プログラミング論I
(6) 復習，課題演習に向けて

電子情報工学専攻　日浦 慎作

ログインしておいてください

プログラミングのコツ

- プログラムは，証明ではない．手順である
  - プログラムで `a=b;` と書いても，aとbがずっと同じであることを意味しない．この数式に出会った瞬間に一度だけ，bの値がaに代入されるだけ．
  - `if`文や`for`文により，計算手順を分岐したり，繰り返すことができる．

- 要するに，手計算や電卓でやる方法を考え，その手順を書いていけば良い．
- 計算だけでなく，数値の入力・表示のタイミングや，数値をどのような変数（入れ物）に保存していくか，もよく考えるとよい．

よくあるミス

- 代入 = と，比較 == を間違えている
- scanf で，変数に & を付け忘れている
- 不要なところに ; が付いている

例

```
for(i = 0; i < 10; i++) ;
{
   printf(“Hello world¥n”);
}
```

Hello world は1回しか表示されない

ミスを防ぐには？

- 段付けをしっかり，まちがいなくつける
- カッコの有効範囲や対応関係に気をつける
- エラーメッセージをよく読む
- ¥n や %d などが適切かどうか注意

どうしても動きがおかしいときは？

- 途中に printf を挟んで，動きを見る

```
int i, sum = 0;
for(i = 1; i < 10; i++) {
      sum += i;
      printf(“i = %d, sum = %d¥n”, i, sum);
}
printf(“sum = %d¥n”, sum);
```

「1 から 10 までの総和を求めよ」という問題に対し，
正解が55になるべきところ，上のプログラムでは45になってしまう．

実行結果

```
i = 1, sum = 1
i = 2, sum = 3
i = 3, sum = 6
i = 4, sum = 10
i = 5, sum = 15
i = 6, sum = 21
i = 7, sum = 28
i = 8, sum = 36
i = 9, sum = 45
sum = 45
```

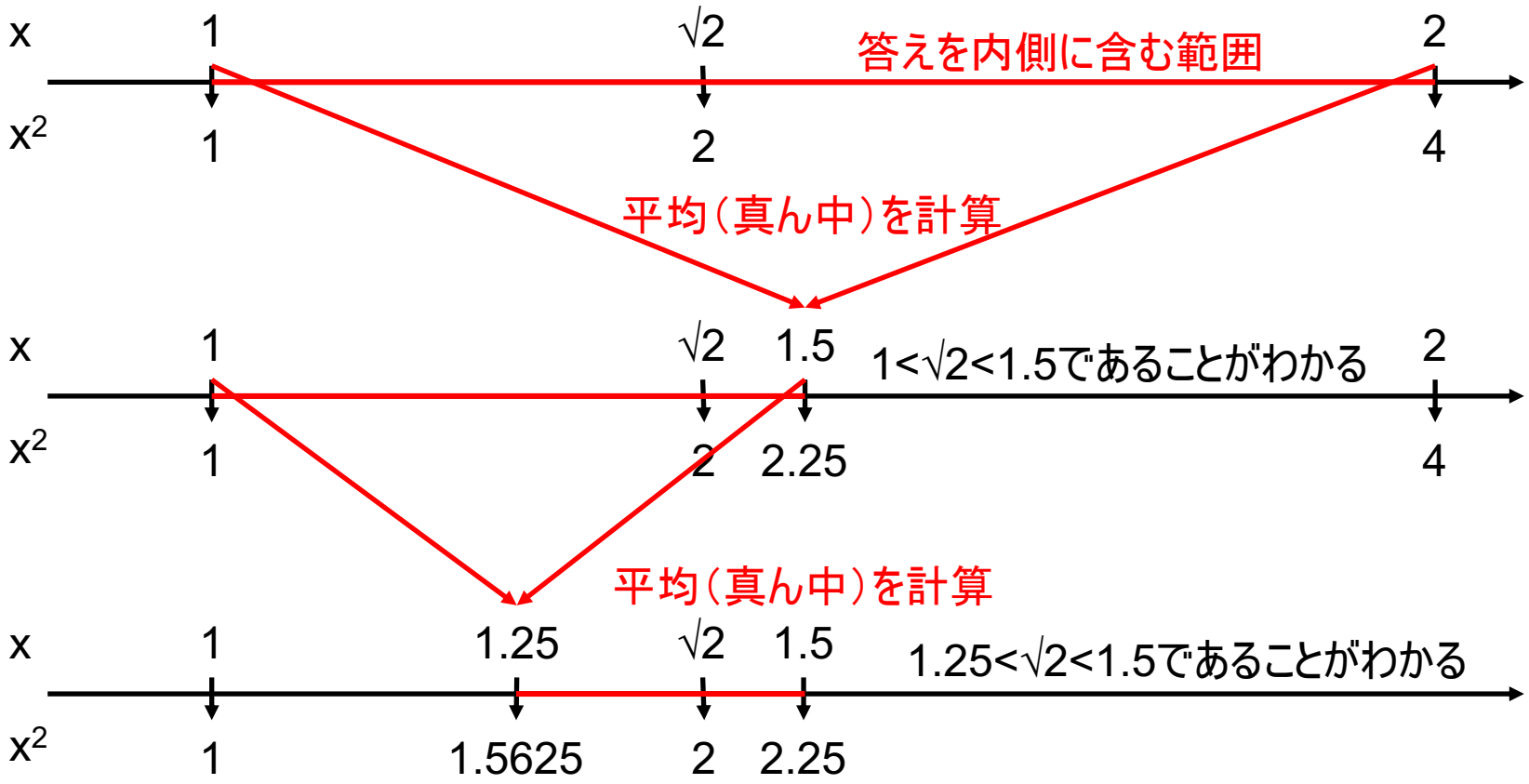
- 1 から 9 までしか計算できていないことが判明.

