プログラミングI
数理物理，総合理学等向け

2018年11月12日 その2

海谷 治彦

目次

- プログラム作成の取り組み方
- main 中の return の意味.
- プログラム開発の三段階
  - 理解，整理，翻訳
- 演習問題

分からない問題は，
まず紙に書いて考えよ！

いきなりキーボードを叩くな！

前提

- 問題(仕様といってもよい)は通常，文章で提示される.
- 文章を読んで，適当にプログラムを書き始めれば，プログラムができるというわけではない.
  - プログラムに慣れている人なら，それでもなんとかなるが.
- 以下の手順が必要.
  1. 文章を読んで，問題の大まかな区切りを把握する.
  2. 区切り毎に，処理手順を考える.
  3. 教科書等や授業でやったことから，処理手順に似た部分をパクって当てはめる.

やってはいけないこと

- とにかくキーボードで何か入力してみる.
- いままで習った文法を，とりあえず，適当にならべてみる.
- 処理の区切りを認識せずに，ぐちゃっと，一塊でプログラムを書こうとする.
- 一つの変数を異なる複数の目的で使う.

数回前より

本日の演習5

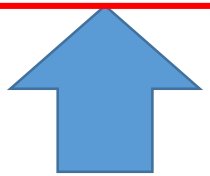
- 0より大きい整数値 x を入力すると，0以上x以下の奇数の値全ての合計値を計算するプログラムを作成せよ.
- oddsum.c というファイル名で提出せよ.

期待される結果の例

入力	結果
10	25
9	25
2	1
0	無言，もしくはエラー警告
101	2601

1. 問題の大まかな区切り

1. 整数 x を入力する.
2. x は 0 より大きい値.
3. 0以上 x以下の奇数.
4. 合計値を計算.
5. 表示 (書いてないけど)



- 最低限，上記のような項目は列挙して書く.
  - 紙に手書きでOK

2-1 各ステップの処理を考える

1. 整数 x を入力する.
   - scanf とかならったなぁ.
   - 教科書には gets + atoi とかかいてあるなぁ.
2. x は 0 より大きい値.
3. 0以上 x以下の奇数.
4. 合計値を計算.
5. 表示 (書いてないけど)

2-2 各ステップの処理を考える

1. 整数 x を入力する.
2. x は 0 より大きい値.
   - if文とかならったなぁ.
   - x>0 じゃなければ何もしなくていいのか. (結果例より)
3. 0以上 x以下の奇数.
4. 合計値を計算.
5. 表示 (書いてないけど)

2-3 各ステップの処理を考える

1. 整数 x を入力する.
2. x は 0 より大きい値.
3. 0以上 x以下の奇数.
   - 奇数って何だっけ？ 1, 3, 5, 7 …
   - 0 <= 奇数 <= x また if 文か？複数あるから while か？
   - 奇数は 2 で割ると 1 余る.
   - もしくは，初期値を1にして，2を順番に足してく
4. 合計値を計算.
5. 表示 (書いてないけど)

2-4 各ステップの処理を考える

1. 整数 x を入力する.
2. x は 0 より大きい値.
3. 0以上 x以下の奇数.
4. 合計値を計算.
   - 変数に累積していけばいいんだな.
   - 数列 a_n の合計値 S_n は, $S_0 = 0$ を初期値として,
   - $S_n = S_{n-1} + a_n$ だったな.
   - $S_n = S_{n-1} + a_n$ は, 単純に s=s+累積する値; という代入文でいいな.

   具体例:

a_n	1	3	5	7	9	・・・
S_n	0+1 → 1	1+3 → 4	4+5 → 9	9+7 → 16	16+9 → 25	・・・

5. 表示 (書いてないけど)

2-5 各ステップの処理を考える

1. 整数 x を入力する.
2. x は 0 より大きい値.
3. 0以上 x以下の奇数.
4. 合計値を計算.
5. 表示 (書いてないけど)
   - printf とか習ったな・・・

ここまで,
紙等に考えを書き下す.

