全学共通科目･工学部専門科目

「計算機科学概論」
アルゴリズムとプログラミング
その3

五十嵐 淳
igarashi@kuis.kyoto-u.ac.jp
大学院情報学研究科
通信情報システム専攻

担当分のメニュー

- 6/21, 6/28: アルゴリズムについて
- 7/5: 出張につき休講
- 7/12, 7/19: プログラミングについて
- (補講の予定・内容は未定です)

講義情報

http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/cs-intro/

今日のメニュー

- 前回の積み残し
  - アルゴリズムの正しさ
- プログラムとプログラミング言語
- JavaScript プログラミング入門
  - とにかく動かしてみよう

アルゴリズムの正しさ

- アルゴリズムが任意の入力に対し停止するか？
- アルゴリズムがその目的を任意の入力に対して果たすか？

  - マージソート・クイックソートは本当に整列するのか？
  - 互除法はいつでも止まるのか？本当に求まったのは最大公約数なのか？

ユークリッドの互除法

入力: 自然数n, m (ただし n ≧ m)

1. k を n ÷ m の余りとする
2. k について場合分け
   - k = 0 ⇒ m を出力として終了
   - k ≠ 0 ⇒ m, k を入力として互除法を行いその出力をそのまま出力とする

再帰(recursion)

互除法の正しさの証明(1/2)

- 以下のふたつは同値
  - 相異なる自然数 m, n の GCD が g である
  - 互いに素な自然数 a, b が存在し
    m = ga かつ n = gb

互除法の正しさの証明(2/2)

- m ÷ n の余りは a ÷ b の余り c の g倍
- しかも、b と c は互いに素、つまり GCD は g

  ⇒ 入力の GCD が再帰を越えて「保存」される
- プラス、入力の和は再帰の度に減っていく

- より厳密な証明には数学的帰納法を使う

数学的帰納法

- 「任意の自然数 n について P(n)」を示したければ、以下のふたつを示せばよい
  - P(0)
  - P(k) ならば P(k+1) (任意の k について)
    - P(k) を帰納法の仮定と呼ぶ
    - ひとつ小さい数については今示そうとしている P が成立することを使ってよい

「任意の n に対し 1からnまでの和 = n(n-1)/2」の証明

- P(0): 1から0までの和 = 0(0-1)/2

- P(k) を仮定して P(k+1) を示す
  - 1 から k+1 までの和
    = 1 から k までの和 + (k+1)
    (P(k) より)
    = k(k-1)/2 + (k+1)
    = (k+1)k/2

数学的帰納法のバリエーション (累積帰納法)

- 「任意の自然数 n について P(n)」を示したければ、以下を示せばよい
  - P(0) かつ P(1) かつ ... かつP(k) ならば P(k+1) (任意の k について)
    - k 以下については P が成立しているとしてよい

累積帰納法を使った
互除法の正しさの証明

「任意の n, m, k について n = m + k ならば、m, k を入力とした互除法の出力は m, k の GCD」

(証明) 累積帰納法による。m ≧ k の場合を示す。

m÷k の余り j について場合わけ:

(1) j=0の場合、出力 k は確かに m, k のGCD

(2) j>0の場合、帰納法の仮定 P(k+j) より、k, j を入力とした互除法の出力は k, j の GCD。
先般の議論より k, j のGCD = m, k のGCD。
m < k の場合も同様。(証明了)

今日のメニュー

- 前回の積み残し
  - アルゴリズムの正しさ
- プログラムとプログラミング言語
- JavaScript プログラミング入門
  - とにかく動かしてみよう

プログラム

コンピュータへの命令書

- 機械語プログラム・バイナリ(binary)プログラム
  - ビット列としてメモリに格納され、ハードウェアの動作を制御する
- ふつうはプログラミング言語(プログラムを書くための人工言語)で書かれる
  - アセンブリ言語
  - 高水準言語
    - 言語なので文法・意味がある!

