프로그래밍 언어론

프로그래밍언어 평가 기준

컴퓨터공학과

이만호

충남대학교
CHUNGNAM NATIONAL UNIVERSITY

프로그래밍언어 평가 기준

학습목표

프로그래밍언어를 평가하는 기준이 무엇이며,
이들 기준에 따라 프로그래밍언어들이 어떻게
평가되는지 학습한다.

학습내용

프로그래밍언어의 주요 평가 기준

- Readability(가독성)
- Writability(작성력)
- Reliability(신뢰성)
- Cost(비용)

목 차

알고가기

다음 중에서 프로그래밍언어에 대해서 틀리게 기술하고 있는 것은?

(a) 널리 사용되는 프로그래밍언어는 매우 우수한 언어이다.

(b) 프로그램 작성에 필요한 필수적인 기능만 지원하는 프로그래밍언어로 작성된 프로그램은 이해하기가 용이하다.

(c) 프로그래밍언어가 다양한 data type을 지원하면 프로그램 작성하기가 용이하다.

(d) 프로그램에서 오류가 발생할 가능성을 많이 지적해주는 프로그래밍언어는 신뢰도가 높다.

| 프로그래밍언어 평가 기준

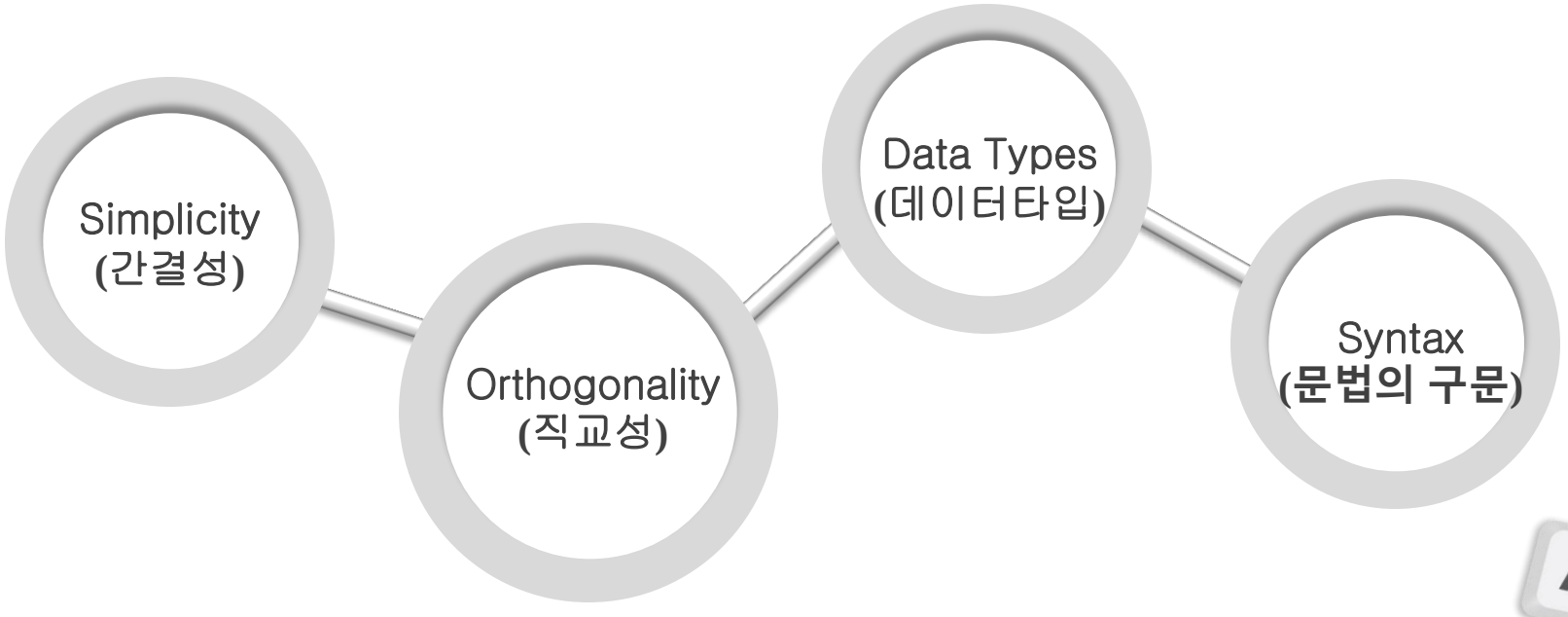
Readability(가독성)	이미 작성된 프로그램을 잘 이해할 수 있는가?
Writability(작성력)	새 프로그램을 작성할 때 쉽게 작성할 수 있는가?
Reliability(신뢰성)	모든 조건에서 프로그램이 정확하게 실행되는가?
Cost(비용)	프로그램 개발 및 유지보수에 시간과 경비가 얼마나 필요한가?

