M-project

プログラミング言語あれこれ

なぜこんなにいろんな種類があるの？

1

M-project

プログラミング言語あれこれ（目次1）

2

M-project

プログラミング言語あれこれ（目次2）

3

M-project

この中に知っているものがありますか？

過去から現在まで、いろんなプログラミング言語が提案され、いまでも使われていたり最早使われなくなったりしています

次のプログラミング言語をあなたは知っていますか？
https://ja.wikipedia.org/wiki/%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%9F%E3%83%B3%E3%82%B0%E8%A8%80%E8%AA%9E%E4%B8%80%E8%A6%A7 にプログラミング言語一覧がある。

- 教えたことがある（それくらい群しい）
- 仕事に使ったことがある／使っている
- まあまあ使える（簡単なプログラムを書くことができる）
- 名前を聞いたことがあるだけ
- 名前を聞いたこともない

Ada, APL, AWK, BASIC, B, Bash/Shell/tcsh/PowerShell, C, CASL, Common LISP, C++, Objective-C, C#, COBOL, Delpji, Forth, FORTRAN, FORTRAN77, Go, Groovy, HTML/CSS, IPL, Java, JavaScript, Kotlin, LISP, LOGO, Lua, MATLAB, Mathematica, Modula-2/-3, MUMPS, Occam, PASCAL, Perl, PHP, PL/0, PL/1, Prolog, Python, R, RPG, Ruby, Scala, Scratch, Simula, Smalltalk, SNOBOL, SPSS, SQL, Swift, Tcl/Tk, TeX, TypeScript, VBA, VB.NET, Visual Basic, WATFOR / WATFIV

アセンブリ言語、機械語

次のシートの図を参照せよ。

4

こんなにもたくさんのプログラミング言語があります

ウイキペディによると、こんなにもたくさんのプログラミング言語があります：
https://ja.wikipedia.org/wiki/%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%9F%E3%83%B3%E3%82%B0%E8%A8%80%E8%AA%9E%E4%B8%80%E8%A6%A7

A

- A
- A+
- ABAP
- ABC
- ABCL
- ActionScript
- ActiveBasic
- Ada
- Advanced Boolean Expression Language（ABEL）
- Agena
- AHDL
- ALGOL
- Alice
- ash
- APL
- Apex
- AppleScript
- Arc
- as
- Atom
- AutoIt
- AutoLISP（英語版）
- AWK

B

- B
- Bash
- BASIC
- BCPL
- Befunge
- BF-BASIC'n
- Bioera
- BLISS
- Bluespec
- Boo
- BrainCrash
- Brainfuck

5

こんなにもたくさんのプログラミング言語があります（2）

C

- C
- C#
- C++
- C++/CLI (C++ Common Language Infrastructure)
- C++/CX（英語版） (C++ Component Extensions)
- CAL
- Caml
- Cantata
- CAP-X
- CASL
- Cecil
- CFScript（英語版）
- Cg
- Chapel
- Chef
- CHILL
- Clipper
- Clojure
- CLU
- Co-array Fortran
- COBOL
- CoffeeScript
- Common Lisp
- Component Pascal
- Concurrent Clean
- Concurrent Prolog
- Constraint Handling Rules
- CPL
- csh
- CUDA C/C++
- Curl
- Curry
- Cω

D

- D
- Dart
- dBase
- Delphi
- Dylan

E

- ECMAScript
- Eiffel
- Elixir
- Emacs Lisp
- Enterprise Generation Language
- Erlang
- Escapade
- Esterel
- Euclid
- Euphoria
- E

6

こんなにもたくさんのプログラミング言語があります（3）

F

- F*
- F#
- Factor
- False
- Fantom
- Ferite
- Ficl
- Flavors
- FlowDesigner
- Forth
- FORTRAN
- Fortress
- FoxPro

G

- GLSL
- Go
- Groovy
- Guarded Horn Claus

H

- Hack
- HAL/S
- Hardware Join Java
- Haskell
- Haxe
- HDCaml
- HLASM
- HLSL
- HML
- HOLON
- HSP
- HQ9+
- Hydra
- HyperTalk

