プログラミング言語

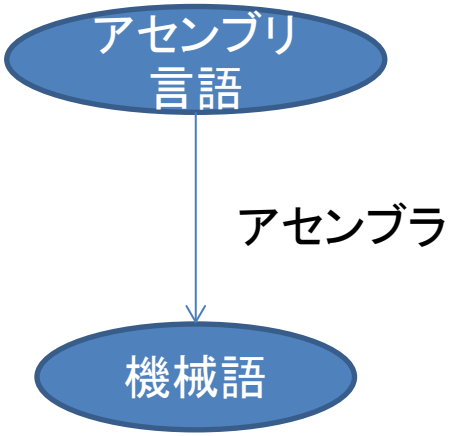
コンピュータリテラシ

2016年5月17日

建部修見

機械語とアセンブリ言語

- 機械語（マシン語）
  - CPUの命令セット
  - i386, x86_64, arm64, mips, powerpc, ...
- アセンブリ言語
  - 機械語とほぼ1対1に対応
  - add R1, R2, R3 # R1 <- R2 + R3
  - 分岐先にラベルが使える
  - アセンブラで機械語に変換する
  - 機械語に依存
    - 異なるCPU間で互換性の問題



高級プログラミング言語（1）

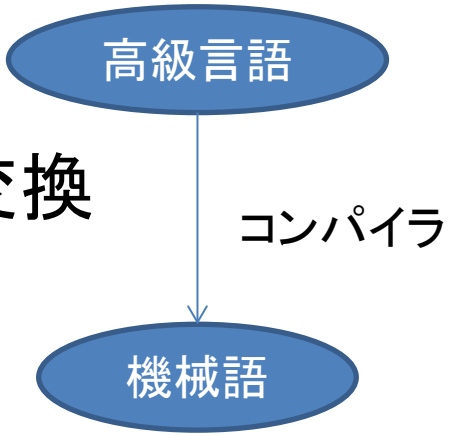
- 1950年代
  - FORTRAN – 科学技術計算
  - COBOL – 事務処理
  - LISP – 記号処理
- 1960～70年代
  - Simula, Smalltalk – オブジェクト指向
  - BASIC, Pascal – 初学者向け
  - C – OSなどシステムプログラム向け
  - Prolog – 論理型
  - ML – 関数型

高級プログラミング言語（2）

- 1980～
  - C++ - Cにオブジェクト指向の考え方を取り入れ
  - Perl, Ruby, Python – スクリプト言語
  - Java, C# - 複数のハードウェアで安全に動作させる目的で設計されたオブジェクト指向言語
  - JavaScript – Webクライアント用スクリプト言語

コンパイル実行とインタプリタ実行

- コンパイル実行方式
  - 高級言語をコンパイラで機械語に変換
  - 異機種では再コンパイルが必要
  - FORTRAN, C, C++など



- インタプリタ実行方式
  - インタプリタは高級言語プログラムを読込、処理を実行する
  - コンパイルは不要だが処理が一般に遅い
  - Perl, Python, Rubyなど

仮想機械による実行方式

- 仮想機械（VM）を定義
- 仮想機械の機械語プログラムにコンパイル
- CPUは仮想機械の機械語プログラムを実行
- ポータビリティとある程度の実行速度
- Java, C#など

プログラミング言語C

- 関数と変数からなる
- 関数は文で構成される
  - 文はセミコロン”;”で終わる
  - { }で複文（ブロック）を単一の文と同等に
- 変数名（関数名）
  - 先頭は英字、アンダースコア”_”
    - 大文字小文字区別あり
  - 先頭以外は数字もOK、31文字まで
- データ型
  - 変数（関数）にはデータ型がある
  - char, int, float, double
  - long, short, signed, unsigned
  - 変数（関数）を利用する前にデータ型の宣言が必要

    int a, b, c;

式

- 全ての式（算術演算、代入）は値を返す
- a++　　# aの値を返し、aをインクリメント（a = a + 1）
- a * b, a / b, a % b
- a + b, a - b
- a > b, a >= b, a < b, a <= b
- a == b, a != b　　# 真なら1、偽なら0を返す
- a || b　　# aまたはbが真なら1を返す
- a = b　　# 右辺のbの値を返す
  - a = b = c

if文

- if (式) 文1 else 文2
  - 式が真なら文1を実行し、偽なら文2を実行する

```
if (s > 0)
    a = 1;
else if (s < 0)
    a = -1;
else
    a = 0;
```

while文

- while (式) 文
  - 式を満たしている間は文を繰り返す

```
s = i = 0;
while (i < n) {
    s = s + i;
    i++;
}
```

for文

- for (式1; 式2; 式3) 文
  - 以下と同じ

```
式1;
while (式2) {
    文
    式3;
}
```

```
s = 0;
for (i = 0; i < n; i++)
        s = s + i;
```