アセンブリ言語

機械語命令に*人間が理解*しやすい名前をつけたもの

- 01001000 を ADD X, Y と呼ぶ、などなど
- ADD X, Y から 01001000 に戻す(簡単な)「翻訳」が必要 → アセンブラ

高水準プログラミング言語

より人間よりの表記でコンピュータへの指示を記述

- X + Y - Z など、演算が入れ子になった式が使える
- 整数、文字などのデータの種類の区別
- 処理のまとまりに名前をつけて「まとまりとして」使うことができる

など

- 機械語への「翻訳」が大変
  - ソースプログラムとターゲットプログラム
- ハードウェアの差異を吸収できる(プログラム再利用)

歴史に名を残した高水準言語(ごく一部)

- FORTRAN (1954) … 数値計算への応用
- Lisp (1958) … 記号処理への応用
- COBOL (1959) … 事務処理への応用
- ALGOL (1958) … アルゴリズム記述用
- Simula (1954)
  … シミュレーション記述・オブジェクト指向
- Prolog (1970)
  … 人工知能への応用・論理プログラミング
- C (1971) … オペレーティングシステム記述
- Smalltalk (1971) … オブジェクト指向
- ML (1973) … 証明戦略記述・関数プログラミング

高水準プログラムの実行方式

- インタプリタ(解釈実行系)による実行
  - 高水準プログラムを読み込みながらその意味通り実行するプログラム
  - 同時通訳のようなもの？
- コンパイラによる実行
  - 高水準プログラムを予め機械語プログラムに翻訳
  - 翻訳書のようなもの？
- ふたつの方式のハイブリッド:
  - 中間水準言語に翻訳して解釈実行
  - 解釈実行しつつ、何度も実行する重要箇所はコンパイル

本講義でふれるプログラミング言語

- JavaScript
- OCaml (予定)
- Prolog (予定)

今日のメニュー

- 前回の積み残し
  - アルゴリズムの正しさ
- プログラムとプログラミング言語
- JavaScript プログラミング入門
  - とにかく動かしてみよう

JavaScript

- ウェブページで「たのしいこと」をする用途で発明
- 当初は誰も大規模なプログラムを書かない「おもちゃ言語」と認識されていた
- Google が GMail を発表して皆腰をぬかした
- 多くのウェブブラウザで動く
  - ただしブラウザ毎に少しずつ動作が違う ;-(
- (Java 言語とはほとんど関係ない)

講義中に見せるプログラムについて

- 講義ホームページで該当するリンクをクリックするとプログラムが動く
- ブラウザの「ページのソースを見る」で閲覧
- 「ページを保存」すれば保存・編集も可能
- 編集したファイルは「開く」で読み込める

プログラムその1

- 機能：ページを読み込むと、メッセージ“Hello, world”を確認ダイアログボックスに表示
- (web ページ自体は空っぽ)

ページのソース

```
<html>
  <body>
    <script>
alert("Hello, world");
    </script>
  </body>
</html>
```

JavaScript プログラムはHTML中の
<script>～</script>
に埋め込まれている

ウェブブラウザの動作

- HTML部分を表示
- <script>～</script> 部(複数あってもよい)を前から順に JavaScript プログラムとして実行

プログラム大解剖

```
alert(“Hello, world”);
```

- JS プログラム = 文(statement)の列
  - 文は(たいてい)セミコロンで終わる
  - これは、ひとつの文からなるプログラム
- 文の種類もいろいろある
  - これは手続き呼出文: 〈手続名〉(〈式〉);
  - 手続alertの機能…括弧内に書かれた式の値を確認ダイアログボックスに表示し OK が押されるのを待つ

文と式

- 文: コンピュータへの命令となる構文単位
- 式(expression): 計算をして値(value)を得るための構文単位
  - 整数定数(..., -1, 0, 1, 2, ...)や文字列定数 “**`Hello, world`**” は式
  - (複数の)式を演算子で組み合わせたもの
    - 1 + 4

プログラムその2

- 機能: キーボードから名前の入力を促し、入力された名前の人に挨拶をする

```
var username =
  prompt("What's your name?");
alert("Hello, " + username);
```

変数宣言・初期化文
`var`〈変数の名前〉`=`〈式〉`;`

- 変数…値をしまっておくための箱
  - 言語によっては、値につける名前、という考え方も
- 変数宣言文…新しい変数を用意する
- 初期化…〈式〉の値を箱に入れる
  - JS では初期化(等号以下の部分)は省略可
- 変数参照…変数の中身の値の取り出し
  - (多くの言語では)宣言せずに参照してはいけない

- 関数呼出**式** 〈関数名〉(〈式〉)
  - 関数 **`prompt`** の機能…〈式〉の値をダイアログボックスに表示し、入力を待つ･入力された文字列を式全体の値とする
    - 「入力された文字列を返す(return)」ともいう

