

# プログラミング言語論 第1回

イントロダクション

□担当：福嶋 慶繁（ふくしま のりしげ）

□連絡先

□E-mail: [evh41111@ict.nitech.ac.jp](mailto:evh41111@ict.nitech.ac.jp)

・学生メールで福嶋慶繁で検索して出てくるアドレス

□場所

□講義室：0211（予定）

□CSE：101A（予定）正門に一番近い部屋

- 講義とプログラミングの演習が半々くらい
- レポートで評価
  - 小テストは毎回出席を兼ねる簡単なもの
    - ・ 出欠確認用
  - 課題はそれなりに時間がかかるもの
    - ・ 課題はトピックごとに出すため毎回ではありません
- 生成AIについて
  - ChatGPT：月額3万円のはレギュレーション規定かけていいですか？
  - レポートに生成AIを使った場合は、ツールとモデルを書いて下さい。

□資料： Moodleにアップロード、web 上のは適宜指示

□教科書：なし

□参考書：

□**コーディングを支える技術 技術評論社**

□プログラミング言語の概念と構造 ピアソンエディケーション

□リーダブルコード オライリージャパン

□コードコンプリート 日経BP

□オブジェクト指向における再利用のための  
デザインパターン ソフトバンククリエイティブ

□プログラミング言語論 コロナ社

□すごいHaskellたのしく学ぼう! オーム社

□プログラミングHaskell オーム社

# 日程（あくまでも予定日・程調整前）

5

	内容
1	イントロダクション
2	構造化プログラミング 1
3	構造化プログラミング 2
4	構造化プログラミング 3
5	オブジェクト指向言語 1 (C++)
6	オブジェクト指向言語 2 (C++)
7	オブジェクト指向言語 3 (C++)
8	オブジェクト指向言語 4 (C++)

	内容
9	関数型言語 (Haskell) 1
10	関数型言語 (Haskell) 1
11	関数型言語 (Haskell) 2
12	ジェネリックプログラミング
13	高能率計算プログラミング
14	ドメイン固有言語(DSL)
15	ドメイン固有言語(DSL)
16	予備

日付等はMoodle参照。 完全オンデマンドだけの日もあります。

# プログラミング言語とは？(1/2)

6

高級言語（人間向きの言語）

```
int main()
```

```
{
```

```
    printf("hello¥n");
```

```
}
```



コンパイル

機械語

0000D80	1C E2 A0 7A 62 FC 9C EB	9E A7 74 7F DD E9 61 48
0000D90	27 A1 E2 BA 7A 3A AA 5D	F3 39 3F A6 07 55 5C D3
0000DA0	98 0C A9 61 24 BE ED 3E	31 EF AF 0C D5 0E 79 FB
0000DB0	DD 3A 05 29 DD 09 DD 3A	B1 DE 2F 9B 7F 01 00 00
0000DC0	FF FF 03 00 50 4B 03 04	14 00 06 00 08 00 00 00
0000DD0	21 00 40 2A BB 11 41 01	00 00 63 06 00 00 21 00
0000DE0	00 00 70 70 74 2F 73 6C	69 64 65 73 2F 5F 72 65
0000DF0	6C 73 2F 73 6C 69 64 65	31 38 2E 78 6D 6C 2E 72
0000E00	65 6C 73 BC 95 CD 4A 03	31 10 80 EF 82 EF B0 E4
0000E10	6E D2 D6 5A 7F E8 B6 17	15 0A 7A 91 FA 00 B3 BB
0000E20	B3 BB A1 49 26 24 A9 ED	F6 E9 8D A8 D8 42 A5 B2
0000E30	48 8E 49 26 DF 7C CC 64	C8 74 BE D5 2A 7B 43 E7
0000E40	25 99 9C 0D F9 80 65 68	4A AA A4 69 72 F6 BA 7C
0000E50	BC B8 61 99 0F 60 2A 50	64 30 67 1D 7A 36 9F 9D























人が計算機に命令を出すための言語



- コンパイルして落ち着く先はどんな言語も機械語
- 言い換えれば，機械語が出ればどんな書き方でもOK
- どうして，そんなにいっぱい書き方があるの？
- 新しい言語がどうしてそんなに出てきているの？