Readability(가독성)

이미 작성된 프로그램을 이해할 수 있는 정도

- Debugging과 프로그램 Maintenance(유지보수)에 readability가 중요함

Readability에 영향을 끼치는 요인들



Readability(가독성)

Simplicity(간결성)

- **너무 많은 종류의 문장을 지원하는 언어는 readability에 좋지 않다.**
  - > 언어가 지원하는 모든 종류의 문장을 잘 알고 사용하기 어렵다.
  - > 보통 프로그래머마다 주로 사용하는 문장이 다르다.
- **동일한 기능의 연산 표현 방법이 많을수록 좋지 않다.**

예 변수 k의 값을 1만큼 증가하기

```
k++,   ++k,   k+=1,   k=k+1
```

- **Operator Overloading**
  - > 하나의 operator(연산자)에 여러 type의 operands(피연산자)를 사용할 수 있다.
  - > 연산자 overloading은 프로그램을 이해하는데 좋지 않은 기능이다.

예

```
int x, y;   float n, m;   x + y, n + m
```

- **함수형언어 (functional language)가 간결성에서 우수함**

Readability(가독성)

Orthogonality(직교성) - 1

언어의 구성 요소를 임의로 조합해서 프로그램을 작성할 수 있는 기능

> 구성 요소들이 의미적으로 서로 독립적이어야 함

※ (수학) 직교하는 x-축과 y-축으로 이루어진 2차원 평면의 모든 점은 x-성분과 y-성분의 조합으로 표현된다.

예1 m과 n의 최대값에 3을 곱한 값을 변수 max3에 저장하기

```
max3 = 3 * (if (m>n) m else n); // C 언어는 지원하지 않음
```

예2 C 언어에서 변수 선언하기

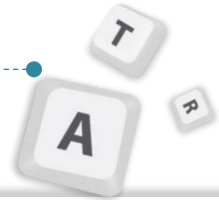
```
{auto,static} {register} {unsigned} {int,float,… } var;
```

예3 어셈블리어의 instruction (op-code operand-1,operand-2)

```
ADD  R1,R2        ADD  R1,SCORE
ADDL R1,R2        ADDL R1,SCORE
ADDF R1,R2        ADDF R1,SCORE
```

비교적 소수의 구성 요소들을 비교적 소수의 방법으로 조합해도 어떤 표현이든지 가능하다.

> 간결성 조건 충족



Readability(가독성)

Orthogonality(직교성) - 2

- **Orthogonality가 부족한 언어에서는 구성 요소의 사용 규칙에 예외적인 경우가 많다.**

예1 C 언어에서 사용자 정의 data type인 배열(array)과 구조체(struct)
구조체는 function의 결과로 return할 수 있으나, 배열은 return할 수 없음

예2 C 언어에서 수식 "a + b"
a와 b의 type(int, float, pointer, …)에 따라 연산 방법과 결과가 다름

- **완벽에 가까운** 직교성을 지원하는 언어는 문제를 유발하기도 한다.
  - > Orthogonality가 readability에 매우 좋은 특성이다.
  - > 상식적이지 않은 조합은 readability를 저하시킬 수도 있다.