以降，段階的に
プログラムにする

1. 整数 x を入力する

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  return 0;
}
```

2. x は 0 より大きい値

```
#include <stdio.h>

int main(void){
int x;
   scanf("%d", &x);
   if(x > 0){
      // この場合だけ処理する
   }
   return 0;
}
```

3. 0以上 x以下の奇数 その1

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  if(x > 0){
  int i=1;
    while(i<=x){ // 1, 3, 5, 7 ･･･と列挙される
      // ここでのiが合計対象
      i=i+2; // i+=2; でもOK
    }
  }
  return 0;
}
```

3. 0以上 x以下の奇数 その2

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  if(x > 0){
  int i=1;
    while(i<=x){
      if(i%2 == 1){ // 一手間増えてるが
        // ここでのiが合計対象
      }
      i=i+1; // i++ でもOK
    }
  }
  return 0;
}
```

4. 合計値

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  if(x > 0){
  int i=1, s=0;
    while(i<=x){
      s = s+i; // s+=i; でもよい.
      i=i+2;
    }
  }
  return 0;
}
```

5. 表示 (完成)

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  if(x > 0){
  int i=1, s=0;
    while(i<=x){
      s = s+i;
      i=i+2;
    }
    printf("%d¥n", s);
  }
  return 0;
}
```

5. 表示 %を使う版 (完成)

```
#include <stdio.h>

int main(void){
int x;
  scanf("%d", &x);
  if(x > 0){
  int i=1, s=0;
    while(i<=x){
      if(i%2==1){
        s = s+i;
      }
      i=i+1;
    }
    printf("%d¥n", s);
  }
  return 0;
}
```

考察

1. 教えてないCの文法は一つも無い.
2. 奇数や合計値等，問題を解くための知識も高校生(中学生)並みのものばかり.
3. 前述のように段階的に書いていけば，論理も別に難しくない.

4. 毎回，諸君のプログラミング作業を見ると，何も紙に書かないでプログラムを作ろうとしてる.
5. そのため，問題中の「区切り」が把握できてない.
6. 問題の区切りに関係なく，前からプログラムを書こうとするので，何のための処理かわからない，わかっていないように見える.

再掲載

分からない問題は，
まず紙に書いて考えよ！

いきなりキーボードを叩くな！

タブ

- 標準的にはスペース8個分.
- 段付けが見やすくなる.

```
#include <stdio.h>

int main(void){
int x, i=1, s=0;
        scanf("%d", &x);
        if(!(x>0)){
                return 1;
        }
        while(i<=x){
                s=s+i;
                i=i+2;
        }
        printf("%d¥n", s);
        return 0;
}
```

main中 の return

- C言語のmain関数中において，return 命令を呼ぶと，プログラムは終了する.
  - main以外の関数では終了はしない (詳細は後日).
- return は整数の引数を必要とする.
- 引数の意味は以下のような慣例がある.
  - return 0 プログラムの実行が成功した場合.
  - それ以外 プログラムの実行が失敗した場合.

- 参考:
  - この return が返す値は，OS(WindowsやLinux)が利用する場合がある.
  - プログラムの実行結果の成否によって，OSが次の行動を変える場合があるため.

例

