国際研究論叢　19（3）：41～57，2006

社会科学系学部におけるアルゴリズム学習教材の開発

中　井　哲　夫[*1]　岡　本　容　典[*2]　下　條　善　史[*3]

Development of Algorithm Learning Materials for Social Science Departments

Tetsuo Nakai[*1]　Yosuke Okamoto[*2]　Zeishi Shimojo[*3]

Abstract

In the department of Management and Information Science at Osaka International University, programming languages such as Basic or C are taught as part of the basics of information technology. Using conventional teaching methods, beginners became discouraged by the monotony of learning correct syntax. In contrast, audio-visual lessons, while motivating, almost always fail to develop students ability to grasp abstract points, such as grammar and algorithms.

Recently an increasing number of students cannot grasp the flow of programs by just being shown a printed Flowchart. To improve this situation we propose a solution to aid visualization of programs and improve motivation called Dynamic Flow Chart（DFC）where students can easily understand the algorithms and flow of programs.

キーワード

アルゴリズム学習教材、フローチャート、Flash、Java、GUI、プログラミング教育

Keywords

algorithm learning material, flowchart, Flash, Java, GUI, programming learning

はじめに

1-1　社会科学系学部でのプログラミング教育

本学経営情報学部では、開学以来、情報基礎教育の一環としてBASICやC言語などのプログラミング教育を続けてきた。

＊1なかい　てつお：大阪国際大学経営情報学部教授
＊2おかもと　ようすけ：大阪国際大学経営情報学部助教授
＊3しもじょう　ぜんし：大阪国際大学法政経学部助教授〈2006.1.12受理〉

従来より、プログラム言語の文法を忠実に教える授業（たとえば、キャラクタベースのコンソールアプリケーションを実習例題とするもの）は、単調な操作の繰り返しや地味な実行結果等によって、初学者の学習意欲を激減させてしまう傾向が強かった。逆に、グラフィクスやサウンドなどを取り入れたvisualなアプリケーションをサンプル例とする授業では、学習意欲を湧き立たせる点では絶大なる効果を持つが、入門者にブラックボックスとして無条件に受け入れさせる部分が多すぎて、文法やアルゴリズムといった抽象的な思考力を多くの学生に上達させるまでにはなかなか至っていないのが実情である。

経営情報学部では、ITに明るい社会人を育成するという目標があり、特にITコースを選択する学生にとっては、プログラム言語の習得は必須の課程であると言っていい。しかし一方、ITコースを選択する学生と言えども、情報工学などを専攻する学生とは違っていわゆる文科系に属する学生であることに変わりはなく､プログラミングにおいて重要な、数理的思考能力における訓練がどうしても不足するという現状がある。

特に近年、プログラミング言語の講義などで、フローチャートを紙上で示しただけではその処理の流れを頭の中でイメージできない学生が増加している。分岐や繰り返しを含まない単純なプログラムなら理解できても、配列変数やループを組み合わせた処理（特に二重ループ処理など）に対して、その流れをイメージできず、そのため最大最小やソートなどの基本的アルゴリズムの学習に支障をきたしている学生も多い。

その原因としては、どの大学でも近年問題になっている、大学生の学力低下という現象も影響していると思われる。特に本学経営情報学部の志願者においては、情報関連の技術を学習する学部でありながら、試験科目にはいわゆる理数系の科目は存在せず、日本語・英語などの語学科目の試験だけで選考している現状であり、数理的、論理的思考能力を確認する場がまったく設けられていないことも大きな要因と考えられる。

このような状況で、数理的、論理的思考能力の訓練を進めるということは、本学部においてのみならず、ほとんどの社会科学系学部においても重要な課題であると思われるのだが、プログラミングの学習においてこれらの訓練を行う際問題となるのは、やはり、テキストベースでのプログラム開発の過程は基本的に最初から抽象能力を必要とする作業であり、プログラミングの過程とその最終成果の双方が、学生にとって興味をひくものであり難いということである。そして一方、最終成果で学習者の興味を引くためにvisualな課題を教材とすると、プログラムの全体は複雑になりすぎ、初学者の学習においては、プログラムの大半をサブルーチンなどに頼った定型的なコードで埋めざるを得なくなってしまう。このように、数理的、論理的能力の訓練と、学習意欲を高める課題を両立することの困難さが、社会科学系学部でのプログラミング学習にはついてまわっているのが現状なのである。