I

- Icon
- ID
- IDL (interactive data language)
- Inform
- InScript
- INTERCAL
- Io
- IPL
- IronPython
- ISWIM

J

- J
- Java
- Jancy
- Java FX Script
- JavaScript
- JHDL
- JScript .NET
- J#
- JSX
- Jolie
- Julia
- Jython

7

こんなにもたくさんのプログラミング言語があります（4）

K

- KEMURI
- Kent Recursive Calculator (KRC)
- Kuin
- KL1
- KornShell (ksh)
- Kotlin

L

- LabVIEW
- Lazy K
- Lava
- LFE（英語版）
- Limbo
- Linda
- Linden Scripting Language (LSL)
- Lingo
- Lisaac
- LISP
- LOGO
- Lola
- LotusScript
- Lua
- Lucid
- Lush
- Lustre

M

- M言語
- m4
- Malbolge
- Mana
- Maple
- MASM
- MATLAB
- Mathematica
- Max
- Mercury
- Mesa
- MIL/W
- Mind
- Mindscript
- Misa
- MixJuice
- ML
- Modula-2
- Modula-3
- MONAmona
- Mops
- MQL4
- MQL5
- MSIL
- MyHDL

8

こんなにもたくさんのプログラミング言語があります（5）

N

- Napier88
- NASM
- Nemerle
- Nim
- Noop
- NScripter

O

- O
- Oberon
- Oberon-2
- Object Pascal
- Object REXX
- Object Tcl (OTcl)
- Objective-C
- Objective Caml (OCaml
- Occam
- Ook!
- OpenOffice.org Basic
- OPS
- Oz

P

- P''
- Pacbase
- PALASM
- PARLOG
- Pascal
- PBASIC
- PCN (program composition notation)
- Perl
- PHP
- Pic
- Piet
- Pike
- pine
- PL/0
- PL/I
- Planner
- pnuts
- Pony
- PostScript
- PowerBuilder
- PowerShell
- Processing
- Prograph CPX
- Prolog
- Pure Data
- PureScript
- PWCT(en:PWCT_(software))
- Pxem
- Python

9

こんなにもたくさんのプログラミング言語があります（6）

Q [編集]

- QtScript
- Quorum

R [編集]

- R
- Racket
- REALbasic
- REBOL
- REXX
- RHDL
- Ring
- RPG
- RPL（Reverse Polish LISP）
- Ruby（汎用プログラミング言語）
- Ruby（ハードウェア記述言語）
- Rust

S

- SAL
- SAS
- Sather
- Scala
- Scheme
- Scratch
- Seed7
- Self
- SFL
- sh
- Shakespeare
- Short Code
- Simula
- Simulink
- SISAL
- SKILL
- SLIP（プログラミング言語）
- Smalltalk
- SMILEBASIC
- SNOBOL
- SPARK（英語版）
- Squeak
- Squirrel
- SPSS
- Standard ML
- Stata
- superC
- Swift
- SystemC
- SystemVerilog

T

- t3x
- TAL
- Telescript
- TeX
- Text Executive Programming Language
- Tcl
- tcsh
- Tenems
- TL/I
- Tonyu System
- TTS
- TTSneo
- Turing
- TypeScript

10

こんなにもたくさんのプログラミング言語があります（7）

U

- Unified Parallel C （UPC）
- Unlambda
- UnrealScript

V

- VBScript
- Visual Basic
- Visual Basic .NET
- Visual C .NET
- Visual C++ .NET
- Visual C# .NET
- Visual Studio
- Verilog HDL
- VHDL
- Viscuit
- Vala
- V

W

- Whirl
- Whitespace
- WICS
- WMLScript
- Wyvern

X

- X10
- XQuery
- XSLT

Y

Z

- zsh

11

プログラミング言語の分類

プログラミング言語の分類の仕方は多様ですが、例えば次のような分類の仕方もあります。
これは言語の汎用度（人間にとっての使い易さ）による分類です。

- **機械語（マシン語）**
- **アセンブリ言語**
- **汎用プログラミング言語**
  - ✓ 手続き型言語
    - BASIC, C, COBOL, FORTRAN, PL/1, Python
  - ✓ 関数型言語
    - APL, Haskell, LISP
  - ✓ 論理型言語
    - Prolog
  - ✓ オブジェクト指向型言語
    - C++, Java, Ruby