예 (a>b)이면 "x = 5"를 실행하고, 아니면 "y = 5"를 실행

```
(Algol): (if (a>b) x else y) = 5;   // C 언어는 지원하지 않음
```

- **함수형 언어가 orthogonality 면에서 우수함.**

Readability(가독성)

Data types

- 적절한 data type이 지원되어야 readability에 좋다.

  > C 언어는 bool 타입을 지원하지 않아 readability가 좋지 않다.

  예 bool 타입을 지원하는 언어 : `tag = true;`
  C 언어 : `tag = 1; (int 또는 bool?)`

- 다양한 data type이 지원되어야 한다.

  > integer, floating-point, character, ...

- 다양한 자료구조(data structure)를 data type으로 정의할 수 있어야 한다.

  > array, record, union, ...

Readability(가독성)

Syntax(문법의 구문)

- 문법에서 구문 형태를 정의하므로 readability에 영향을 끼친다.
- identifier(식별자)의 형태
  - > 길이에 제한이 있는 경우 readability가 떨어진다.

```
A, A3, Avrge, AverageOfScore
```

  - > 특수 문자를 identifier에 사용가능하면 readability가 좋다.

```
AverageOfScore, average_of_score
```

- 특수 단어(Special words)
  - > 의미를 자연스럽게 알 수 있는 특수 단어들은 readability에 좋다.

```
if, while, for, then, else, begin, end, class, ……
```

- 형식과 의미(Form and meaning)
  - > 표현 형식은 같거나 비슷한데 경우에 따라 다른 의미를 가지면 readability가 떨어진다.

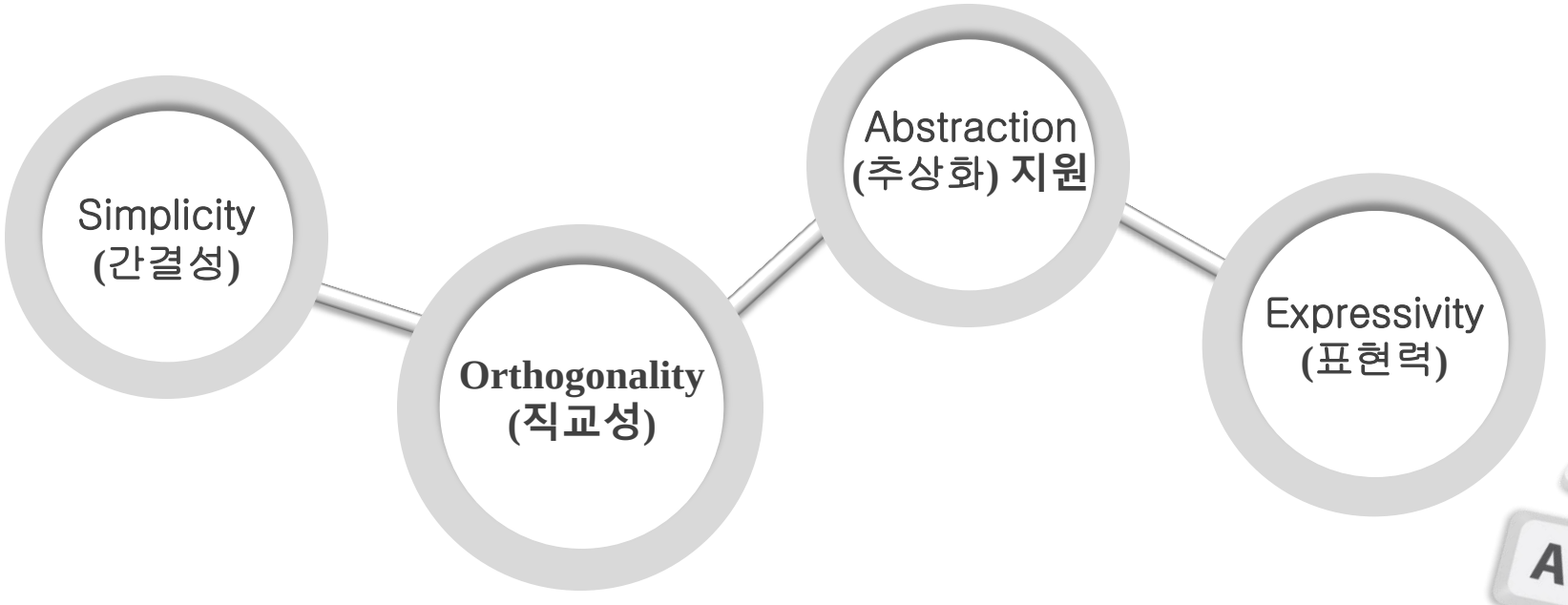
예

```
static int gvar;                                  // "static"의
int foo(...) { static int lvar ... })             //   의미가 다름
```

