仕事の進め方を学ぶプログラミング教育

荒川 淳三

今日文科系大学において広く行われているプログラミング教育は，その最大の目的を計画に基づく仕事の進め方の体得におくとき，日本の文科系大学に大きく不足している動的知識教育を強化する意味において，またプログラミング習得水準を飛躍的に高める点において，絶大な効果を発揮する筈である．本論文はこの可能性を求めて，筆者が1993年度から札幌大学経営学部において展開してきたプログラミング教育の成果報告である．

未解決の問題が多く残されているものの，PADエディタの活用方法を工夫することなどによって，一つの教育方式の確立に近づきつつあると信ずるものである．

1．はじめに

コンピュータとネットワークによって代表される情報処理技術が社会を支える下部構造として定着した今日の大学では，文科系の学部においてもプログラミング教育が広く行われている．しかし何のためにプログラミング教育を行うのかということになると，大多数の賛同を得られるような考え方は確立されておらず，実際にプログラミング言語教育を担当している教員の中からも疑問が投げかけられているのが実情である．

この問題に真っ向から答えるものではないが「文化系学部におけるプログラミング言語教育は，これを仕事の進め方を学ぶ場として活用するとき素晴らしい成果を期待できる」というのが筆者の考えである(1)．

理由の第1は仕事の進め方の基本，すなわち周到な計画に基づく仕事の進め方を体得しておくことは，学生たちが社会に出たとき，何にも増して重要と思われるからである．

第2は仕事の進め方，すなわち動的知識の教育が不十分であることが指摘されている日本の大学において，文科系の学部においてはそれが特に甚だしいと思われるからである．理科系の学部では実験科目が動的知識習得の場としての役割を担っているが，文科系学部にはこれがない．

第3は難解な課題に繰り返し取り組むプログラミング教育は，PDCAの体得に好都合だからである．一つのプログラミング課題に周到な計画のもとに取り組んで完成させ，その結果を計画との対比において反省し，次の課題にこの反省を活かした周到な計画に基づいて取り組む．これはまさにPDCAに他ならない．

第4は周到な計画に基づいて仕事（課題）を行うという正しい進め方により，プログラミング学習自体が大きな成果を挙げることが期待できるからである．

本文は以上の考えのもとにプログラミング教育を実施した結果の分析と，今後の方向づけである．

Junzo Arakawa 札幌大学経営学部

2．仕事の進め方について

筆者の提案がプログラミング教育を通じて周到な計画に基づく仕事の進め方を学ばせることにあるので，周到な計画に基づくということが仕事の進め方においてどんな重みを持つのか，あるいはこれを身につけたらどの程度役に立つのかについて，最初に確認しておくことが必要であろう．

仕事の推進において基本的なことは，内容が詳細に分かっている仕事は（自分にできない部分は出来る人に依頼することも含めて）一般に実行可能であり，内容の分かっていないことは実行できないということである．

また仕事を処理するに当たって，特に知的な仕事においては，個人間の能力差は極めて大きく，これは仕事の実行段階での差とは考えにくく，能力差の大半は計画（設計）の質に起因すると判断される(2)．

複雑な知的業務の典型としては，システム開発を挙げることができるが，ここにおいても周到緻密な計画を立案し，業務の見通しを立ててから実行に入ることが鉄則である．具体的には開発に着手してから完了するまでの過程をいくつかのフェーズに分割し，各々のフェーズをどのような技法やツールを用いて展開するかを詳細に定めたプロジェクトライフサイクルを設計し，以降の作業はこの設計(計画)に従って推進する．開発プロジェクトの成否は設計されたライフサイクルの精度にかかっている．

仕事の計画において特徴的なのは，的確な分析が必要であるということである．仕事の内容をよく理解し計画を立てるには，仕事を十分に小さな単位業務（タスク）に分割し，個々のタスクについて理解することが必要であることは，デカルトの方法論においても主張されている通りである．

仕事の結果が計画との対比において評価され，問題点が次の計画の立案時に修正されれば，計画の質は着実に向上する．評価作業は極めて重要であり，これを確実に行うためには，計画自体が実績との客観的定量的対比を可能にするものでなければならない．