このような現状を改善する方法として筆者らは、プログラミングをvisualizeすることによる解決を試みた。これはプログラミングによって作成されるプログラムの内容をvisualな課題にすることではなく、たとえその課題が数理的なアルゴリズムの訓練だけをめざしたものだとしても、プログラミング作業の過程自体をvisualにすることによって、アルゴリズムやプログラムの動作を視覚的に理解しやすくし、学習者の興味も引いて、学習意欲

を高めるものにしてしまうというアイディアである。このようなプログラミング学習環境をここでは DFC（Dynamic Flow Chart）と呼ぶことにする。以下ではDFCの詳細を述べることにする。

1-2　従来のプログラミング学習過程

従来、情報工学部などで行われてきた、プログラミング言語を習得するために設けられてきた学習の課程は一般的に以下のようなプロセスをたどることが多い。すなわち、まず習得する言語を一種類に定め、最初にその言語の基本的な文法のあらましを学習する。次にアルゴリズムの組み立て方を学習して、簡単な課題を設けてプログラミングを体験する。その後、基本的な入出力、制御構造、ファイルの入出力などを学習して、徐々に複雑なアルゴリズムを経験していき、関数、配列変数、サブルーチンなどを学習していくといった順でプログラミングを習得していく。

これはいわゆる理科系の学生に対してプログラミング言語を習得させる際の手順であるが、このような学習過程を社会科学系の学生にたどらせる際問題になるのは、まずプログラミングという概念自体を理解していない学生が多く、仕事を一連の論理的操作に分解するといったアルゴリズムの考え方になじんでいないのが一般的であるという部分である。たとえば数学の問題を解くためには、最初の過程から論理的ステップを重ねていって、最終的な解答にたどり着くというプロセスを踏む経験が重要であるが、社会科学系の学生にとっては数学は受験科目ではなく、このような作業に対する慣れと経験が十分でないことも多い。

情報工学部のような、理数系の学習経験を十分に重ねてきている学生に対するプログラミングの学習であれば、従来の方法で問題のないこともあるのだろうが、これをそのまま社会科学系の学生に適用してはいけないというのが、我々が経営情報学部でプログラミング教育を行ってきた際の経験則である。

社会科学系の学生に対しては、まずはアルゴリズムやプログラミングという概念を肌で感じさせ、理解させることが必要である。そのために、プログラムの課題を十分に興味の持てるものにする必要があり、しかしだからといって理解できないほど複雑なプログラムであってはならない。論理的思考に慣れていない学生が十分理解できる程度の単純なアルゴリズムからプログラミングという概念を感覚的に理解させ、そこから数理的、論理的思考の経験を積ませていく必要があると思われるのである。

2　DFC

2-1　DFCの概念

DFC はプログラミングを学習する初学者が、コーディングにおけるテキスト編集のわずらわしさや、課題のプログラムの最終的な成果の単調さから学習意欲をそがれることなく学習できることをめざして作られた、汎用的なプログラム学習のための環境である。この環境の中心はウィンドウ環境の上につくられた visual な入力画面である。学習者はこの画面上にアルゴリズムを表すフローチャートで使われる何種類かの部品を配置し、それぞ