フィボナッチ数

$$\mathrm{fib}(k) = \begin{cases} 1 & (k = 0 \text{ or } k = 1) \\ \mathrm{fib}(k-1) + \mathrm{fib}(k-2) & (k \geq 2) \end{cases}$$

fib関数の定義。kという引数をとり、kが0か1のとき1を返し、
それ以外のときfib(k - 1) + fib(k - 2)を返す。
自分自身の関数を再び呼ぶので再帰呼び出しという

```
int
fib(int k)
{
     if (k == 0 || k == 1)
          return (1);
     else
          return (fib(k - 1) + fib(k - 2));
}
```

テスト

- Cではmain関数を実行する

```
#include <stdio.h>

int
main()
{
        int i;

        for (i = 0; i < 46; i++)
                printf("fib(%d) = %d¥n", i, fib(i));
        return (0);
}
```

main関数の定義。整数型の変数iの宣言。
iを0から45まで1ずつ増やしながら繰り返す。
printfは””で囲まれた文字列を表示する関数。
この関数はstdio.hで宣言されている。
文字列中 %d はその後の引数（整数型）の値を表示する。

プログラム1

```
#include <stdio.h>

int
fib(int k)
{
	if (k == 0 || k == 1)
		return (1);
	else
		return (fib(k - 1) + fib(k - 2));
}

int
main()
{
	int i;

	for (i = 0; i < 46; i++)
		printf("fib(%d) = %d¥n", i, fib(i));
	return (0);
}
```

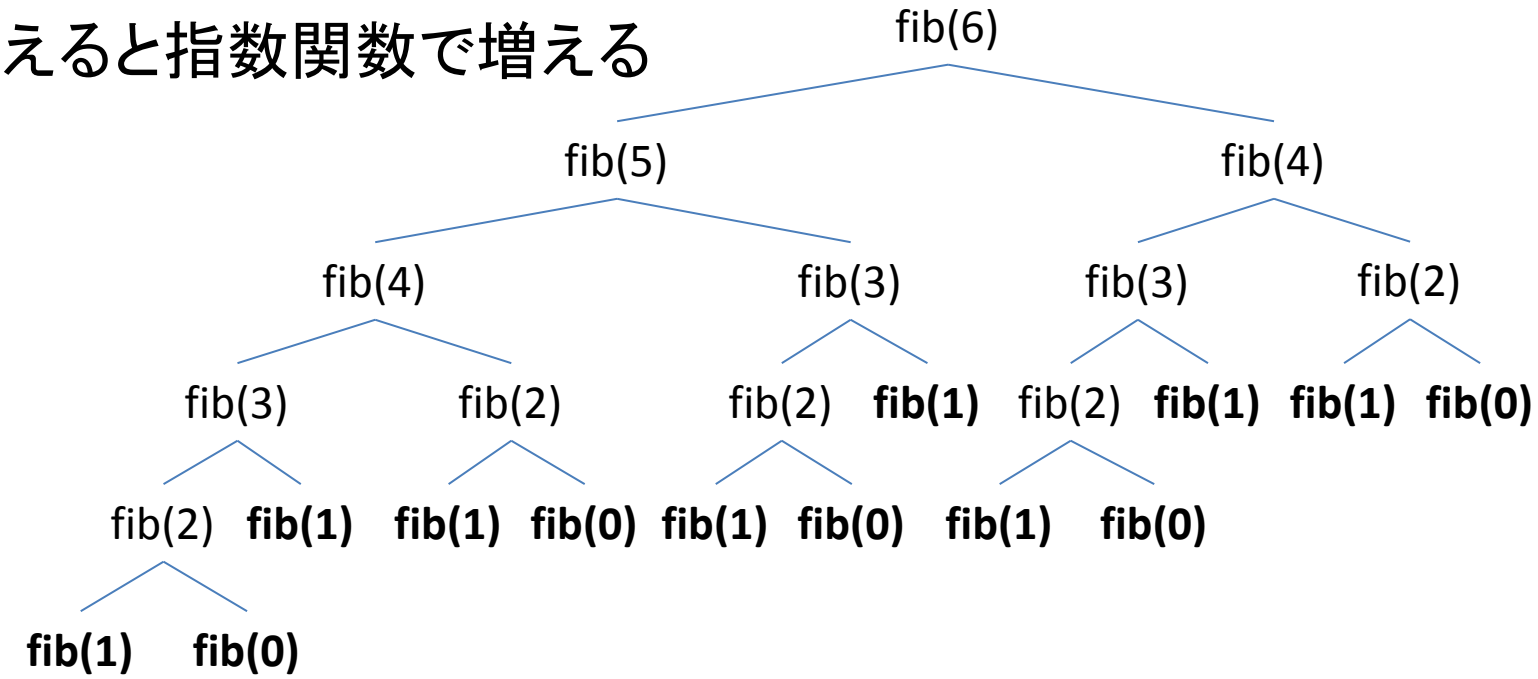
演習

- プログラム1をfib.cで保存し、以下のようにコンパイル、実行してみよう

```
$ cc -O -o fib fib.c    # -Oは最適化オプション
                        # -o fib でコンパイル後の実行ファイル名を指定する
$ ./fib
```