以上より周到な計画を立てたのちこの計画に従って仕事を行うということは，仕事の進め方の基本であり，これを身につけることで業務推進能力を飛躍的に高めることが期待できること，またそこにおいては「分析」という作業が大きな比重を占めることがわかる．

勿論，仕事の進め方は他の方法論同様，身につけている静的知識を活用するための知識(動的知識)であり，基盤となる静的知識を身につけて始めて，動的知識が活きてくる．周到な計画に基づく仕事の進め方はこの意味において学生たちが身につけるべき必要条件であって，十分条件ではない．

3．動的知識体得の場としてのプログラミング教育の展開

仕事の進め方を体得させるためのプログラミング教育は，次のようなものになろう．

(1) 講義のほかに，学生たちには次々にプログラミング課題が与えられる．

(2) 課題は徐々に高度化し，途中からは的確な計画がなければ簡単には仕上げられないレベルに達する．

(3) 学生たちは与えられた課題に対して，まず詳細な計画書を作る．計画書作成段階においては，作業の分析が十分に行われる．

(4) 完成した計画書はチェックされ，不都合な点，不十分な点が指摘される．

(5) コーディング以降の作業は，指摘に基づいて修正された計画書をもとに行われ，課題完了時に十分な反省（良かった点，改善すべき点）がなされ，ソースプログラムとラン結果を含む全成果物が実績報告書として提出され評価される．

以下に，筆者が勤務する札幌大学経営学部のプログラミング論Ⅰにおいて試みている教育のこれまでの結果を要約したい．この科目

は2年次の選択科目であり，履修者数は全学生の3/4前後である．履修者のほとんどは1年次の必修科目である「情報処理概論」の履修を終えている．プログラミング言語としてはフォートランを用いている．

3.1 計画

動的知識教育の場としてのプログラミング教育を始めるに当たって，1993年度はじめに立てた計画は以下の通りである．

(1) 計画書の書き方

B4版の計画書専用用紙を印刷し，これに手書きで次の項目を記述させることにした．手書きとしたのは，端末で処理の流れを記述するための適切なツールが準備されていなかったことによる．

計画書における記述項目の設定は，学生たちの分析作業を適切に誘導することを意識して行った．

(A) 機能概要

作成すべきプログラムの備えるべき機能を簡潔明瞭に記述する．

(B) 入力項目と入力形式

プログラムが外部から読み込むデータを列挙し，そのレイアウトを記述する．

(C) 出力項目と出力形式

プログラムが外部に出力する項目とレイアウトを列挙する．

(D) データ定義

プログラムで宣言する全てのデータについて，データ名と型式とを列挙する．

(E) テスト入力と出力

プログラムが計画どおりの機能を発揮することを確認するための入力とそのときの出力とを，フォーマットを含めて記述する．

(F) スケジュール

課題の与えられた日，計画書の提出期限，実績報告書（ソースプログラムとランの結果を含む）の提出期限に合わせて計画書作成，コーディング，テスト，実績報告書作成のスケジュールを計画し記述する．

(G) 処理の流れ

プログラムの処理の流れをフローチャートで記述する．

(H) 反省

課題の終了後記述する．

(2) 実績報告書の書き方

返却された計画書を修正したのちここに記述される主な事項は，スケジュールの実績と反省とである．

反省として記述されるべきは計画どおり行えた事項，計画どおり行えなかった事項，取り組み中に発生したバグその他の問題点，次回以降改善すべき事項などである．

(3) 予想される問題点と対策

この教育を展開するに当たって予想された問題点は次の通りである．

(i) コーディングに先立って詳細な計画書を作ることがどこまで徹底できるか

(ii) 計画書との対比による実績の評価がどこまで徹底できるか

(iii) 担当教員の負荷が過大になることを避けられるか

(iv) 教育成果の測定をどのように行うか

(i), (ii)についての難しさは，プログラミングの課題が宿題として与えられるため，課題の途中において担当教員の目が届きにくいことである．

各課題については，これを付与した1週間後に計画書を提出させ，その1週間後にチェックした計画書を返却し，さらに1週間後に学生は最終結果（実績報告書，ソースプログラム等）を提出するという工程を基本としたが，(i), (ii)についてその趣旨を繰り返し説明し，あとは学生たちを信用するしかないと考えた．