れを線で結んでいくことによってアルゴリズムを組み立てていく。それぞれの部品にはアルゴリズム上の機能に応じたアトリビュート入力画面が用意され、学習者はそこにさまざまな式やパラメータを入力して、アルゴリズムに具体的な動作を割り当てていく。最終的にアルゴリズムを組み立て、アトリビュートを指定した時点で、プログラミングはひとまず終了する。次の画面はこのプログラムを実際に動作させることのできるインタプリタとデバッガである。画面上には現在制御の行われているフローチャートの当該ブロックが強調して示されるようになっており、またその時点での各変数の内容がリアルタイムで表示されるようになっている。プログラム作成者はインタプリタによってプログラムを実行させると、DFC はあらかじめ設定された速度で、プログラムの各ステップを次々と実行し、その指示通りに制御を移動させ、また変数の内容を更新していく。実行画面ではそれらが同時にそれぞれのウインドウ上で表示され、変化していく。

もしもプログラム作成者の意図とは異なる動きをした場合、それは変数の値、またはフローチャート上の制御の進み方で判別できる。プログラム作成者はフローチャートを追跡しながら問題のある部分を推測し、先刻のプログラム作成画面上でそれを修正する。そして再びインタプリタによって実行する。これを繰り返して正しい動作をするプログラムが作成されることになる。

このような、フローチャートの入力によってアルゴリズムを設計し、アトリビュートの入力によってプログラムを作成することができ、さらにそのプログラムをインタプリタによって実行して画面上でデバッグすることのできる環境を、Dynamic Flow Chartと呼んで今回作成した。

2-2　DFCによる学習の効果

この DFC はすでに紹介したようにプログラムの作成と実行、デバッグという作業自体が visual であるため、プログラミング作成の作業における各段階を初心者が確認し理解しながら進めていくことができるようになっている。社会科学系学部の学生に対する既存の学習方法によるプログラミング学習の問題点は、(1)プログラムそのものと、作成したプログラムによる実行結果が、双方とも単調なテキストで表されるため、もともと関心のあった学生以外には興味を引きにくく、学習意欲が高まりにくい。(2)学習意欲を喚起するため、プログラムの実行結果が華美な visual 系言語を使用して初学者の学習教材にすると、プログラムの実行結果には興味を引くものの、プログラムコード自体が複雑になりすぎ、初学者が簡単に記述することには無理があるため、詳細に内容を理解せずにただ書き写すだけのコードが大半を占めることになる。そのため、プログラムの動作を十分に理解しないまま学習することになり、達成度も満足度も高くできない。というものであるのは前章でも述べた。DFC によるプログラミング学習は、課題そのものは初学者の理解にあわせた、初歩のアルゴリズム問題のような、比較的少ないコードで記述できるもので構わない。したがって(2)の問題は発生しない。またプログラムの作成も、実行とデバッグの手順もテキストベースではなくvisual ベースで行われるので、プログラミングの作業自体が興味を引き、しかもプログラムの動作を文字通り目の当たりにすることができるので、プログ

ラムの全体が理解しやすく、また間違いの発見と修正もしやすくなると考えられる。この仕組みにより、数理的、論理的思考によるアルゴリズムの構築と、それを具体的にコードに変換していく作業がシームレスに行えることになり、プログラミングの学習の敷居を高くしている要因のひとつである、アルゴリズムからコードへの変換というステップが、DFC ではまったく必要ないことになる。このことも、初学者の学習負担を軽減している。このようなことから、DFC を用いたプログラミング学習は、初学者の経験するさまざまな困難を軽減し、より容易な入門のプロセスを提供する学習教材として有効であると期待できるのである。

しかし DFC も完全でないことは言うまでもない。DFC に期待できるのは、アルゴリズムとプログラミングの関係を感覚的につかむと言うところまでである。一般的なプログラミング言語は、コマンドや関数などを、正確に文法にのっとった形式で記述してやらなければ、正しい動作が期待できない。フローチャートによるアルゴリズムの記述は、プログラミングの観念を理解させるところまでは有効でも、実際の言語によるプログラミングを習得するためには、やはり従来の課程での学習と経験が必要であると言わざるを得ない。プログラミング学習において DFC という環境が担当するのは、数理的、論理的思考、すなわちアルゴリズミックな考え方への入門を果たすまでのところである。アルゴリズムの構築について十分に学習した後は、実際の言語でプログラミングの経験を重ねることが、プログラミング学習には必要である。