```
#include <stdio.h>

int main(void){
int x, i=1, s=0;
  scanf("%d", &x);
  if(!(x>0)){ // xが0より大きくなければ即終了
    return 1; // この命令によって以降には行かない
  }
  // 以下，xが0より大きい場合の処理
  // {} の入れ子が一段浅くなるので私は好き
  while(i<=x){
    s=s+i;
    i=i+2;
  }
  printf("%d¥n", s);
  return 0;
}
```

なぜプログラムは難しい？

- なぜプログラミングは難しいか？

以下のどこかでつまづいている！

1. コンピュータに行わせたいことを理解
2. 理解したことを説明できるレベルまで整理
3. コンピュータにわかる言葉に翻訳

なぜ、あなたはJavaでオブジェクト指向開発ができないのか
—Javaの壁を克服する実践トレーニング
小森 裕介 (著), エスエムジー株式会社 (著)
技術評論社 ; ISBN: 477412222X ; (2004/12) 2289円

プログラミングとは何か？

- コンピュータにやらせたいことの手順を，コンピュータのわかる言葉で書く.

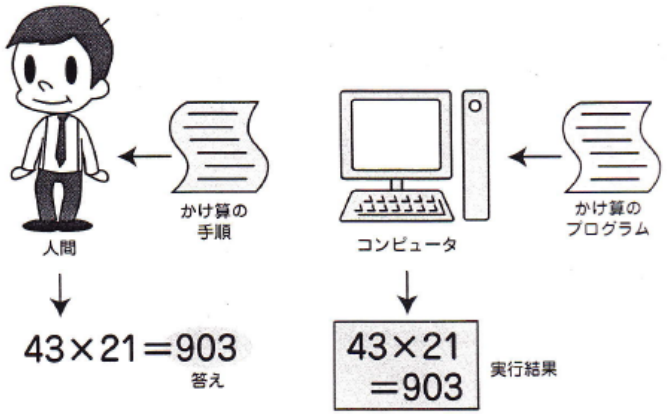
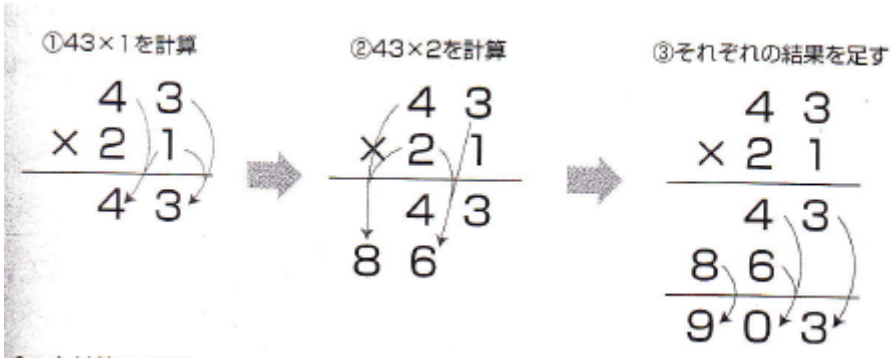
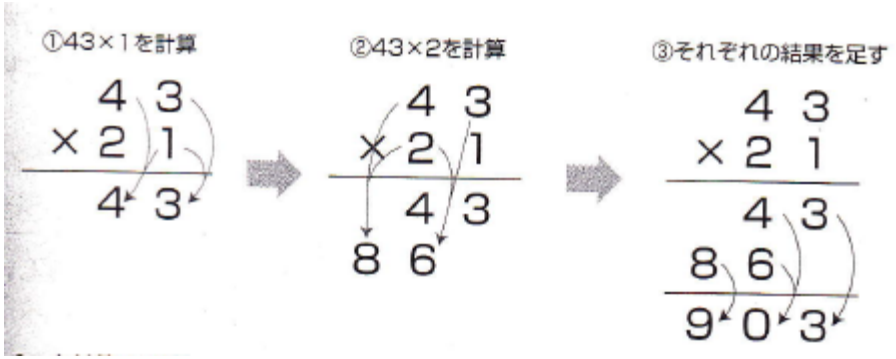


図1-2　かけ算とプログラムの対比



理解の失敗例

- もし，以下にあるような掛け算の手順を追い，意味が分からなければ，掛け算のプログラムはできない！



- もし，銀行業務が理解できなければ，その業務支援ソフトウェアは作れない.

何故，考案ではなく理解か？

- コンピュータにやらせたいことの多くは，現実世界の業務や手順の一部である.
  - 放射性物質の飛散予測の計算の一部（全部）
  - 銀行業務の一部
- そのような業務や手順は，その道の専門家が考案する.
  - 原子力専門家，物理学専門家，気象学専門家
  - 銀行員
- 我々，コンピュータ技術者は，これらを理解し，計算機で(どれだけ)肩代わり可能か判断する.
  - 判断のためには理解が必須！

整理の失敗例

- 個々の掛け算の計算手順はわかるが，それを一般化（整理）して，他人に説明できない．
  - 例えば，算数や数学にあるように N や x みたいなパラメータを使って，計算手順を一般化できないとか．
- 銀行業務も個々の決済等の業務を一般化した手順として書けなければ，ソフトウェアを作れない．

整理のポイント

現実業務と計算機が可能なことのバランスが重要

- 現状の計算機(プログラム言語)で実現不可能な整理をしてもシステムはできない.
  - 微分方程式で飛散予測を整理できても，それをコンピュータで直接実行はできない.
- 業務と大きく剥離した形で整理しても，そもそも整理されているか確認しようが無くなる.
  - Cプログラムを直接見せられても分からない物理学者も多いだろう.
  - 結果として，計算機にやらせたいことが整理されているか，確認しようが無くなる.

説明相手は誰？

基本的に相手は二種類を想定する.

- 計算機
  - 計算機にやらせることを想定しているのだから，計算機に説明することを想定しないと.
  - 曖昧さや，直観は通じない.
- そもそもの業務専門家
  - コンピュータ技術者の理解が合っているかは，往々にして，専門家じゃないとわからない.
  - 専門家は人間だから，基本，長大なプログラムやアルゴリズム記述の意味はわからない.

整理に創造は必要か？

- 現実的には，どう整理するかについて，新しい整理法や整理様式を新規に創造する必要は無い．
  - 情報科学科では，このような創造をめざしている人もいる．
- 理解したことを，既存の様式を真似て，整理する技術をまずは覚え使えるようにすること．
- 例
  - PCのフォルダや組織階層等，階層的な構造(木構造)を整理するにはよく知られた様式が既に存在する．

    (コンポジットパターンと呼ばれる，ググってみて)
  - この様式を超える整理法を創造するのは多分無理．
- 既存の整理法を数多く知った上で，より良い整理法を創造することを最終的には目指してほしい．

翻訳の失敗例

- ずばり，プログラム言語を知らない．