プログラム1の計算時間

- fib(k)はfib(k - 2)とfib(k - 1)を呼び出す
- fib(0) = fib(1) = 1なので、fib(k)はfib(k)回関数fibを呼んでいる
- 黄金比$\varphi = \frac{1+\sqrt{5}}{2}$とおくと$\mathrm{fib}(k) \cong \frac{\varphi^{k+1}}{\sqrt{5}}$と近似できることが知られている
- kが増えると指数関数で増える



フィボナッチ数

- f(k)を覚えておくことで順番に求められる
  - f(0) = 1
  - f(1) = 1
  - f(2) = f(1) + f(0)
  - f(3) = f(2) + f(1)
  - f(4) = f(3) + f(2)
  - f(5) = f(4) + f(3)

配列

- 変数の列

  int a[10];

  – 整数型変数a[0], a[1], …, a[9]の宣言

配列を用いたフィボナッチ数

```
int
fiba(int n)
{
	int f[46] = { 1, 1 };
	int i;

	if (n > 45)
		return (-1);
	if (n == 0 || n == 1)
		return f[n];
	for (i = 2; i <= n; ++i)
		f[i] = f[i - 2] + f[i - 1];
	return f[n];
}
```

f[0]からf[45]までの整数型の配列を宣言
f[0]とf[1]を1に初期化。他は未定義

配列は45までしか準備していないので
nが45より大きいと配列外を参照する
ため -1 を返す

演習（2）

- 配列を用いた関数fibaを呼ぶようにプログラム1を修正したプログラム2を作成し、fiba.cと保存し、コンパイルし、実行ファイルfibaを作成しよう
- timeコマンドを用いると時間を計測することができる。以下を実行して時間を比べてみよう。

  $ time ./fib

  $ time ./fiba

オプション演習（１）

- 以下のようにf(3)を計算するときにはf(0)は必要ないため上書き可能である
  - f(0) = 1
  - f(1) = 1
  - f(2) = f(1) + f(0)
  - f(0) = f(2) + f(1)　# f(3) = f(2) + f(1)
  - f(1) = f(0) + f(2)　# f(4) = f(3) + f(2)
  - f(2) = f(1) + f(0)　# f(5) = f(4) + f(3)
- このとき、配列の要素数は3で足りる
- fiba.cを修正して上記のようにフィボナッチ数を求めるプログラムfiba2.cを作成してみよう。fiba2.cをコンパイルし、実行ファイルfiba2を作成し、動作を確かめよう。また時間を計測してみよう。

行列を用いたフィボナッチ数

$$\begin{pmatrix} \mathrm{fib}(k+2) \\ \mathrm{fib}(k+1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathrm{fib}(k+1) \\ \mathrm{fib}(k) \end{pmatrix}$$

よって

$$\begin{pmatrix} \mathrm{fib}(k+1) \\ \mathrm{fib}(k) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{k} \begin{pmatrix} \mathrm{fib}(1) \\ \mathrm{fib}(0) \end{pmatrix}$$

行列のべき乗の計算

このとき、$Q = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$とおくと

$$Q^k = \begin{cases} E & (k = 0) \\ \left(Q^{k/2}\right)^2 & (k\text{は2以上の偶数}) \\ QQ^{k-1} & (k\text{は奇数}) \end{cases}$$

たかだか$2\log(k)$回の行列積計算で求まる

```
void
matvec(int r[2], int a[2][2], int v[2])   // r = a v
{
      r[0] = a[0][0] * v[0] + a[0][1] * v[1];
      r[1] = a[1][0] * v[0] + a[1][1] * v[1];
}

                                          // a1 = a2 a3
void
matmul(int a1[2][2], int a2[2][2], int a3[2][2])
{
      int i, j, k, l;

      for (i = 0; i < 2; ++i)
            for (j = 0; j < 2; ++j) {
                  l = 0;
                  for (k = 0; k < 2; ++k)
                        l += a2[i][k] * a3[k][j];
                  a1[i][j] = l;
            }
}
```

```
void
fibmat(int n, int a[2][2])        // a = Q^n
{
      int a2[2][2], a3[2][2];

      if (n == 0 || n == 1) {
            a[0][0] = 1;
            a[0][1] = 1;
            a[1][0] = 1;
            a[1][1] = 0;
            return;
      }
      if (n / 2 * 2 == n) {
            fibmat(n / 2, a2);
            matmul(a, a2, a2);
      } else {
            fibmat(n - 1, a2);
            fibmat(1, a3);
            matmul(a, a2, a3);
      }
}
```

```
int
fibm(int n)
{
	int ini[2] = { 1, 1 };
	int ret[2], a[2][2];

	if (n == 0 || n == 1)
		return (1);
	fibmat(n - 1, a);
	matvec(ret, a, ini);
	return (ret[0]);
}
```

課題（３）

- 行列を用いた関数fibmを呼ぶようにプログラム1を修正したプログラム3を作成し、fibm.cと保存し、コンパイルし、実行ファイルfibmを作成しよう
- fibmを実行して時間を計測してみよう

オプション演習（2）

- fib(46)を計算するとマイナスになってしまう。理由を考えてみよう
- 正しくfib(46)を計算できるようにプログラムを改良してみよう
  - ヒント：8バイト整数のデータ型はlong long
  - long long型の変数を表示するprintfのフォーマットは%lld