12

プログラミング言語の分類（2）

次のような分類の仕方もあります。
これは、プログラムをコンピュータが実行できる形に変換する方式の違いで分類したものです。

- **スクリプト言語**
  - ✓ AWK, JavaScript, Lua, Perl, Python, RGSS
- **インタープリタ言語**
  - ✓ BASIC, LISP, Perl, PHP, Python, Ruby
- **コンパイラ言語**
  - ✓ C, C++, C#, COBOL. FORTRAN, Java, PHP
- **LL言語**
  - ✓ AWK, JavaScript, Lua, Perl, PHP, Python, Ruby

13

機械語

機械語(machine language)とは、機械（＝そのとき使っているコンピュータ）だけが理解できる言語のこと。

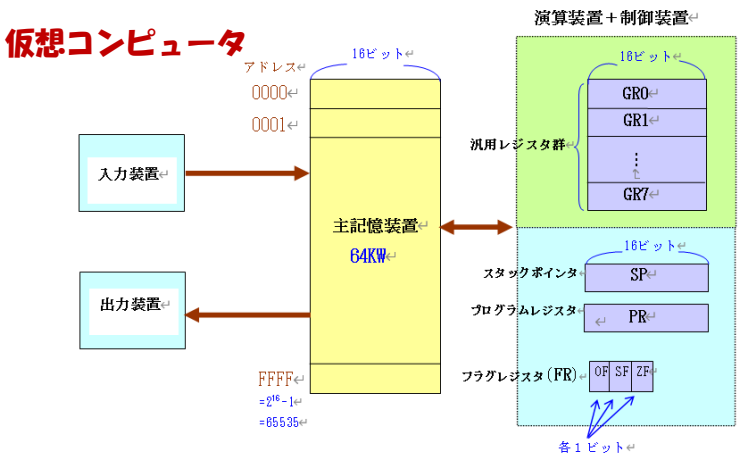
機械語は、コンピュータのプロセッサ（処理装置）が直接解釈（＝処理回路に渡す信号に変換）してすぐに実行することが可能であるような、一連の命令を 0,1 だけで記述したデータそのものであるが、それをプログラミング言語とみなしてこのように呼びます。

機械語とアセンブリ言語を理解するには、コンピュータが基本的にどのような構造をしているかを知らなければならない。

例として COMET II（情報処理技術者試験で使われる仮想的なコンピュータの）というコンピュータを考えよう（ただし、説明を簡単にするために、変更した部分もあります）。

14

仮想コンピュータ



15

機械語プログラムと記憶装置

細かいことは分からなくて結構。ここでは、（主）**記憶装置**と、**演算装置**内の**汎用レジスタ群**にだけ注目して下さい。

記憶装置は、機械語の1つの**命令**（命令は16ビット（＝2進数16桁）で表されます）や変数やデータなどを記憶しておく部分で、0000 から FFFF まで（4桁の16進数＝16桁の2進数）の番号（**アドレス（番地）**と言います）が付けられています。

例えば、

「00AB 番地にあるデータと、00CD 番地にあるデータを足してその結果を 00EF 番地に入れなさい」

という命令は

001500AB 003500CD 002500EF

となります。**先頭の16進3桁**の 001 は「**下位4桁の16進数**が表す番地に入っているデータをGR5 レジスタに入れなさい」という命令（**4桁目**の5が GR5 を表している）で、先頭の 003 は「下位4桁の16進数が表す番地に入っているデータを GR5 に足しなさい」という命令を表しています。最後の 002 は「GR5 レジスタの内容を下位4桁が表す番地に入れなさい」という命令です。

機械語のプログラムは、このような命令が実行される順に記憶装置に入れておかれます。、

16

アセンブリ言語

コンピュータは 0, 1 の並びである機械語しか理解できませんが、これを人間が書いたり読んだりするのは大変なので、人間にも分かりやすくするために略語を使って1対1に対応させたものを**アセンブリ言語**といいます。

