와 알고리즘은 처음이지?

4) 시간 복잡도

Presented by sunkyoo.hwang@gmail.com

알고리즘 성능 분석

- 주어진 문제를 '잘' 해결하는 '좋은' 알고리즘인지를 판단하려면?
  - '잘' 해결하는지를 판단하려면?
    - 정확성 테스트
    - 수학적 귀납법
    - 다양한 테스트 케이스 사용
  - '좋은' 알고리즘인지 판단하려면?
    - 효율성 테스트
    - 알고리즘의 자원(resource) 사용량을 분석: 실행 시간, 메모리 사용량, 통신 용량 등
    - 보통 실행 시간에 대한 분석만 다룸
    - 대용량 입력 데이터에 대한 처리 능력 판단

시간 복잡도

- 시간 복잡도(time complexity)
  - 입력의 크기가 커짐에 따라 연산 시간(연산 횟수)이 얼마나 증가하는지를 근사적으로 표현
    - 연산의 실행 횟수를 입력 데이터의 크기(n)에 관한 함수 형태로 표현
    - 알고리즘을 직접 구현하지 않고도 알고리즘의 효율성을 가늠할 수 있음
  - 실행 시간 측정 방법의 한계: 알고리즘을 구현한 프로그램의 실행 시간은 실행 환경(하드웨어, 운영체제, 언어, 컴파일러 등)에 따라 달라지므로 절대적인 실행 시간 비교는 성능 파악에 한계가 있음

- 빅오 표기법(big-O notation): $O(f(n))$
  - 대표적인 시간 복잡도 표현 방법
  - 연산 횟수를 구체적인 수식으로 표현하지 않고, 최고차항의 차수만으로 표현

    (e.g.) $4n^2 + 2n + 5 \rightarrow O(n^2)$
  - 점근 표기법(asymptotic notation)의 한 방법으로 실행 시간의 상한(upper bound)을 표현

시간 복잡도

- 대표적인 빅오 표기법의 예

빅오 표기법	이름	설명
$O(1)$	상수 함수(constant function)	입력의 크기와 무관하게 일정한 시간 소요
$O(\log n)$	로그 함수(logarithmic function)	입력의 크기의 로그에 비례하는 시간 소요. 대체로 매 단계마다 입력의 크기를 절반으로 줄여가는 형태로 동작.
$O(n)$	선형 함수(linear function)	입력의 크기에 비례하는 시간 소요
$O(n \log n)$	로그-선형 함수(log-linear function)	대부분의 효율적인 정렬 알고리즘의 시간 복잡도 (e.g.) 퀵 정렬
$O(n^2)$	이차 함수(quadratic function)	보통 2중으로 중첩된 반복문을 사용 (e.g.) 버블 정렬
$O(n^3)$	삼차 함수(cubic function)	보통 3중으로 중첩된 반복문을 사용
$O(2^n)$	지수 함수(exponential function)	(e.g.) 부분집합 구하기
$O(n!)$	팩토리얼 함수(factorial function)	(e.g.) 순열 만들기

시간 복잡도 예제

- 예제 #1)

```
int middle(int data[], int n)
{
    return data[n / 2];
}
```

n 값에 관계없이 1회 연산이 실행: $O(1)$

- 예제 #2)

```
int summation(int data[], int n)
{
    int sum = 0;

    for (int i = 0; i < n; i++)
        sum = sum + data[i];

    return sum;
}
```

for 반복문 내부 문장이 n 번 실행: $O(n)$

Presented by sunkyoo.hwang@gmail.com

시간 복잡도 예제

- 예제 #3)

```
int variance(int data[], int n)
{
    int sum = 0;

    for (int i = 0; i < n; i++)
        sum = sum + data[i];

    double mean = (double)sum / n;
    double var = 0.0;

    for (int i = 0; i < n; i++)
        var += (data[i] - mean) * (data[i] - mean);

    return (var / n);
}
```

for 반복문 내부 문장이 $2n$ 번 실행: $O(n)$

Presented by sunkyoo.hwang@gmail.com

시간 복잡도 예제

- 예제 #4)

```
int find(int data[], int n, int target)
{
    for (int i = 0; i < n; i++)
        if (data[i] == target)
            return i;

    return -1;
}
```