ちょっと複雑なプログラミングの例

<u>平方根を計算してみよう</u>

- 自分で平方根を計算するにはどうする？
  - 二乗した値が大きいか，小さいか確かめて，
    徐々に数値を調整していく方法が考えられる．
- 実行例（2の平方根の計算）
  1. $2^2 = 4 > 2$ だから，$\sqrt{2}$は2より小さい
  2. $1^2 = 1 < 2$ だから，$\sqrt{2}$は1より大きい
  3. $1.5^2 = 2.25 > 2$ だから，$\sqrt{2}$は1.5より小さい
  4. $1.25^2 = 1.5625 < 2$ だから，$\sqrt{2}$は1.25より大きい
  5. $1.375^2 \fallingdotseq 1.89 < 2$ だから，$\sqrt{2}$は1.375より大きい

数直線で表すと



- この手順をプログラミングしてみよう！

まずは日本語で

1．数値を入力させる(変数inputに入れる)
2．数値を挟み撃ちする区間の，左端と右端の値を決める
   1．左端(小さい方)は 1，右端(大きい方)はinputにしておく
3．区間の中央の値(平均)を求める(これをcenterとする)
4．inputがcenterの二乗より大きいかどうか調べる
   1．大きい場合は，区間の右側(大きい側)に答えがあるので，
      左端の値をcenterに変更する
   2．小さい場合は，区間の左側(小さい側)に答えがあるので，
      右端の値をcenterに変更する
5．3.から繰り返す

プログラム例

```
#include <stdio.h>

//1より大きい数の平方根を計算するプログラム
int main(void) {
  double input, lower, upper, center;
  int i;

  printf("Input a number (> 1) :");
  scanf("%lf", &input);

  lower = 1; //区間の左端の初期値は1
  upper = input; //区間の右端の初期値は，入力値
  for(i = 0; i < 30; i++) {
    center = (lower + upper) / 2.0; //区間の真ん中（平均）を求める
    printf("%f¥n", center);
    if(center * center < input) { //答えが平均の二乗より大きい
      lower = center; //区間を右側へ（大きい方へ）短縮する
    }
    else {
      upper = center; //区間を左側へ（小さい方へ）短縮する
    }
  }
  return 0;
}
```