  - この辺を補う図書は腐るほど出版されている．
  - 所謂，プログラム言語の授業は結構，この辺のみが重視されている．
- 言語を知っていても，一般化した手順の記述との対応が分からない．
  - 配列，リスト，スタック，木等は知っていても，それが「整理されたコンピュータにやらせたいこと」の何に対応するか分からない．
- 日本語を知っていても，コミュ力が無い人が居るのと同じ．
- Cの文法を知っていても，プログラム化する分野の理解法，整理法は身につかない．

本当に難しいのは理解と整理

- 理解
  - 株取り引きの業務が理解できてますか？
  - ゲーム内での3D表示の人間の描画法を理解できてますか？
  - SPEEDI (スピーディ)の放射性物質の飛散予測法を理解できますか？
  - 迷惑メールのみを除去する手法を理解できますか？
- 整理
  - 上記を個々の事例ではなく，一般化して整理できますか？（整理する既存法を知ってますか？）
  - 整理したことを人間と計算機の双方に都合よく説明できますか？

具体的に役立つ実践

- 理解
  - 具体例を複数，見てみる，書いてみること．
  - 前回の演習なら，具体的に入力が7の場合，2の場合，10の場合等の処理を手で書いてみる．
- 整理
  - 具体例を比較し，差異を認識すること．
  - 有名な整理法(アルゴリズム，パターン，イデオムとかいわれる)を覚えること．
    - 現代的なプログラミングはある意味，暗記物．
  - 前回の演習なら，

「奇数の数列」や

「数列の合計値の計算」を考える．

計算機屋は下請けか？

- 大筋で YES
  - 対象分野の一部もしくは全部を肩代わりする下請け業務.
- しかし，計算機技術により業務・生活の方が変化することも最近は多い.
  - 技術主導で世の中のあり方・やり方が変わる.
  - 例
    - 検索技術の発達による情報整理法の変化.
      - 図書館等 VS Webやデスクトップ検索
    - 携帯端末の発達による待ち合わせ法の変化.
      - いまどき，きっちり場所と時間を指定しなくても集まれる.
    - SNSの発達により，グループ，伴侶，家族等の在り方の変化.

昨年度の問題

本日の演習

- 2以上の正の整数を入力すると，その整数以下の素数の中で最も大きいものを表示するプログラムを作成せよ.
- 素数は2以上の整数とする.
- 入力に問題がある場合や解が無い場合には，main関数は0以外の数を返すようにせよ. (return 0; ではないということ)
  - 何か警告を出してもよいし，出さなくてもよい.
- 以下，期待される結果の例.

入力	出力
14	13
2	2
-5	解なし
863	863

理解

1. 素数って何だっけ？
   - コレは算数を思い出す，もしくはググる.
2. 数値を入力するっていうけど，入力法は？
   - C言語の入力法を思い出す.
3. 入力したものが2以上の整数であるには？
   - 整数であるとは？
   - 2以上であるとは？
4. 入力に問題がある場合の対処法？
   - 上記3に当てはまらない場合をどう探す？
   - C言語でのreturnを思い出す.
5. 入力数値以下の最も大きな素数ってどう求めるの？
   - 入力された数値から，順に素数かチェックして，ダメなら数値を一個小さくすればいいんじゃない？

整理

1. 入力値をとにかく2以上の整数に限定する. それ以外は「入力に問題があり」とみなす.
2. 入力された数値 N から順に N, N-1, N-2 … 2 と素数か否かをチェックする.
3. ある数 X が素数であるなら, その数より小さい数(X-1, X-2 … 2)で割った余りが全てゼロでない.
   ⇒ 余りゼロのものがあったら素数じゃない.

4. 上記を具体的な数値で試してみる. たとえばN=8の場合等.

翻訳

1. 整理1に対応するプログラムを書く.
   - gets, fgets, scanf 等，いろいろありましたね.
   - 数値が2以上かどうかは単にif文で判断.
2. 整理2に対応するプログラムを書く.
   - 入力値Nから降順(N, N-1, N-2, … 2)にループ.
3. 整理3に対応するプログラムを書く.
   - ある数Xが素数かも，2からX-1までループして，割ったあまり(%演算)を使って吟味する.
4. 2のループ中で，3の素数チェックをする.

解答例

