プログラミング言語 0
イントロ・講義予定
Programming Languages 0
Introduction and Lecture Plan

田浦

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能)
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出

- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出 ← データ「型」のチェック
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出 ← データ「型」のチェック
    - 実行しながら
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出 ← データ「型」のチェック
    - 実行しながら
    - 実行前に ←「静的型検査」

- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出 ← データ「型」のチェック
    - 実行しながら
    - 実行前に ←「静的型検査」
  - メモリ管理を自動化 ← ゴミ集め
- 性能：プログラムが速く動く

ロードマップ

- 生産性：正しいプログラムが簡単に書ける
  - わかりやすく書ける ← 「宣言的」に書ける ← 関数型プログラミング ← OCaml
  - 短く書ける ← 再利用性が高い ← 多相性 (同じコードが色々な種類のデータに適用可能) ← 関数型, オブジェクト指向プログラミング ← Python
  - 間違いを検出 ← 不当な操作を検出 ← データ「型」のチェック
    - 実行しながら
    - 実行前に ←「静的型検査」
  - メモリ管理を自動化 ← ゴミ集め
- 性能：プログラムが速く動く
  - インタープリタ, コンパイラ

Roadmap

- Productivity : can write correct programs easily

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively"

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data)

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations ← check data "types"

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations ← check data "types"
    - at runtime

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations ← check data "types"
    - at runtime
    - before execution ← "static type check"

- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations ← check data "types"
    - at runtime
    - before execution ← "static type check"
  - automate memory management ← garbage collection
- Performance : programs run fast

Roadmap

- Productivity : can write correct programs easily
  - can write intuitively ← can write "declaratively" ← functional programming ← OCaml
  - can write concisely ← can write reusable code ← polymorphism (the same code can apply to various types of data) ← functional, object-oriented programming ← Python
  - detect errors ← detect invalid operations ← check data "types"
    - at runtime
    - before execution ← "static type check"
  - automate memory management ← garbage collection
- Performance : programs run fast
  - interpreter, compiler

理解目標

- 「安全な言語」と「そうでない言語 (C, C++, Fortran など)」の違いを学ぶ.
- 安全な言語の設計に対するアプローチ，特に「動的型検査」 vs. 「静的型検査」
- 高級言語を使ってみる
  - 関数型 OCaml, オブジェクト指向 Python
- 言語を安全にするもうひとつの重要技術：自動メモリ管理（ゴミ集め）
- （安全でない）C 言語をだいぶ安全にするツール Valgrind, 保守的ゴミ集めを使ってみる
- 言語処理系を作るツール (字句・構文解析器の生成器) を使ってみる
- 単純なコンパイラを作る

Objectives

- learn differences between "safe languages" and "unsafe languages (e.g., C, C++, Fortran, etc.)"
- approaches to desigining safe languages. In particular, dynamic type checking vs. static type checking
- try a few high level programming languages
  - functional OCaml, object-oriented Python
- another critical technology to make languages safe: automatic memory management (garbage collection)
- try Valgrind and conservative garbage collector, tools to make (otherwise unsafe) C/C++ languages a lot safer
- try compiler construction toolkit (lexical analyzer generator, parser generator)
- build a simple compiler

授業の形式

- 講義: 基本概念の説明
- 演習, 小課題: 簡単なことを定着
- 発表: または 課題:
  - 発表: 一部の人が基本概念を踏まえた上で，応用として論文を購読し発表．題材は HP を参照．4 月中くらいに発表を宣言してもらう．ラスト数回のどこかで発表．
  - 課題: 発表しない人は最終レポートを提出

Format

- lecture: basic concepts
- small exercises and assingments: to understand basics
- presentation or a report
  - presentation: in the last class (or perhaps two), volunteers read papers and talk about them instead of the final assignment (make sure you discuss details with me in advance)
  - the final report:

成績のつけかた

- 小課題をまともに提出
- 発表 もしくは 最終レポート
- 試験はしない

Evaluation

- submit enough number of small assignments
- presentation or the final assignment
- no exams