例えば、前述の機械語

001500AB 003500CD 002500EF

は

```
LOAD  GR5,X    XのデータをGR5に入れよ（Xは番地00ABに付けた名前）
ADD   GR5,Y    YのデータをGR5に足せ（Yは番地00CDに付けた名前）
STORE GR5,Z    GR5に入っているデータをZに入れよ（Zは番地00EFに付けた名前）
```

のように書きます。

機械語もアセンブリ言語もコンピュータ（より正確には、CPU）の仕様に依存して決まりますので、同じ製造メーカーのコンピュータであっても機種ごとに違います。

17

汎用プログラミング言語

汎用プログラミング言語とは、その名の通り、特定の用途やコンピュータの機種に依存せずに、様々な用途のプログラムを書くことができるプログラミング言語です。

高級言語とか高水準言語とも言います。

当然、人間にとってもやりたいことを記述しやすく、また記述されたプログラムを読んで理解することも容易な言語です。

例えば、前述の機械語やアセンブリ言語と同じことは

Z←X+Y

と書くことができます（言語によって、書き方は少し異なります）。

18

機械語・アセンブル言語、汎用プログラミング言語の違い

X と Y の和を Z に代入するプログラム

機械語(8080A)	アセンブリ言語(8080A)	汎用高級言語	
00111010	LDA X	Z=X+Y	(FORTRAN)
00000001	LXI H,Y	COMPUTE Z=X+Y.	(COBOL)
10000000	ADD M	z:=x+y;	(PASCAL)
00100001	STA Z	Z←X+Y	(APL)
00000010		(SETQ Z (+ X Y))	(LISP)
10000000		Z=X+Y;	(PL/I)
10000110		_z is _x+_y	(Prolog)
00110010		z=x+y;	(C,C++,Java)
00000011			
10000000			

19

手続き型言語

手続き型言語で書くプログラムでは、**計算手順**（**アルゴリズム**＝基本的処理や操作がどのような順序で実行されるべきであるか）を記述します。

代表的な言語： BASIC, C, C++, COBOL, FORTRAN, Pascal, PL/I, Python, …

次のシート以降では、これらの言語のいくつかで n! を求めるプログラムを書いてみます。考え方として、

(A) **反復的定義**

$$n! = 1 \times 2 \times \cdots \times n$$

(B) **再帰的定義**

$$n! = \begin{cases} 1 & (n = 0 \text{ のとき}) \\ (n-1)! \cdot n & (n > 0 \text{ のとき}) \end{cases}$$

等がありますが、どちらを使うかによってプログラムは違ったものになります。

20

Pascal

Pascalによるプログラム例

再帰的関数の定義

```
program factorial(input, output);
  var n: integer;
  function f(n: integer);
  begin
    if n=0 then f := 1
    else f := f(n-1)*n
  end;
begin
  readln(n); .
  writeln(n, "! = ", f(n))
end.
```

反復的実行

```
program factorial(input, output);
  var f, i, n: integer;
begin
  readln(n);
  f := 1;
  for i := 1 to n do f := f* i;
  writeln(n, "! = ", f)
end.
```


21

FORTRAN

FORTRANによるプログラム例

```
      FUNCTION F(N)
      F=1
      DO 10 I=1,N
   10 F=F*I
      RETURN
      END

      READ(5,100) N
  100 FORMAT(I2)
      M=F(N)
      WRITE(6,200) M
  200 FORMAT(I20)
      STOP
      END
```

```
      READ(5,100) N
  100 FORMAT(I2)
      F=1
      DO 10 I=1,N
   10 F=F*I
      WRITE(6,200) F
  200 FORMAT(I20)
      STOP
      END
```


22

Cによるプログラム例

```
#include <stdio.h>
int f(int n) {
  if (n==0) return 1;
  else return f(n-1)*n;
}
main() {
  int n;
  scanf("%d", n);
  printf("n != %d", f(n));
}
```

```
#include <stdio.h>
main() {
  int f, i, n;
  scanf("%d", n);
  f = 1;
  for (i = 1; i <= n; i++)
    f = f*i;
  printf("n != %d", f);
}
```