実行結果

```
Input a number (> 1) : 2
1.500000
1.250000
1.375000
1.437500
1.406250
1.421875
1.414062
1.417969
1.416016
1.415039
1.414551
1.414307
1.414185
1.414246
1.414215
1.414200
1.414207
1.414211
1.414213
1.414214
1.414214
1.414213
1.414214
1.414214
1.414214
1.414214
1.414214
1.414214
1.414214
```

1.41421356

```
Input a number (> 1) :3
2.000000
1.500000
1.750000
1.625000
1.687500
1.718750
1.734375
1.726562
1.730469
1.732422
1.731445
1.731934
1.732178
1.732056
1.731995
1.732025
1.732040
1.732048
1.732052
1.732050
1.732051
1.732050
1.732051
1.732051
1.732051
1.732051
1.732051
1.732051
1.732051
```

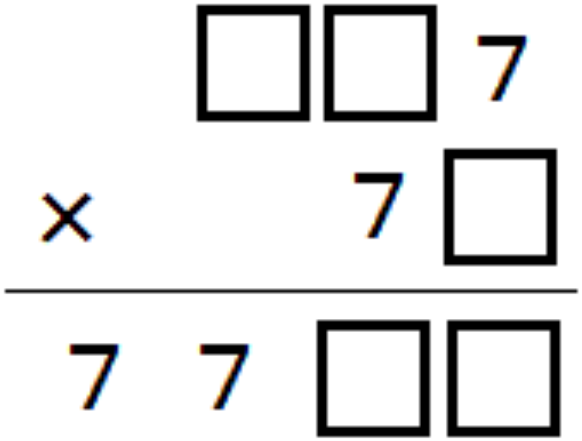
1.7320508

参考情報（余談）

この計算方法では，1回計算するごとに区間の大きさが半分になる．つまり，繰り返し回数をnとすると，$1/(2^n)$ の精度で答えが求められる．

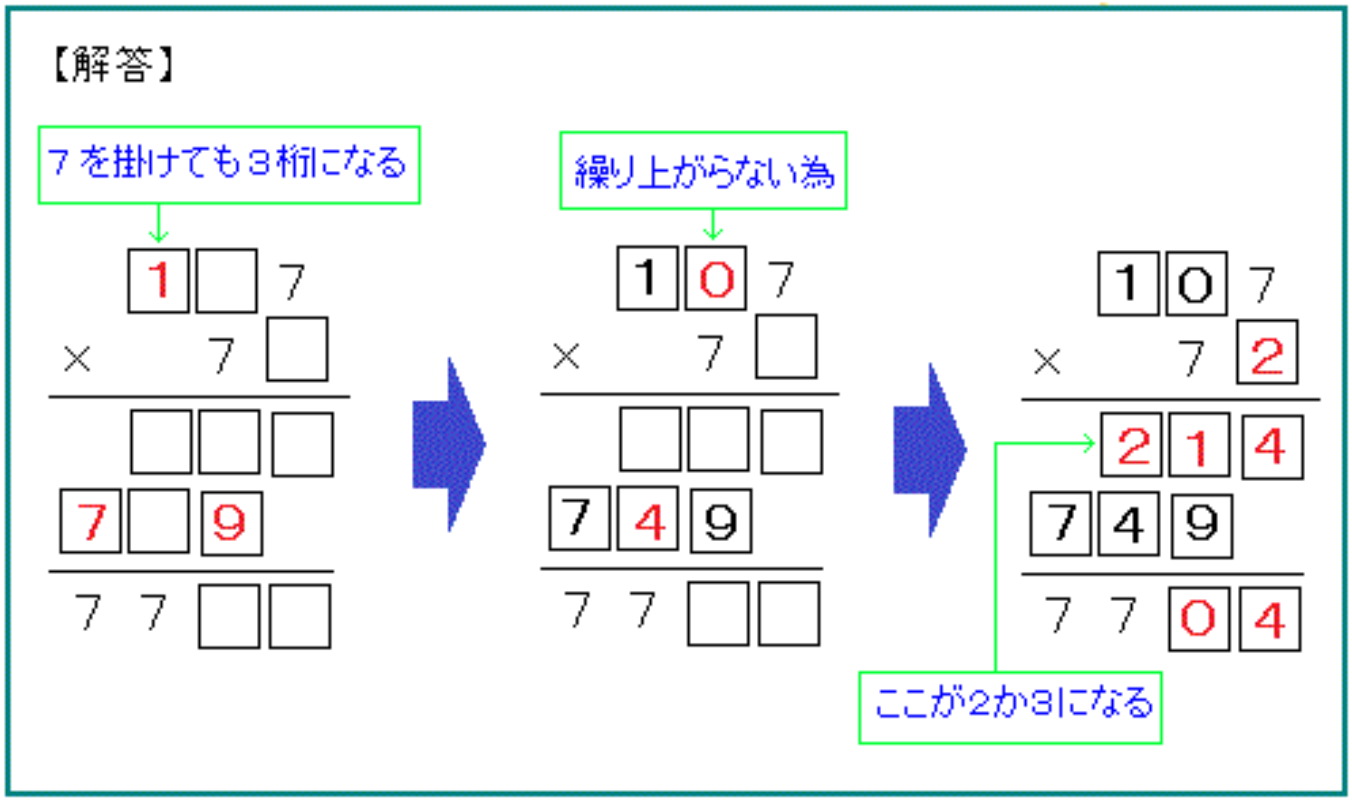
$2^{10} = 1024 \fallingdotseq 10^3$ なので，10回繰り返すごとに10進数で3桁分の精度向上がある．30回繰り返すと，約9桁の精度で計算ができる．

