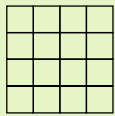






## 縦横線分描画計算機 (4) 13

- 4x4の格子を描いてみよう



なるべく少ない命令で描くにはどうすればよいか？工夫してみよう。

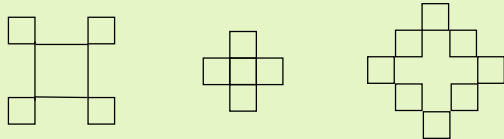
## 本日の提出課題 (出席の代わりに) 16

- 学籍番号、氏名をページに書き込み、この課題の実行結果をプリントアウトして授業の最後に提出してください。



## 縦横線分描画計算機 (5) 14

- いくつかの練習問題



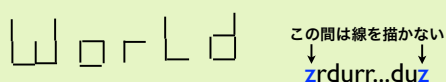
## 縦横線分描画計算機の問題点 17

- そもそもほとんど役に立たない。 それでもプログラミングを体験することはできる。
- 計算できる事柄が少なすぎる。
- 一般の問題が解けない。たとえば足し算や掛け算ができない。
- 計算機への入力が存在しない。出力のみ 一人前のコンピュータと同等の計算能力を持っていること

縦横線分描画計算機はチューリング完全ではない！普通のコンピュータよりも能力が低い。これから我々が扱う計算機はチューリング完全（普通のコンピュータで計算できることは計算量や計算時間を気にしなければすべて計算できる）である。Cの処理系もJavaScript言語の処理系もチューリング完全である。

## 命令の追加 15

- r, l, u, dのみでは一筆書きの図形しか描くことができない。そこで、もう一つ命令を追加する。'z'によってペンの上げ下ろしができる。ペンが上がっているときはペンを動かしても線は描かれない。'z'を一回行くとペンが上がり、もう一回おこなうとペンが下がる。最初はペンは下がっている。
- 以下の文字を書いてみよう。



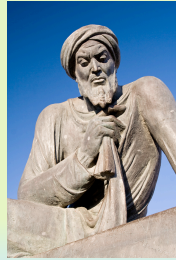
## プログラミング言語の構造 18

- 一般のプログラミング言語の構造は、本質的には縦横線分描画計算機のプログラミングと変わらないが、複雑な動作を表現するためのしくみや簡潔に書くためのしくみが含まれている。
- 特に、命令を繰り返し実行するためのしくみなどが含まれる。
- また、複雑なデータ構造を利用することができる。

```
int main(){
    int i;
    for (i = 0; i < 10; i++){
        printf("%5d", i);
    }
    putchar('\n');
    return 0;
}
```

## ◇◇ アルゴリズムとは何か ◇◇

- 「アルゴリズム」という言葉は9世紀のイラク バグダットの数学者 アル・フアリズミーから来ていると言われている。アルゴリズムは計算機上である特定の問題を解くための方法であり、プログラム自体もアルゴリズムと言えるが普通はプログラムよりも抽象的な方法論を指すことが多い。
- 計算機が与えられたとき、その上である問題を解決するためのアルゴリズムを考える場合、そのアルゴリズムを用いることによって必ず解けなければならない。すなわち、計算機が停止しなくなったり（永遠に計算し続けること）、エラーを吐いて止まってしまうことがあるとアルゴリズムとは言えない。



al-Khwārizmī

## ◇◇ 単純なアルゴリズムの例（3） ◇◇

### JavaScriptでの実現 (FireFox, version 1.7以上)

```
function euclid(a, b){
  while (b > 0)
    [a, b] = [b, a % b];
  return a;
} /* euclid */
```

### JavaScriptでの実現 (Chrome, ECMA-262レベル)

```
function euclid(a, b){
  while (b > 0){
    var c = b;
    b = a % b;
    a = c;
  }
  return a;
} /* euclid */

var a = 35;
var b = 56;
puts(a + ', ' + b + ', ' + euclid(a, b))
```

## ◇◇ 単純なアルゴリズムの例（1） ◇◇

### ユークリッドの互除法

アルゴリズムの中身はわからなくても良い

二つの正整数a, bの最大公約数 (GCD) を計算するには以下のようにすれば良い。

```
while (b > 0)
  (a, b) = (b, a % b)
```

この2点が言えないとアルゴリズムと呼べない。

最後のaの値が元のa, bのGCDとなる。

正当性：止まれば必ず正しい答えを出す  
 停止性：必ず停止する（計算が終わること）

これを言うためには、多くの場合、数学的な方法が必要となる。

## ◇◇ 単純なアルゴリズムの例（4） ◇◇

### OCamlでの実現

```
let rec euclid a b =
  if b > 0 then euclid b (a mod b)
  else a in
let a = 35 and b = 56 in
Printf.printf "%d %d %d\n" a b (euclid a b)
```

### Pythonでの実現

```
def euclid(a, b):
  while b > 0:
    a, b = b, a % b
  return a

a, b = 35, 56
print a, b, euclid(a, b)
```