担当教員の負荷軽減策としては，①授業中に行う小テストについては答案用紙をその場で回収再配付し学生たちに採点させる，②月～金の16時20分から20時まで端末室で自学自習を行う学生たちを支援する補助指導員

に，暇を見計らって計画書のチェックをさせる，③最終報告書は担当教員がチェックすることで計画した．

成果測定は非常に難しい問題であるが，当面は（受講者には極秘で）年度末に毎年同一の実技試験を行い，年度ごとのプログラミング能力の向上度を比較することとした．

3.2 実績

1993年度から上記計画に従ってプログラミング教育を展開してきたが，これまでの実績を要約すれば以下の通りである．

(1) コーディングに先立つ計画書作成

学生たちは指定された期限までに計画書を作成して提出した．この点においては計画は成功であり，それなりの成果を収めたと考えられるが，問題点もいくつかあった．

最大の問題点は，フローチャート形式で手書きする処理の流れである．少し難しい処理では間違いが多く，更に複雑な処理では，明らかに他の学生のフローチャートを真似たと思われるものが大半となる．学生たちの自分でとことん考える姿勢と考えるための素地は不十分で，フローチャートを手書きする煩わしさがこれに拍車をかけたようである．

この問題を解決するためには流れ図レベルで思考する訓練を強化することと，処理フローの作成を支援するエディタを導入することが有効であると考えられる．

エディタについては1993年度の途中から，ふだんの学習に使用している端末に搭載された一太郎の罫線機能を利用してフローチャートを書くことにし，これによって計画書全体を同一のツールで作れるようになった．

しかしワープロの罫線機能を用いてフローチャートを書くことは非常に面倒で，学生たちの苦情も多かったため，また構造化プログラミングの体得も重要と考え，1994年度のはじめにPADエディタ（TOPCORN，日立中部ソフトウェア㈱）を導入し，計画書はワープロとPADエディタで書くことになった．

図1はフォートランプログラミングのためのPADの一例である．

なお，導入されたPADエディタには，フォートラン言語に対してのソースプログラム自動生成機能はないが，C言語については所定の規則に従って書かれたPADとソースプログラムとの間の双方向変換機能がある．PADエディタの導入によって，処理フローの記述と推敲は容易になったが，学生たちがPADを用いて処理の流れを考える姿勢はまだまだ不十分で，完成したソースプログラムからPADを作成している学生が少なくないのが実情である．