Writability(작성력)

새 프로그램 작성의 용이성

Writability에 영향을 끼치는 요인들



Writability(작성력)

Simplicity(간결성)

- 지원되는 구문 구조가 간결할수록 writability에 좋다.

- 지원되는 구문 구조가 많은 경우의 writability 저하 요인
  - > 프로그래머가 많은 구문 구조의 기능을 모두 이해하기 어렵다.
  - > 자주 사용하지 않는 구문 구조를 잘못 사용할 수 있다.
  - > 효율적이고 명쾌한 구문 구조를 몰라서 사용하지 않을 수 있다.

- 소수의 구문 구조를 조합하여 사용하는 것이 좋다.
  - > Orthogonality(직교성)와 관계됨

Writability(작성력)

Orthogonality(직교성)

- **Orthogonality가 지원되는 프로그래밍언어는 writability이 우수하다.**
  - > 간결한 구문 구조만 지원되어도 프로그램 작성이 불편하지 않다. (간결성과 관계됨)
  - > 모든 구성 요소들을 임의로 조합해서 사용할 수 있다.

- **완벽에 가까운 orthogonality를 지원하는 언어는 readability가 떨어질 수 있다.**
  - > ("Readability – Orthogonality" 참조)

Writability(작성력)

Abstraction(추상화) 지원

Abstraction(추상화)

> 복잡한 자료구조(data structure)나 operation(연산)을 간단히 사용할 수 있도록 정의하는 것.
> 자료구조나 연산의 구현의 세부 사항은 감춰진다.
> 오늘날 대부분의 프로그래밍언어 설계의 중요 개념이다.

종류) Data abstraction, Process abstraction

Data abstraction

> 복잡한 자료구조(data structure)를 정의해서 사용할 수 있다.

예 stack, tree, graph, complex number, ……

Process abstraction

> 복잡한 연산(operation)을 정의해서 사용할 수 있다.

예 function – `factorial(n), max(a[],n), …`
procedure – `sort(a[],n), …`

Writability(작성력)

Expressivity(표현력)

Expressivity

> 기존의 구성 요소와 동일한 기능을 수행하지만 좀 더 편리한 구문 지원

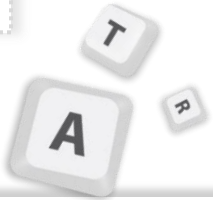
예
`switch` : 중첩된 `if-then-else`
`for` : counter를 사용하는 while
`k++, ++k, k+=1, k=k+1` : 1을 증가시키는 다양한 방법

APL 언어는 매우 강력한 연산자를 지원하여 expressivity가 좋다.

> 행렬 더하기, 곱하기, 전치행렬 구하기 등이 하나의 연산으로 가능

> Writability는 매우 뛰어나나, readability가 매우 떨어짐.

