

プログラム言語論

亀山幸義

筑波大学コンピュータサイエンス専攻

筑波大学 情報科学類 講義

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 1 / 7

静的 vs 動的

- 宣言された変数はすべて、式の中にあられるかどうか。
- 宣言された変数はすべて、プログラム中で使われるかどうか。
- 実行中に使用されるスタックの量。
- MiniML 言語のプログラム (ペアを表すデータ構造を持つ) に対して、実行中に使用されるヒープの量。
- MiniC 言語や MiniML 言語のプログラムに対して、プログラムの型が整合的であるかどうか。

C, ML では、1,5 は静的に決められる。2,3,4 は決められない(実行時の情報)。

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 3 / 7

静的型付け vs 動的型付け

- Java, ML, C は静的型付け: コンパイル時にプログラムの型の整合性を検査 (あるいは推論) する。実行時には (ほとんど) 型の整合性を検査しないですむ。
- Ruby, Perl, Python, Lisp, Scheme などは、動的型付け: コンパイル時には型の整合性はチェックしない。実行時に型が整合しない演算を行えば、エラーを発生させる。

これらの帰結:

- 前者は、より高い信頼性を得やすい。
- 後者は、より柔軟なプログラミングが可能。

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 5 / 7

Quiz

質問: 今日の授業では、Java と Ruby の言語の特徴を学んだ。それらを踏まえ、具体的なプログラムを1つ想定して、どちらの言語で記述するのがよいか、自分の考えを述べなさい。

来週までの宿題: 授業ホームページの指示にしたがって、Java と Ruby のサンプルプログラムを走らせ、継承、サブタイピング、動的ルックアップ、および Java の overload/override について理解を深めなさい。(サンプルプログラムは、coins マシンの `~kam/java` および `~kam/ruby` にある。)
[宿題については、レポート提出は求めないが、各自、必ず学習しておくこと。]

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 7 / 7

Java

- C/C++ 言語に似た構文
- ブロック構造言語; 変数は静的束縛
- 静的な型システム、型安全性
- 信頼性、安全性を重視

Ruby

- スクリプト言語
- 動的言語 (静的型付けをしない)
- クラス定義は宣言ではなく実行文
- 開発効率を重視

共通すること: オブジェクト指向であること (動的ルックアップ、情報隠蔽、継承、サブタイピング)

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 2 / 7

静的 vs 動的

コンパイラとインタープリタの差 (再訪問)

- 多くのコンパイラは、プログラム全体 (あるいはプログラムの1つのモジュール全体) を読みこみ、可能な限り多くの静的情報を取得して、それを生かした効率良いコードを生成する。
- 多くのインタープリタは、そのようなことをしない。

静的情報を使って効率がよくなる例:

- 局所変数が、スタックフレーム中のどこにあるかを解析すれば、実行時には、「局所変数がどこにあるかを探す」必要がなくなる。
- 「a+b」という式で、a,b が整数型とわかっていれば、整数型かどうかのチェックはいらす、いきなり機械語の「加算」命令を使える。

現在の Optimizing Compiler は、上記以外にも様々な効率化をはかっている。

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 4 / 7

今日の課題

- Java のサブタイピングと、動的ルックアップの基礎を理解する。
- Ruby を Java と比較して、動的型付け言語について考察する。

亀山幸義 (筑波大学コンピュータサイエンス専攻) プログラム言語論 筑波大学 情報科学類 講義 6 / 7