23

BASIC

BASICによるプログラム例

```
   10 INPUT N
   20 GOSUB 1000
   30 PRINT F
   40 END
 1000 F=1
 1010 IF N=0 THEN 1050
 1020 FOR N=1 TO N
 1030 F=F*I
 1040 NEXT I
 1050 RETURN
 1060 END
```

```
10 INPUT N
20 F=1
30 FOR I=1 TO N
40 F=F*I
50 NEXT I
60 PRINT F
70 END
```


24

関数型言語

関数型言語では、プログラムは関数の集まりであり、関数を定義することがプログラミングであり、すべての計算は関数の適用によって行われます。　手続き型言語と違い、変数へ代入するという概念は(初期値設定以外では)ない。

広い定義では Lisp (LISP) やその方言の1つである Scheme は関数型言語に分類されるが、これらの言語では変数の書き換えが可能なため、厳密な定義ではこれらは関数型言語の機能を備えた手続き型言語であるもいえる。

代表的な関数型言語には、APL、Haskell, Lisp、ML、Scheme などがあります。

25

関数型言語LISPのプログラム例

（例）n! を再帰的関数として定義する

Lisp によるプログラム例

```
(DEFFUN f (n)                          …  関数 f(n)を定義する
  (COND ((greaterp n 0) f(n-1)*n)      …  n>0 なら f(n-1)*n が f(n)の関数値
        (T 1)                          …  そうでないなら 1 が f(n)の関数値
  )
)                                      …  定義はここで終わり

(f 5)                                  …  これは 5! の計算
```


26

論理型言語

論理型言語では、問題を論理的関係として記述することがプログラミングである。

代表的言語：　Prolog, GHC, OPS

（例）親子兄弟関係

Prolog によるプログラム例

```
sibring(_x,_y):-sibring(_y,_x).        x と y が兄弟なら y と x も兄弟である
father(_x,_y):-parent(_x,_y),man(_x).  x が y の親で男なら、x は y の父親である
man(A).                                A は男である
man(B).                                B は男である
parent(A,C).                           A は C の親である
?-father(_x,C).                        C の父親は誰か？
```


27

オブジェクト指向型言語 (1)

オブジェクト指向型言語では、「もの」(object, オブジェクト)を主体として考えます。すなわち、プログラムが対象としようとしている世界に属す「もの」(＝オブジェクト)を中心に据え、それらの特徴を記述することがプログラミングです。

例えば、レストランという世界を考えると、個々の客、個々のウェイター、個々のコックなどがそれぞれオブジェクトであり、個々の客やウェイターやコックは誰も決められた一定の仕事(動作)をします(言い換えれば、それぞれに特徴的な独自の機能＝役割を持っています)。例えば、客は料理の注文をし、ウエイターは客からの注文をコックに伝えたり、調理が終った料理を客に出したりし、コックは料理を作る。このような機能(動作)のそれぞれを Java ではメソッド(method)といいます。

一方、こういった仕事をするためにはいろいろな種類の道具や材料(コックは鍋や包丁など、ウエイターはメニューや注文伝票やPDAなど)が必要であるし、個々のウエイターや個々の客や個々のコック(すなわち、個々のオブジェクト)を区別する名前や年齢、性別などの性質・特性(属性、attribute)を持っています。このような属性を保持するもの(変数や配列など)をオブジェクト指向型言語の1つである Java ではフィールド(field)といいます(他のオブジェクト指向言語ではデータメンバーとかメンバー変数とか、単にメンバーとかいうものもあります)。

28

オブジェクト指向型言語 (2)

こういったオブジェクトの属性・性質、すなわち、仕事・動作(機能・能力)や道具(情報・データ)をきちんと決めることによって、抽象的概念である「客」とか「ウエイトレス」とか「コック」といったものを Java の世界における具体的なものとして特徴付けることができます。