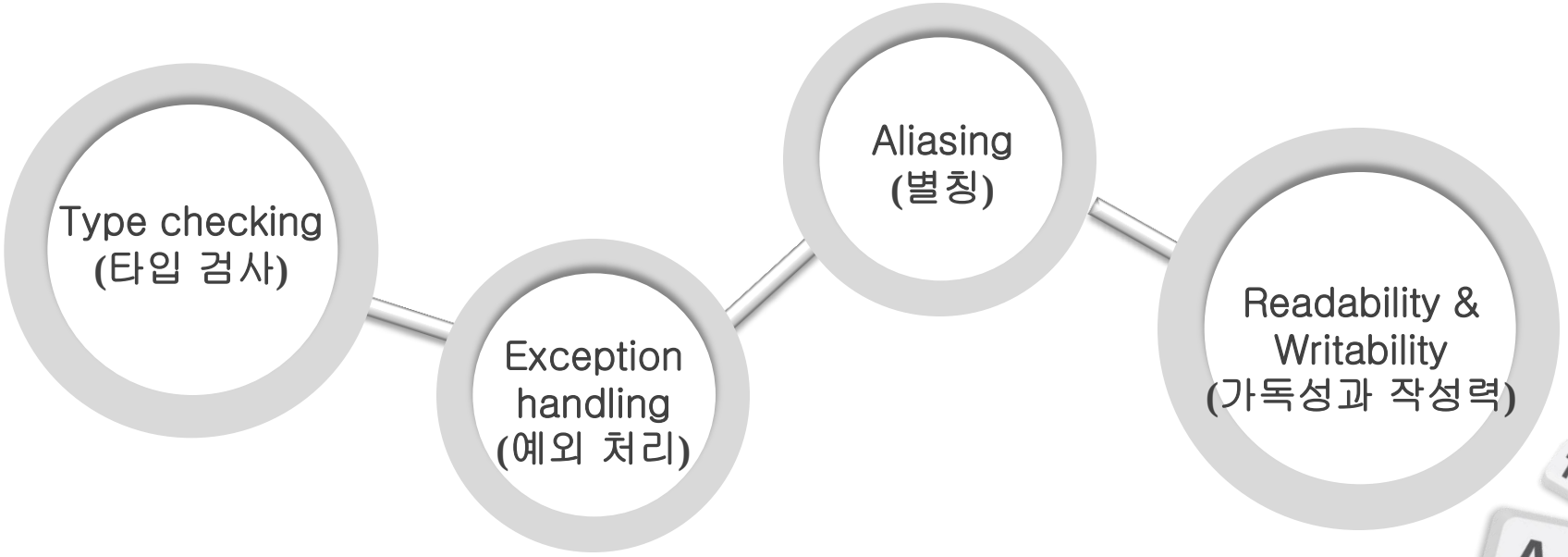
예
행렬 A와 B 곱하기 : `A +.x B`



Reliability(신뢰성)

모든 조건에서 프로그램이 정확하게 실행되는 정도

Reliability에 영향을 끼치는 요인들



Reliability(신뢰성)

Type checking (타입 검사)

Type checking

> Reliability에 영향을 주는 매우 중요한 요소이다.

> Compile할 때나 프로그램 실행 시 type error를 검출한다.

예

```
foo(float x) { ... }
main() { int k; ... call foo(k); ... }
```

Type error는 일찍 발견할수록 오류 수정 비용이 적다.

> Type checking은 compile할 때 하는 것이 바람직하다.

> 프로그램 실행 시 이루어지는 type checking은 비용이 크다.

Reliability(신뢰성)

Exception handling (예외 처리)

Exception handling (예외 처리)

> 프로그램 실행시 run-time error가 발생한 경우, 이를 인식하고 오류를 처리할 수 있는 필요한 조치를 취할 수 있는 기능

예
C++, Java, Ada: 예외 처리 기능을 지원함
C, Fortran: 예외 처리 기능을 지원하지 않음

Reliability 향상에 긍정적인 영향을 끼친다.

Reliability(신뢰성)

Aliasing(별칭)

Aliasing(별칭)

> 메모리의 한 저장 장소를 여러 이름으로 접근할 수 있는 기능

예

```
(C 언어)
union { int n; float x; } foo;
foo.n
foo.x
```

Aliasing은 매우 조심해서 사용해야 한다.