2-3　開発言語とプラットフォーム

DFC の構想は2003年に始まっており、現在では２つのバージョンが存在している。ひとつは2004年９月に開発された DFC（Java版）であり、もうひとつは2005年９月に開発された DFC（Flash版）である。それぞれの特長と仕様はこの後の章で詳説するが、いずれも上記の構想に基づいて開発されたものであり、初学者のプログラミング学習に十分使用しうるものとなっている。

開発に使用した言語は Java ならびに Flashと、どちらもネットワークに親和性の高いオブジェクト指向型の言語であり、Webなどの仕組みを使用したネットワークアプリケーションを、visual なインタフェースで構築しやすい言語仕様になっている。また双方ともプラットフォームを選ばない言語でもあり、現にDFCの開発行程においても、異なるプラットフォーム間での分割開発と、分散デバッグをまったく支障なく行うことができた。そしてそのことはDFCというプログラム学習環境にも、プラットフォームを選ばず、また Webによる学習環境の提供に対しても容易に実現することができるという効果をもたらすことになった。

現在のところ、上記の二つのバージョンとも、ネットワークを有効利用した学習環境の提供などについて、具体的に実現されたものはない。しかし、今後はネットワーク経由で提供され、家庭など大学の教室以外での学習を可能にした、プログラミング学習環境としてのシステム構築を念頭においてバージョンを重ねていくことになると考えている。次章ではこれら二つのバージョンのDFCについて仕様と特長の詳細を述べていくことにする。

3　Java版DFC

3-1　Java版システムの概要

2004年にJavaで開発したアルゴリズム学習教材ソフト、「DFC」の概要は以下の通りである。

画面構成

本ソフトの実行時の画面の一例を図3-1に示す。

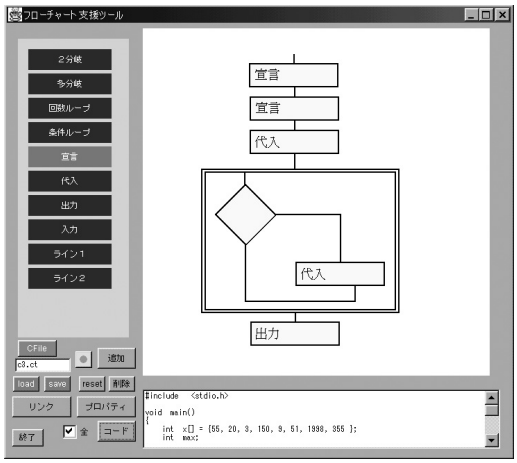

図3-1　実行画面例

図3-2に示すように、本ソフトの起動時の画面は大きく4つの部分で構成される。

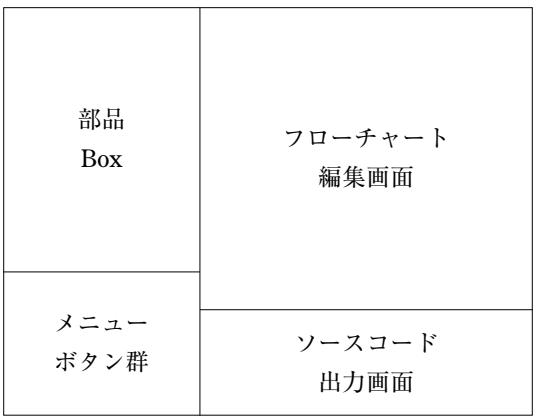

図3-2　画面構成

「部品Box」は、フローチャートを構成するユニットのリストで、本システムでは表3-1のようにC言語の構文と対応している。

表3-1　部品リスト

部品名	対応するC言語の構文
2分岐	if文
多分岐	switch文
回数ループ	forループ
条件ループ	whileループ
宣言	変数の宣言と初期化
代入	変数の演算
出力	printf（　）
入力	scanf（　）
ライン1	部品の連結に使用
ライン2	部品の連結に使用