### Rubyでの実現

```
def euclid a, b
  while b > 0 do
    a, b = b, a % b
  end
  return a
end

a, b = 35, 56
puts format("%d %d %d", a, b, euclid(a, b))
```

## ◇◇ 単純なアルゴリズムの例（2） ◇◇

### Cでの実現

```
#include <stdio.h>
#include <stdlib.h>

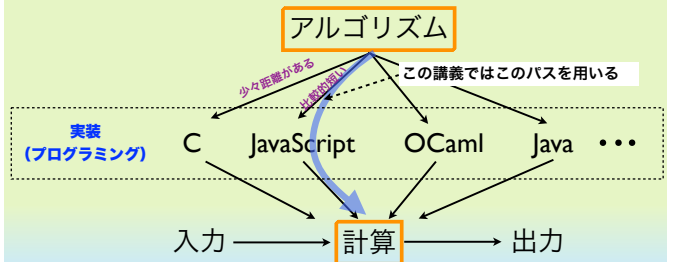
int euclid(int a, int b){
  while (b > 0){
    int am = b;
    int bm = a % b;
    a = am;
    b = bm;
  } /* while */
  return a;
} /* euclid */

int main(){
  int a = 34;
  int b = 56;
  printf("GCD(%d, %d) == %d\n", a, b, euclid(a, b));
  return 0;
} /* main */
```

Cだと行数が多くなり本質的な部分が少々見えにくい

## ◇◇ アルゴリズムと実装の関係 ◇◇

- 色々な実装に対応するために、アルゴリズムはある程度抽象的なものとなる。
- 現実の実装はその環境にしたがって、変える必要があるが、アルゴリズム自体の考え方が変わるわけではない。



## ◇◇ プログラミング言語 JavaScript について ◇◇

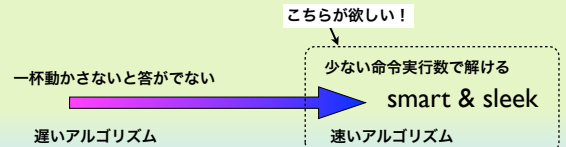
25

- JavaScriptの正式名称はEcmaScriptであるがここでは、JavaScriptと呼ぶ。JavaScriptはもともとMozillaによって作られた言語であるが、Ecma Internationalによって標準化されており、その仕様はECMA-262に定義されている。Javaという名前が使われているが、Java言語とは関係ない。FireFox, ChromeなどのWebブラウザでは直接JavaScriptを実行できるエンジンを備えている。特に「v8エンジン」(Google Chrome)と呼ばれる処理系や「spider monkey」(Mozilla Firefox) 処理系は、処理速度が極めて高速であることが知られている。
- この授業ではプログラミングにJavaScriptを用いる。JavaScriptは細かいところを気にしなくてもブラウザ上で普通に実行可能であり、C言語などに比べて柔軟性が高く、プログラムも書きやすい。本格的なソフトウェア開発の道具として用いるのではなく、アルゴリズムを記述するための道具として用いる。

## ◇◇ 計算量について ◇◇

28

- 色々なコンピュータ上でプログラムとして実現できるアルゴリズムはなるべく「速い」ものであることが望ましい。
- 実際の計算速度はプログラムの書き方やコンピュータそのものの速度などに依存するので一概に速度を決めることはできない。
- そのため、「良いアルゴリズム」と「悪いアルゴリズム」を見極めることは、かなり難しい場合がある。
- この講義では最後の部分で計算量の標準的な考え方について学ぶ。



## ◇◇ JavaScriptの特徴 ◇◇

26

- 文法がシンプルである。
- Webブラウザがあれば自宅でも大学でもどこでも動く。コンパイラ不要、インストール不要、もちろんタダ。
- とても高速である。Javaと同じくらいの速度 (Cコンパイラの半分くらいの速度) で計算することができる。

この授業の第2回～第4回にJavaScriptを用いてプログラミングの講義と演習を行う。

## ◇◇ 本日のまとめ ◇◇

29

- コンピュータ (計算機) の基本的な考え方と動作原理について述べた。(もちろん、細かな動き方やプログラムの書き方については述べていない。)
- アルゴリズムは究極的にはプログラムと同義である。しかし、現実のプログラミングは言語仕様などによって色々に変化するので、アルゴリズムはそれよりは抽象的なレベルで議論する。
- より計算量の少ないアルゴリズムを設計することが目標であるが、抽象的なレベルではなかなか何が最適なのかを述べることは難しい。

## ◇◇ この講義で当面学ぶこと ◇◇

27

- 欲しいものを見つけ出すこと。特に多くの可能性の中から本当の答えを見つけ出す方法について学ぶ。
- これを「探索」と呼ぶ。
- 多くの場合、探索は木構造と呼ばれるデータ構造を辿る作業となる。
- 問題によっていくつかの辿り方を考える。