虫食い算を解いてみよう



- ルール
  - □には1桁の数字が入る
  - 最上位桁は 0 ではならない

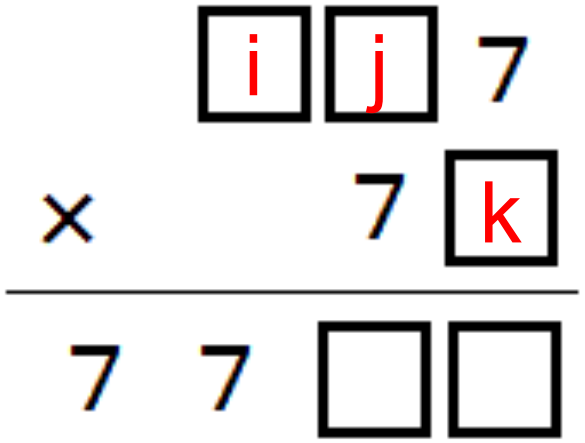
人間による解き方



- 総当りは無理なので，決まるところから決めていく

参考（問題の出典） http://task.naganoblog.jp/e1694679.html

コンピュータでの簡単な解き方



□に数値を順に入れてみて，成立するものを探す

- i は 1～9の値，j と k は 0～9の値を，すべて試す
  - ３重のforループになる
- １行目は 100i + 10j + 7 で求められる
- ２行目は 70 + k で求められる

- それらを掛けた結果が，77□□ならよい
  - つまり 100で割った値が77ならよい（整数演算は切り捨て）

プログラム例

```
#include <stdio.h>

int main(void) {
  int i, j, k, l, m, n;

  for(i = 1; i <= 9; i++) { //最上位桁は0はダメ
    for(j = 0; j <= 9; j++) { //2桁目以降は0でも良い
      for(k = 0; k <= 9; k++) {
      l = 100 * i + 10 * j + 7; //1行目の値
      m = 70 + k; //2行目の値
      n = l * m; // 積を求める
      if(n / 100 == 77) { //積が 77xx かどうか確認
        printf("%d * %d = %d¥n", l, m, n);
      }
      }
    }
  }
  return 0;
}
```

結果

```
107 * 72 = 7704
```

実行時間は 0.001秒でした

課題演習について（告知）

- 課題演習の座席表
  - 座席指定です．１回の部屋も使います．
- ルールの確認

ここから

- 過去の課題の解説をします
- 残り時間は，演習問題の残りをやったり，
  試験準備の資料作成に使えます