「メニューボタン群」は、図3-3のようになっており、本ソフトの機能（メニュー）を実行させるボタン群である。

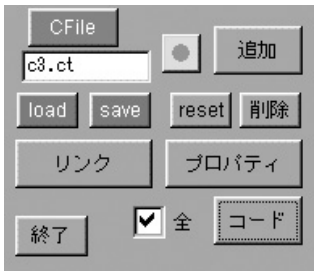

図3-3　メニューボタン群

操作手順

(1)「部品Box」からフローチャート(FC)の部品を選択し、画面中央の「FC 編集画面」に配置してゆく（**追加**ボタン)。

(2) 配置した部品を削除したり、起動時の状態に戻すことができる（**削除**ボタン, **reset**ボタン)。

(3) 配置した部品は簡易的に決められた既定値で初期化されているが、より詳細に設定することもできる（**プロパティ**ボタン)。

例えば、FC 編集画面上の「回数ループ」部品をクリックして選択し、**プロパティ** ボタンを押すと、図3-4のような「詳細設定プロパティ」ダイアログが表示される。

図3-4　プロパティダイアログ

(4) かなり複雑な構文（２重ループやループ文の中に分岐文が入った複合文など）に関しては、現バージョンではそれらの部品間の連結を詳細に設定することで対処している（リンクボタン）。

図3-5　リンク情報ダイアログ

例えば、ＦＣ編集画面上の複合文の内部にあるなにか部品を選択して、リンクボタンを押すと、図3-5のような「リンク情報」ダイアログが表示される。既定値を適切に変更することで、複雑な構文のフローチャートも作成可能となっている。

(5) ＦＣが完成すれば、実行ボタン（●）を押す。
図3-6のような「デバッグ」画面が別のウィンドウとして表示され、プログラムの流れがアニメーション的に且つ効果音付きで表示される。また、プログラムカウンタや各処理の変数値がリアルタイムに一覧表で示される。

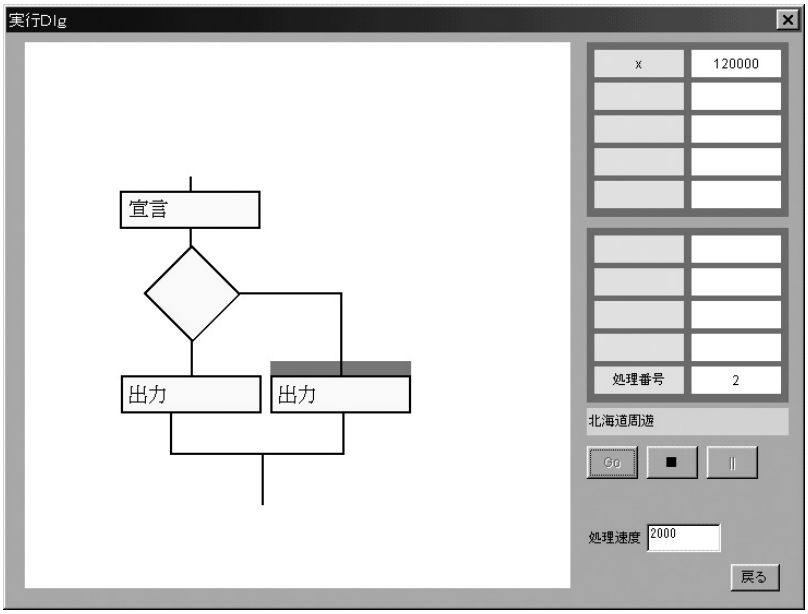

図３-６　デバッグ実行画面

(6) また、FC の各部品をクリックして コード ボタンを押すと、その FC に対応したＣ言語のソースコードが「ソースコード出力画面」に表示される。「**全**」のチェックボックスにチェックが付いている場合、全ソースコードが表示される。例えば、１から100までの整数の合計を求めるFC は、図３-７のようになるが、その全ソースコードが出力画面に図３-８のように表示される。尚、 CFile ボタンを押すと、全ソースコードがＣのソースファイルとして指定したファイル名で保存される。