処理フローの他にも，テスト入力と出力，

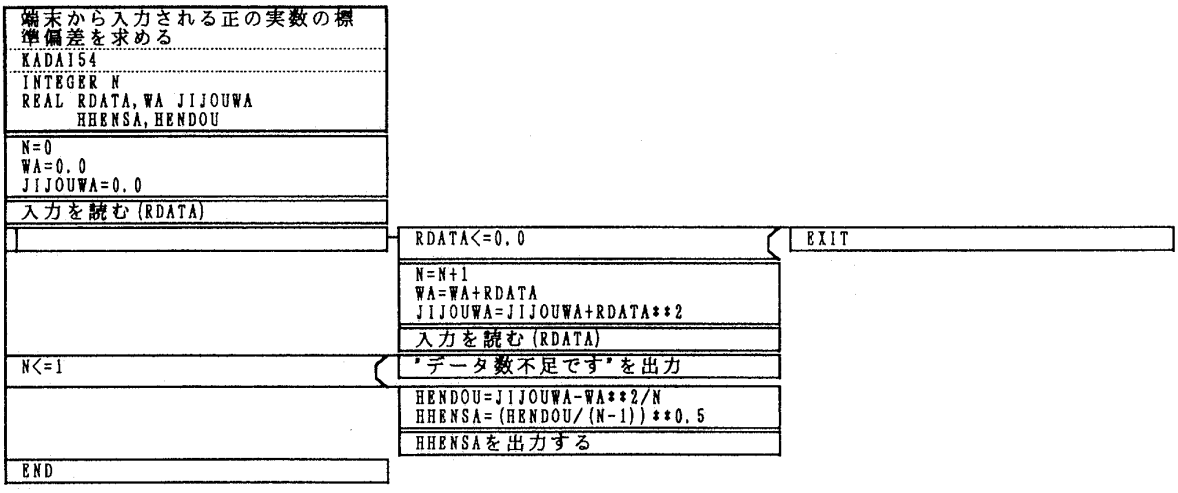

図1 フォートラン言語教育で用いたPADの例

スケジュールなどの記述に不備が目立つ.

テスト入出力については，1年間の教育ではプログラムの品質を保証するためにどのような考え方でテストケースを設定したらよいかまでカバーするのは困難である.

スケジュールについても1年間の教育では扱う課題が比較的単純なものに止まらざるを得ず，課題がいつ与えられ，いつ計画書を提出し，これがいつ返却されいつが最終提出期限か以上にスケジュールを分析し細分化する必然性がないのが実情である.

(2) 実績報告書の作成

実績報告書に関する問題点の1つは，課題に取り組む過程で生じたバグの把握が不完全で，結果的にある課題での経験が次の課題に十分に活かされないままに終わっていることである.

もう1つの問題点は，担当教員の負荷の都合で，実績報告書のチェック結果を提出者に返却できていないことである．ここの報告書に一応の目を通し，結果を集約して学生たちに説明し注意を促す現状では，この仕組みが十分に活かされているとは言えない.

(3) 担当教員の負荷

小テストの結果を学生たちに採点させる試みは問題なく実行され，不出来な自分の答案を友人に見られることのないよう学生たちが発奮するという，付随的効果も見られる.

計画書を補助指導員にチェックさせることも実行され，非常に有効である.

しかし前述のように，最終報告書をチェックし注意点を記入する負荷は非常に大きく，現在のところこれを行えてはいない．この対策としてはチェックすべき実績報告書を簡略化すること，最終チェックも補助指導員に手伝わせること，プログラミング教育のクラス編成（1993年度，1994年度とも100名以上）を小さくすることが考えられる.

(4) 成果測定

教育成果の測定方法としては仕事の進め方習得度の測定とプログラミング能力向上度の測定の2つが少なくとも必要であるが，現在のところ年度末に毎年ほぼ同一の問題で実技試験を行い，年度ごとの総合的成果を比較するに止まっている.

表1 年度末実技試験の成績比較

正解数	1992年度	1993年度
3/3	5人(8.3%)	37人(37.8%)
2/3	21人(35.0%)	24人(24.5%)
1/3	13人(21.7%)	7人(7.1%)
0/3	1人(1.7%)	2人(2.0%)
無資格	20人(33.3%)	28人(28.6%)
合計	60人(100.0%)	98人(100.0%)

表1は1992年度，1993年度の年度末に行った実技試験の成績の比較である.

試験ではフォートラン言語によるプログラミング課題3問を与え，2時間の試験時間内に何問正解できたか，すなわち正しいソースプログラムとラン結果のリストを提出できたかを調べた．不正行為を防ぐために4種類の問題を準備したが，両年度で用いた問題は，平均的には同じである．表において無資格とは最後までついてこられず，年度の途中でプログラミング課題の提出を諦めたため，最終試験の受験資格を失った学生たちである.

表から明らかなように，新しい教育方法を採用した1993年度の成績は前年度よりはるかによいが，これは必ずしも新しい方法の成果とは言いがたい.

1992年度においては，教える側に生じた突発的理由で，このクラスを担当する教員が2度にわたって変わるという事態を招いた．表の示すところは，むしろ，教え方によって如何に教育成果に大きな違いが生じるかということであろう.