オブジェクト指向言語では、こういった抽象的な概念(世界)をクラス(class)といい、各クラスに属す個々の「もの」をオブジェクトといい(そのクラスのインスタンス(instance、「具体例」の意)ともいう)、各オブジェクトが行う動作・仕事・機能・操作をメソッド(method)といい、特定のオブジェクトに特定のメソッドを実行して欲しいときにメソッドに渡すデータ・情報をメッセージ(message)といいます。オブジェクト指向でないプログラミング言語ではメソッドのことを通常、関数(あるいは、手続き)といい、メッセージのことを関数の引数というのが普通です。

共通の特性をもつオブジェクの集団を抽象的に表したものをクラスといい、オブジェクト指向型言語ではいくつかのクラスを定義することがプログラミングです。

代表的言語: Smalltalk, C++, C#, Java, JavaScript, Delphi, Ruby, PHP

29

Java

Java によるプログラム例

「銀行口座」という世界をプログラムとして表す。

「口座」を表す bankAccount というクラスは、次のようなフィールドやメソッドを持つものとして定義することができる:

- フィールド name:口座所有者(文字列)
- フィールド number:口座番号(整数)
- フィールド balance:残高(整数)
- メソッド deposit:預金操作
- メソッド withdraw:預金引き出し操作
- メソッド main:このメソッドから実行が開始される

30

Java(つづき)

```
class bankAccout {   // クラスの定義
  String name;       // 口座所有者は文字列で表す
  int number;        // 口座番号は整数
  int balance;       // 残高は整数
  void deposit(int amout) {   // 預金に関する操作を表すメソッド
    balance += amount;
  }
  void withdraw(int amount) {   // 預金の引き出しを表すメソッド
    balance  - = amount;
  }
```

```
  public static void main(String[ ],args) {
    bankAccount a = new bankAccount();
    a.name = "Taro";
    a.number = 12345;
    a.balance = 0;
    a.deposit(10000)
    a.withdraw(5000);
    System.out.println(a.balance);
  }
}
```


31

コンピュータがプログラムを実行できるためには翻訳あるいは通訳が必要

人間にとってより分かりやすい言語(アセンブリ言語や、BASIC, FORTRAN, C, Pascalなどの汎用プログラミング言語)で書かれたプログラムは、コンピュータのハードウエアが直接理解して実行できるプログラム(=機械語)に翻訳する必要があります。

アセンブリ言語で書かれたプログラムを機械語に翻訳するプログラムをアセンブラ(assembler)といい、汎用言語で書かれたプログラムを機械語に翻訳するプログラムがコンパイラ(compiler, 翻訳系)といいます。

コンパイラは実行前に前もって翻訳をすませるのに対し、実行時にプログラムを1行1行逐一通訳するようなプログラムをインタープリタ(interpreter, 通訳系)といいます。

以上を総称して言語プロセッサといいます。

32

アセンブラ（1）

機械語でプログラムを組むのはきわめて大変なので，機械語命令を8ビットの2進数（命令コード）で書いたり，命令の対象になるデータの格納されている場所（主記憶装置上のアドレス）を16ビットの2進数で書いたりする代わりに，命令に決まった名前（ニモニック）を付けておき，また，アドレスにも自由に名前（ラベル）を付けることができるようにしたものがアセンブリ言語（アセンブラ語）です。

アセンブリ言語は，こういった名前付けなどがうまくできるようにする命令（DS命令やDC命令）を備えていると同時に，そのようなアセンブリ言語で書かれたプログラムを機械語に翻訳する作業（この作業は，アセンブラと呼ばれるプログラムが行います）がきちんとできるようにするためにアセンブラへ情報（プログラムの始まり，終わり，実行開始位置など）を伝えるための命令（アセンブラ制御命令）も持っています。さらに，マクロ機能のような高級な機能を持っていることもあります。

33

アセンブラ（2）

アセンブリ言語は人間にとっては機械語でプログラムを書くよりはるかに簡単ですが，コンピュータのハードウエアはアセンブル言語で書かれたプログラムを直接理解することができないので機械語に翻訳してあげる必要があります。