> Reliability에 부정적인 영향을 끼친다.

Reliability을 높이기 위해 aliasing 기능을 제약할 수 있다.

Reliability(신뢰성)

Readability와 Writability

Readability와 Reliability

> 이해하기 쉬운 프로그램은 오류 가능성이 적다.

Writability와 Reliability

> 프로그램을 쉽게 작성할 수 있다면 오류 가능성이 적어진다.

> 알고리즘을 자연스러운 방법으로 프로그램을 작성할 수 없는 언어는 자연스럽지 않은 방법으로 프로그램을 작성해야 하므로 오류 가능성이 크게 된다.

Cost(비용)

Cost(비용)에 영향을 끼치는 요인

- **프로그래머 양성 비용**
  - > Simplicity, Orthogonality, 프로그래머의 경험의 영향을 받음
- **프로그램 개발 비용**
  - > Writability와 응용분야와의 적합성의 영향을 받음
- **프로그램을 compile/interprete하는 비용**
- **프로그램의 실행 비용**
  - > 언어 설계의 영향을 받음: type checking 방법 등
- **Compiler/Interpreter 비용**
  - > 무료 또는 저렴한 경우 널리 사용됨 (Java)
  - > 고가인 경우 잘 사용되지 않음 (Ada)
- **Reliability**
  - > Reliability가 낮은 언어로 작성된 프로그램은 오류로 인한 비용 부담이 매우 크다.
- **유지보수(maintenance) 비용**
  - > 비교적 오래 사용되는 대형 software system의 뷰지보수 비용은 개발 비용의 2~4배에 달함
  - > Readability와 writability의 영향을 받음

| 평가 기준 사이의 충돌(trade-off) 관계

Reliability vs. 실행비용

- Type checking을 철저히 하면 Reliability 향상, 실행비용 증가
- 배열의 index checking을 하면 Reliability 향상, 실행비용 증가

```
int  a[5];  ...  k=8;  ...  a[k]  ...
```

Readability vs. Writability

- APL 언어는 새로운 symbol을 사용하여 매우 강력한 연산자를 제공한다.
  (Writability 향상, Readability 저하)

행렬 A와 B 곱하기 : `A +.x B`

Writability vs. Reliability

- Pointer의 사용은 유연한 기억장소 접근이 가능하여, writability 향상, reliability 저하.
- Variant record(가변 레코드)의 사용은, writability 향상, reliability 저하.

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 다음 중에서 readability 면에서 가장 좋다고 평가할 수 있는 언어는?
   (a) 언어의 구성 요소를 임의로 조합해서 프로그램을 작성할 수 있는 언어
   (b) 논리형(bool type)을 표현하는데 정수형을 이용하는 언어
   (c) k++, ++k, k+=1, k=k+1과 같이 다양한 방법으로 k의 값을 증가 시킬 수 있는언어
   (d) Operator overloading을 지원하는 언어

평가하기 2

2. 다음 중에서 reliability 면에서 가장 좋다고 평가할 수 있는 언어는?

(a) 하나의 기억 공간을 여러 변수 이름으로 사용할 수 있는 언어

(b) Type checking을 철저히 수행하는 언어

(c) 프로그래머가 pointer를 조작할 수 있는 언어

(d) 다른 사람이 작성한 프로그램을 이해하기 어려운 언어

정리하기

프로그래밍언어 평가 기준

Readability, Writability, Reliability, Cost

프로그래밍언어 평가 기준에 영향을 끼치는 요인

Simplicity(간결성), Orthogonality(직교성), Data type,

Syntax(문법의 구문), Abstraction(추상화),

Expressivity(표현력), Type checking,

Exception handling(예외처리), Aliasing(별칭)

주안점

각각의 프로그래밍언어 평가 기준에 어떤 요인들이 어떻게 영향을 끼치는가?