仕事の進め方の習得度測定については現在のところ成案はないが，後述のように，毎年何回か与えられる標準課題（測定用課題）に対してコンパイル1回だけで正解を得る比率の測定が，実行可能であり有用ではないかと

考えている．

仕事の基本が，実行に先立って如何に周到緻密な計画を立てるかの習慣化にあると考えれば，PADまでの段階で完璧を期し，ここから自動生成されたソースプログラムの1回だけのコンパイルで課題を完成させ得る比率は，仕事の進め方の習得度の良い指標となるからである．

4．今後の進め方

仕事の進め方（動的知識）習得の場としてのプログラミング教育のこれまでの試行結果は以上の通りであり，1995年度以降に以下の課題を持ち越している．

- 学生たち一人一人がPADレベルで深く考える状態を実現する．
- プログラムの品質保証のためにはどんなテストケースを設定すべきかを教える．
- プログラミングの過程で生じるバグをより正確に把握させる．
- この教育に必要な負荷の軽減策をさらに進める．
- 仕事の進め方の習得度測定を実現する．

幸い札幌大学経営学部では，プログラミング論Ⅰ，プログラミング論Ⅱの内容を1995年度から一部改めることにしている．

(1) プログラミング言語は，フォートランからC言語に変わる．結果的にPADのソースプログラム自動生成機能を使うことが出来る．

(2) これまでプログラミング論Ⅰではフォートラン言語の基礎的内容を，プログラミング論Ⅱでは応用的内容を扱うことにしていたのに代え，次年度からはプログラミング論Ⅰ，プログラミング論Ⅱを通してC言語の基礎を教える．すなわち同じC言語について同じ姿勢で，2年間をかけた息の長い教育が可能になる．

(3) クラス編成を50名以下に縮小する．

この変更を機に，新しい年度からは前述の懸案問題を，次の方策に策を集約して解決したいと考えている．

(1) PAD 1枚に計画書･実績報告書の全てを集約し，簡略化を図る．これにより計画書，実績報告書の収集返却，チェック作業の負荷を軽減する．クラスの縮小でさらなる負