□ 今使われている言語ランキング

□ <https://www.tiobe.com/tiobe-index/>

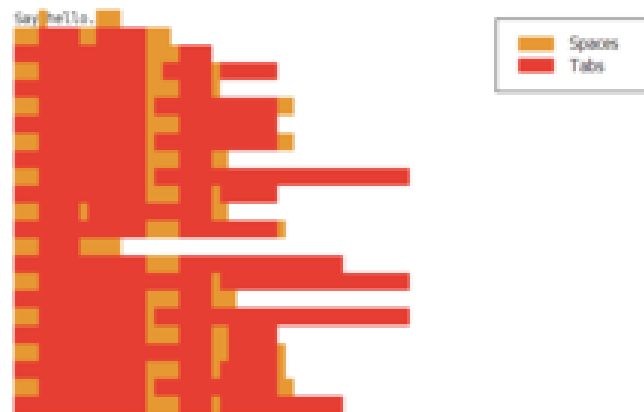
Apr 2022	Apr 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	13.92%	+2.88%
2	1	▼	 C	12.71%	-1.61%
3	2	▼	 Java	10.82%	-0.41%
4	4		 C++	8.28%	+1.14%
5	5		 C#	6.82%	+1.91%
6	6		 Visual Basic	5.40%	+0.85%
7	7		 JavaScript	2.41%	-0.03%
8	8		 Assembly language	2.35%	+0.03%
9	10	▲	 SQL	2.28%	+0.45%
10	9	▼	 PHP	1.64%	-0.19%
11	16	▲	 R	1.55%	+0.44%
12	12		 Delphi/Object Pascal	1.18%	-0.29%
13	14	▲	 Go	1.09%	-0.14%
14	15	▲	 Swift	1.00%	-0.19%
15	13	▼	 Ruby	0.88%	-0.35%
16	11	▼	 Classic Visual Basic	0.83%	-0.71%
17	23	▲	 Objective-C	0.82%	+0.15%
18	18		 Perl	0.79%	-0.21%
19	37	▲	 Lua	0.78%	+0.45%
20	19	▼	 MATLAB	0.74%	-0.25%



- Kotlin 2011
  - Javaの拡張
- Swift 2014
  - Objective-Cの拡張
- Go 2009
  - C言語の拡張
- Rust 2010
  - C++の置き換えを

## □Whitespace

- タブ, スペース, 改行だけで書く言語
- スペースが0, タブが1, 改行が終端記号
- 2進数が表現できればなんでも書ける
  - スペーススペース : 加算
  - スペースタブ : 引き算
  - スペース改行 : 掛け算
  - タブスペース : 割り算



- 記述が短い
  - 速く動く
  - バグが少ない
  - 読みやすい
  - 改変しやすい
  - 学習しやすい
  - 並列性が高い
  - いろんな環境で動く
- 言語がほしい（無理）

# 本講義の目的

## □ 達成目標

□ **計算のパラダイム**と、対応するプログラミング言語の仕組みを説明できる.

□ **「手続き型」**、**「オブジェクト指向型」**、**「関数型」**言語のベースとなる基本概念を理解し、具体的に説明できる

□ **型**、**関数**、**名前**や**スコープ**、**ジェネリクス**等の様々なプログラミング言語のアイディアの、意味と役割を説明できる.

- 複数のプログラミング言語に触れることで、様々なプログラミング言語の概念とパラダイムを理解する
  - C, C++, Haskell, Java, Python
- 特に抽象度の高い概念を理解することを目指す
  - スコープ、名前空間
  - 継承、カプセル化、ポリモーフィズム
  - コンストラクタ・デストラクタ
  - インターフェース
  - 高階関数・ラムダ式
  - 静的型付け・動的型付け
  - 型推論
  - ジェネリックプログラミング, テンプレート, 型変数

# Q.プログラミング言語、複数勉強する 必要あるの？ 14

- A1. あります。複数の言語を見て、はじめて理解できることがあります。
- A2. 主流となるプログラミングパラダイムは時を経て移り変わってきています。（機械語→アセンブラ→手続き型→オブジェクト指向型）複数のパラダイムを理解しておくことで、「変化に強いプログラミングスキル」を身につけることができます。
- A3. 「履歴書を強くする」ことができます。

**プログラミング言語の歴史に関する動画を見て感想を書きなさい**

**□**提出はオンラインテキストで.

「ハッカーになろう」, Eric S. Raymond, 山形浩生訳

<https://cruel.org/freeware/hacker.html>

を読んで以下について答えなさい.

- ここでいう「ハッカー」は悪い意味のハッカーではなく原義のハッカー(≒熟練プログラマ)の意 (著者は区別のため, システム侵入などをするものを「クラッカー」と呼んでいる).
- [レポート1-1] この文書の「プログラミングを身につけること」を読み, 要点を簡潔にまとめよ.
- [レポート1-2] 他の部分で共感できる部分について, 「自分を取り組めそうなこと」について述べよ.

※課題の提出は3, 4回先の講義時に説明しますが, **レポートを溜めないこと**を強く推奨します



「この **5 つすべて** (Python, Java, C/C++, Perl, LISP) を勉強しておくのがいちばんいいのです。これらはもっとも重要なハッキング用言語だというだけでなく、それぞれプログラミングに対してまったく違ったアプローチをしているので、どれも非常に有益な勉強となるでしょう。」

# まとめ

## □イントロ

## □本講義の目的・目標

### □パラダイムの理解

- 構造化プログラミング
- オブジェクト指向
- 関数型言語

### □いろいろな概念の理解

- 型, 関数, 名前・スコープ, ジェネリクス