→ 최선의 경우 for 반복문 안의 비교 연산이 1회, 최악의 경우 n 번 실행: $O(n)$

- 예제 #5)

```
void bubble_sort(int data[], int n)
{
    for (int i = n - 1; i > 0; i--)
        for (int j = 0; j < i; j++)
            if (data[j] > data[j + 1])
                swap(data[j], data[j + 1]);
}
```

→ 이중 for 반복문 내부 문장이 $\frac{1}{2}n(n-1)$ 번 실행: $O(n^2)$

Presented by sunkyoo.hwang@gmail.com

시간 복잡도 예제

- 예제 #6)

```
int sum(int n)
{
    if (n == 1)
        return 1;

    return n + sum(n - 1);
}
```

재귀 함수가 n번 호출되고, 함수 내부에 단일 연산만 있음: $O(n)$

- 예제 #7)

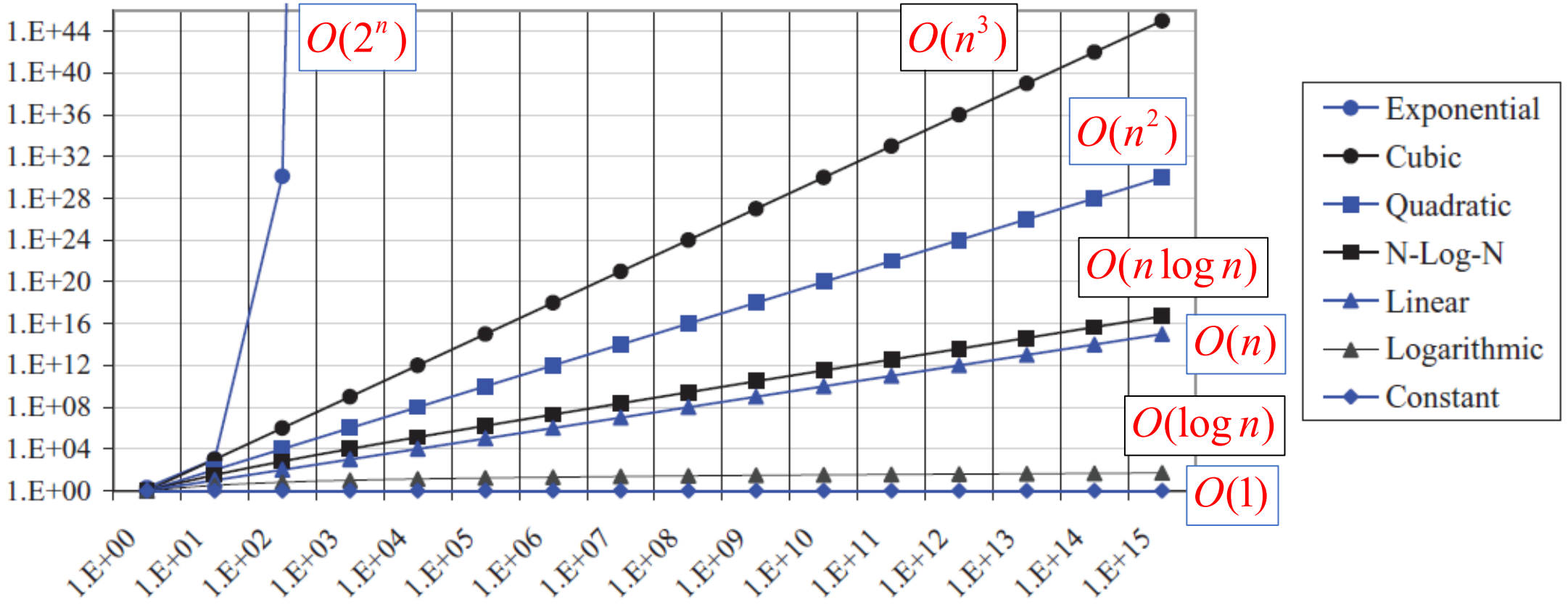
```
void func(int n)
{
    if (n == 1) return;

    func(n - 1);
    func(n - 1);
}
```

재귀 함수가 매번 2번의 새로운 호출을 수행함.
func(n)이 1번 호출되면, func(n-1)은 2번 호출되고,
func(n-2)는 4번, func(n-3)은 8번 등등으로 호출됨:
$O(2^n)$

Presented by sunkyoo.hwang@gmail.com

시간 복잡도 비교

- 주요 시간 복잡도 증가율 비교



Michael T. Goodrich, et. al., "Data Structures and Algorithms in C++," 2nd Edition, Wiley, 2011.