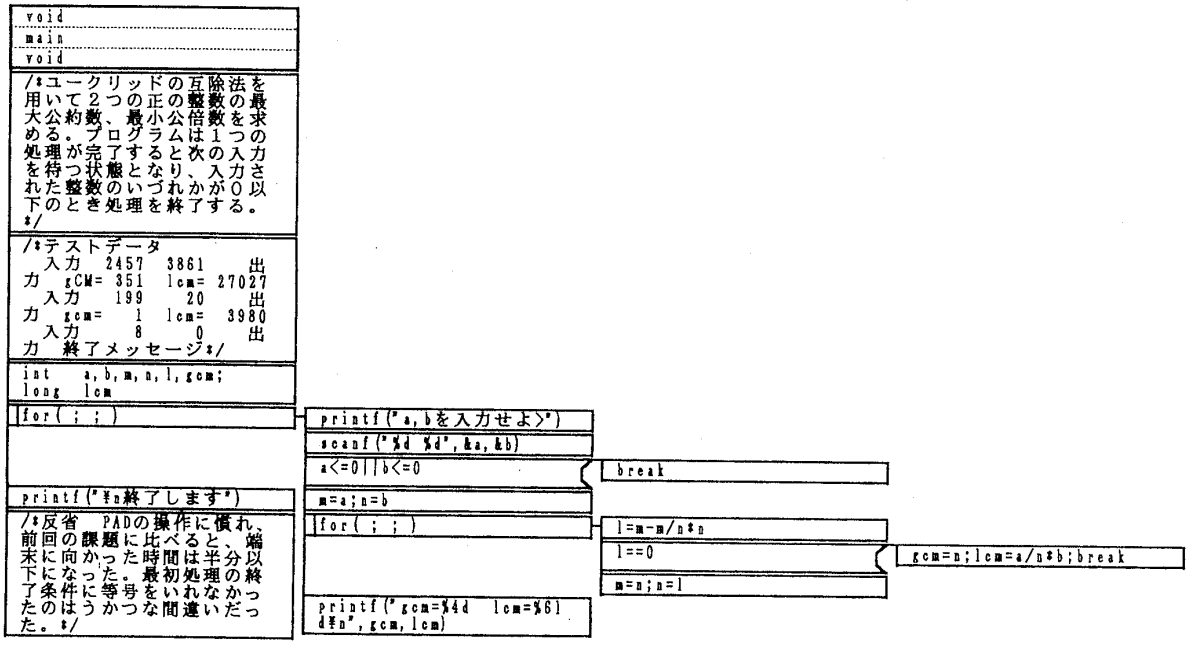

図2　機能概要，テストデータ，反省などを含むPADの例

荷の軽減が可能になるため，実績報告書を提出者に返却することも可能になる．図2は機能概要，テストデータ，課題の反省などを書き込んだPADの例である．

(2) 最初の段階で時間を十分にかけ，さまざまな論理をPADで表現する訓練を十分に行う．PADをプログラミング的記号で記述すると，これだけで学生たちの思考が幻惑されがちなので，この段階のPADは日常的言語で記述させる計画である．

(3) 2年間かけてテストケースの設定法，スケジュール管理，バグの正確な把握法などをじっくり訓練するカリキュラムを作り上げる．

(4) ソースプログラムはPADから自動生成することとし，「自動生成1回」で最終結果を得ること，すなわちPAD以前の入念な取り組みで全てのバグを除去することを精

```
#include<stdio.h>
void main( void )
{
/*ユークリッドの互除法を用いて２つの正の整数の最大公約数、最小公倍数を求める。
  プログラムは１つの処理が完了すると次の入力を待つ状態となり、入力された整数の
  いづれかが０以下のとき処理を終了する。*/
/*テストデータ
  入力  2457  3861    出力  gCM= 351  lcm= 27027
  入力   199    20    出力  gcm=   1  lcm=  3980
  入力     8     0    出力  終了メッセージ*/
    int   a,b,m,n,l,gcm;
    long  lcm;
    for( ; ; ) {
        printf("a,bを入力せよ>");
        scanf("%d %d",&a,&b);
        if( a<=0||b<=0 ) {
            break;
        }
        m=a;n=b;
        for( ; ; ) {
            l=m-m/n*n;
            if( l==0 ) {
                gcm=n;lcm=a/n*b;break;
            }
            m=n;n=l;
        }
        printf("gcm=%4d   lcm=%6ld¥n",gcm,lcm);
    }
    printf("¥n終了します");
/*反省  PADの操作に慣れ、前回の課題に比べると、端末に向かった時間は半分以下に
  なった。最初処理の終了条件に等号をいれなかったのはうかつな間違いだった。*/
}

C>
C>
C>euclid
a,bを入力せよ>2457 3861
gcm= 351  lcm= 27027
a,bを入力せよ>199 20
gcm=   1  lcm=  3980
a,bを入力せよ>8 0

終了します
C>
```

図3 PADから自動生成されたC言語ソースプログラム

力的に推進する．図3は図2のPADから自動生成されたソースプログラムとランの結果とである．

なお1995年度からはプログラミング論を履修する母集団がかなり変わると考えられる．具体的にはプログラミング論Ⅰ，Ⅱが同じ科目名のもとにC言語を学ぶクラスと表計算・データベース等の応用パッケージを学ぶクラスに分かれるので，(目標を定めた学生が集まるという理由で) 質的向上が期待されるためである．このため1994年度以前との比較が難しくなるが，以上述べた考え方で着実に実績を積み上げ，動的知識を学ぶ場としてのプログラミング教育を納得の行く水準にまで確立したいと考える．

謝辞 1995年1月11日に札幌学院大学で開催されたアルゴリズム教育ワークショップでは熱のこもった討議が展開され，前回に引き続き啓発されるところ多大でした．ワークショップを開催された札幌学院大学社会情報学部の皆様方に心からお礼申し上げます．

文献

(1) 荒川淳三：問題解決過程としてのプログラミング，社会情報，札幌学院大学社会情報学部紀要，Vol.2，No.2，pp.75－80，(1993)．
(2) 小林忠嗣：知的生産性向上システムDIPS，p.33，ダイヤモンド社，(1992)．