演算子 +

- 加算
  - 式 **1+1** の値は **2**
- 文字列の連結
  - 式 **“Hello, ” + “Igarashi”** の値は **“Hello, Igarashi”**
- 整数と文字列を「足し」たら? (**“Hello,” + 2**)
  - JSでは: **“Hello,2”**
  - 言語によってはエラーになる
    - 実行時エラー or 実行前エラー

プログラムのフォーマットについて

- 空白文字と改行は区別されない
  - ただし全角空白は使ってはいけない
- 英文字の並びの間、ときには記号間の空白の有無は重要
  - var と username の間の空白は重要
  - 個数はどうでもよい
- 行頭の空白(字下げ、インデント)は読みやすさのため
- // から行末まではコメントと呼ばれプログラムの実行には影響しない

プログラム その3

- 機能: キーボードからふたつの数を入力させ、その平均を表示する

```
function average(x, y) {
  return (x + y) / 2;
}

var a = Number(prompt(...));
var b = Number(prompt(...));
var c = average(a, b);
alert("The average is " + c);
```

関数･手続き定義

- 文の並び､式に名前をつける
  function 〈名前〉**(**〈変数列〉**)**
  {〈文の並び〉**}**
- 呼び出し側から渡される情報(引数(ひきすう))を格納する変数の宣言
- alert などと同様に呼び出せる
- return 文(**return** 〈式〉**;**)で〈式〉の値を呼び出し元に返す
- (訂正: プログラム = 文と関数定義の列)

その他

- 除算演算子 /
- 文字列を数値に変換する Number 関数

プログラムその4

- 機能: キーボードからふたつの数を入力させ、その最大公約数を表示する

- 最大公約数を計算する関数の定義
  - % は余りを計算する演算子

```
function gcd(x, y) {
  var z = x % y;
  return (z==0) ? y : gcd(y, z);
}
```

条件分岐式と比較演算子

- 条件分岐式〈式1〉?〈式2〉:〈式3〉
  …〈式1〉(の値)の真偽で場合分け、真なら〈式2〉の値、偽であれば〈式3〉の値が式全体の値
- 比較演算子(値が真偽になるような式を構成する)
  - **==** … 両辺が等しい
  - **!=** … 両辺が等しくない
  - **<** … より小さい、**>** … より大きい
  - **<=** … 以下、**>=** … 以上

ちょっと脱線: 条件文

- 「～が真なら～をする、偽なら～をする」という文レベルでの条件分岐もできる

```
if (〈式〉) {〈文の並び〉}
    else {〈文の並び〉}
```

  - セミコロンで終わらない文なので注意

- `return (z==0) ? y : gcd(y,z);`
  は

```
if (z==0) { return y; }
  else { return gcd(y,z); }
```

  でもOK

ユークリッドの互除法

入力: 自然数n, m (ただし n ≧ m)

1. k を n ÷ m の余りとする
2. k について場合分け
   - k = 0 ⇒ m を出力として終了
   - k ≠ 0 ⇒ m, k を入力として互除法を行いその出力をそのまま出力とする

再帰(recursion)

プログラムその5

- 機能: キーボードからふたつの数を入力させ、その最大公約数を表示する

```
var x = Math.max(m, n);
var y = Math.min(m, n);
var z = x % y;

while (z != 0) {
  x = y;  y = z;  z = x % y;
}
```

繰り返し構文 while

while (〈式〉**) {**〈文の並び〉**}** …

- 〈式〉が真である限り〈文の並び〉を繰り返し実行する
- 〈式〉が偽なら何もしない

代入文: 〈名前〉 =〈式〉;

- 変数の(箱の)中身を更新する
  - (初期化を伴う)変数宣言文との違いに注意

```
while (z != 0) {
   // この時点での y, z を
   // それぞれ新たな x, y とする
   x = y;  y = z;  z = x % y;
}
```

ここまでのまとめ

JS プログラム = 関数･手続き定義と文の列

- 文 … 命令の単位
- 式 … 値を計算するひとまとまり
  - いろいろな値の種類(数値、文字列、真偽値)
- 変数 … (式の)値を格納する箱
  - 宣言･初期化･参照･代入
- 関数･手続定義 … ひとまとめの処理に名前をつける
- 条件判断と繰り返し