```
// maxprime.c
#include <stdio.h>

int main(void){
int n, i;
  scanf("%d", &n); // とりあえず数値を得る

  if(!(n>=2)){return 1;} // 2以上ではないとすると終わり

  i=n;
  while(i>1){ // n, n-1, n-2 ... 2 と素数の候補i
  int j, modzero=1; // 最初は候補が素数だと仮定. modzero==0 なら割り切る数があるとみなす
    j=i-1;
    while(j>1){ // 候補iより小さい数jで
      if(i%j==0){ // 候補iを割る余りがゼロなら
        modzero=0; // 候補iは素数じゃない
        break; // コレは今回無くてもOK. でもあったほうが効率的
      }
      j--;
    }
    if(modzero){ // modzero==1のままなら割り切れた数は無い
      printf("%d¥n", i); // 答えを表示して
      break; // iのループを抜ける
    }
    i--;
  }
  return 0;
}
```

演習8 総合演習 (1/3)

- 図書には ISBN というコードがついており，10桁のコードがついている．(ISBN10と呼ばれる)
  - 最近は13桁のコードもあるがここでは扱わない．
- このコードは，書き間違え等を防止するため，以下のような細工がしてある．
  - N桁目の数値とNを掛けた数値の総和が11の倍数になるようにする．
  - 10をXと表す．
- 例えば，
  - 1580087736 は，
    $10\times1+9\times5+8\times8+7\times0+6\times0+5\times8+4\times7+3\times7+2\times3+1\times6 = 11 \times 20$
  - 487408852X は，
    $10\times4+9\times8+8\times7+7\times4+6\times0+5\times8+4\times8+3\times5+2\times2+1\times10=11 \times 27$

である．

段階を追う (2/3)

- ISBNを入力すると，入力ミスが無いか否かを前述の頁の規則に基づくプログラムを作成せよ.
- プログラムは以下の6段階を経て作成せよ.
- 各段階の6個のプログラム全てを提出せよ.

1. キーボートからISBNを表す文字列を入力し，それを画面に表示する. (isbn1.c)
2. 入力された10個の文字を整数に変換した結果を表示する. (isbn2.c) Xは10とする.
3. 入力された文字が 0から9の数値とXのみじゃない場合，警告を表示してプログラムを終了する. (isbn3.c)
4. 3.に加え，N桁目の数値とNを掛けた数を10個表示する. (isbn4.c) Xは10とする.
5. 4.で表示した数の総和を表示する. (isbn5.c)
6. 5.で計算した総和が11の倍数か否かを判定する. (isbn6.c)

isbn2.c isbn3.c に関するヒント (3/3)

- 文字として入力された数字をプログラム中の整数に変換する場合，以下の処理でよい．(ctoi.c)
- 他の方法を知っている者はソレを使ってもよい．
- X については，最初に if 文等で，場合分けして，10と判定してよい．

```
#include <stdio.h>

int main(void){
char c='5'; // ココは適当に書き換えてください，もしくはscanfを使う
int v;
  // scanf("%c", &c); // scanfで文字を指定してもよいかも
  if(c=='X'){ // X は強制的に10
    v = 10;
  }else{
    v = c-'0';
  }
  printf("%d¥n", v);
}
```

本日は以上