これを行うプログラムのことをアセンブラ（assembler）といいます。

翻訳前のプログラムを原始プログラム（ソースプログラム，ソースコード）といい，翻訳した結果を目的プログラム（オブジェクトコード，目的コード）といいます。

34

アセンブリ言語と，翻訳された機械語

アセンブラ語プログラム			機械語コード(relocatable code)	
			相対番地（LC）	コード
EXPL	START	BGN		
BGN	LD	GR1,A	0000	10 10 000B
	LD	GR2,B	0002	10 20 000C
	SUBA	GR1,GR2	0004	21 12
	JPL	DONE	0005	65 00 0008
	LD	GR1,GR2	0007	10 12
DONE	ST	GR1,MAX	0008	11 10 000D
	RET		000A	81 00
A	DC	12	000B	000C
B	DC	34	000C	0022
MAX	DS	1	000D	0000
	END			

35

汎用プログラミング言語とコンパイラ

アセンブラといえども，人間にとってはまだ使いにくい。アセンブラは機械語とほぼ1対1対応なので，それぞれのマシンに依存した仕様になってしまい，コンピュータごとに別々のアセンブラを覚えなければならないという難点もある。

そこで，人間の使う言語（自然言語）により近い言語を使ってプログラムを書くことができるようにすると，今度はそのような言語で書かれたプログラムを機械語に翻訳するプログラムが必要になる。

自然言語に近いプログラミング言語で，特定用途に限定せずどんな目的のプログラムでも書けるようにデザインされたものを汎用プログラミング言語（general purpose programming language）とか高級言語（high-level language）とかコンパイラ言語といい，そのような言語で書かれたプログラムを機械語に翻訳するプログラムをコンパイラ（compiler）という。

36

コンパイリング

汎用プログラミング言語（高級言語、コンパイラ言語と）で書かれたプログラムを機械語に翻訳するプログラムをコンパイラ（compiler）といい、コンパイル作業を行うことをコンパイリング（compiling）といいます。下図では高級言語のプログラムが機械語に直接翻訳されていますが、アセンブリ言語（一般に、中間言語）に翻訳してからそれをアセンブラで機械語に翻訳することが多いです。

37

38

スクリプト言語

スクリプト言語は、プログラムの記述や実行を比較的簡易に行うことができる（例えば、変数に型をつけない）ような言語の総称です。多くの場合、スクリプト言語はインタープリタ型言語であり、コンパイラ型言語に比べて実行までに必要な処理に手間がかからないという特徴を持っています。ただし、インタプリタ方式でなく性能向上のために実行時コンパイルなどを利用するものも多いです。

アプリ・ソフトの動作内容を台本のように記述して制御するためのプログラミング言語の例としては、OS の１つである UNIX のシェルスクリプト、ウェブブラウザなどに対する JavaScript などがあり、比較的単純なプログラムを記述するためのプログラミング言語としての例としては、Perl や PHP などがあります。

39

JavaScriptのプログラム例

```
<script type="text/javascript">
  // main
  for (n=1; n<=50; n++) {
     document.writeln(n, "! = ", factorial(n));
  }
  // definition of factorial
  function factorial(n) {
     if (n==0) {
        return 1;
     }
     else {
              return factorial(n-1)*n;
     }
  }
</script>
```

n! を求めるプログラムと実行結果

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
```


40

ＬＬ言語

LL言語とは、Lightweight Language（略してLL．**軽量プログラミング言語**）のことで、短い記述で処理を書くことができ、手軽に取り扱うことができるプログラミング言語の総称です。

代表的言語：Perl, PHP, Python, Ruby

CGI（ウエブサーバがブラウザなどからの要求に応じてプログラムを実行する仕組みの1つ）を Perl で記述した例（単に Hello とだけ画面に出力する）

```
#!/usr/bin/perl
print "Content-type: text/html ¥n¥n";
print "Hello";
```


41

LL言語のプログラム例

棒グラフを描く Python のプログラム例

```
import matplotlib.pyplot as plt
import numpy as np
values = [9143041, 4151178, 57738, 26627]     # このデータを棒グラフにする
indx = np.arange(4)                           # カテゴリ数
plt.bar(indx, values)
plt.xticks(indx, ['All-Ku', 'All-Shi', 'All-Gun', 'All-Islands'])
plt.show()                                    # グラフの表示
```


42