(7) 最後に、 save ボタンを押すと現在表示されている FC がファイルに保存される。これによって後で修正したい場合、 save ボタンで保存した FC を編集画面上に再現することで可能となっている。

３-２　システムの特色

Java 版 DFC の特色は以下の通りである。

(1) visual なデバッグ機能

　従来のデバッグ機能（変数値の表示など）に加え、ループや分岐などの実行の流れがアニメーション的に効果音付きで示される。

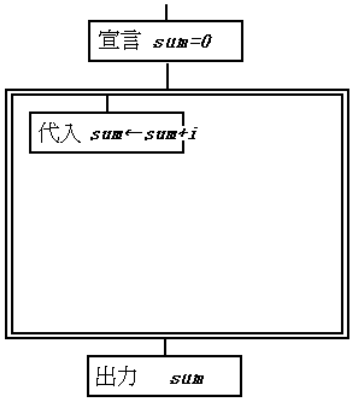

図3-7　合計プログラムのＦＣ

```
void  main()
{
    int  sum = 0;
    for(int i = 1; i <= 100; i++ )
        sum += i ;
    printf("sum = %d ¥n", sum );
```

図3-8　全ソースコードの出力

(2) プロパティダイアログ

各部品のプロパティの詳細な設定をするためのダイアログが用意されている。これによって、配列変数の初期化のような実用的なレベルの FC（例えば、ソートプログラム）の作成も可能となっている（図3-9参照)。

(3) 入出力の実装

出力及び入力の詳細設定ダイアログによって、例えば配列変数の表示などに対応しており、処理結果の出力などの具体的なプログラムの作成も可能となっている。

(4) リンク情報

本ソフトウェアでは、できるだけ構造化プログラミングの指針に従ってＦＣを作成するようにしている。従って、いわゆる GOTO 文を多用したスパゲッティプログラムとならないように設計している。例えば、ループ文は一つの独立した部品として扱うようになっており、分岐（IF文）とライン(GOTO文)でループを構成するようにはしていない。ただしこのような設計の場合、複合文などを部品を配置してラインで結ぶだけでは処理の流れを一意に自動で設定できない場合がある。そのような場合に対処するため、本ソフトウェアでは部品間のリンクを手動で再設定するための「リンク情報」ダイアログを用意している。これによって入り込んだ FC のリンク部分を確実に一意に決定することができるようになっている。

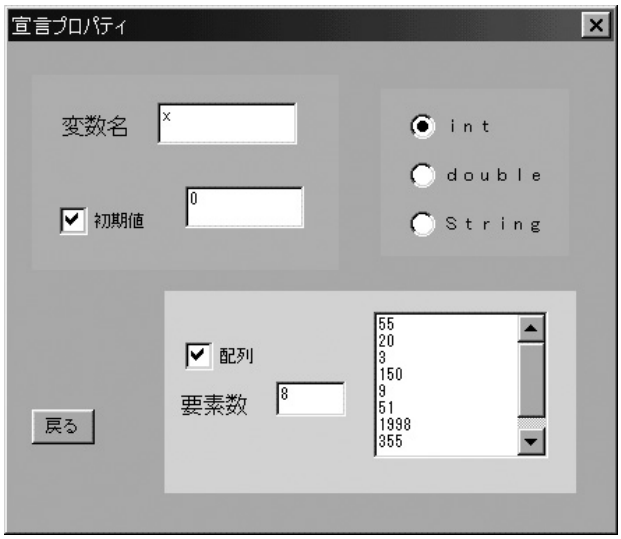

図３-９　配列変数の初期化の例

(5) ソースコード生成

FC 個々の部品ごとのソースコード及び FC 全体のC言語のソースコードが部品をクリックするだけで「出力画面」に表示される。これによって、FC の一部分がどのようなC言語のソースコードに対応するのかを学習することができる。

(6) FC の保存とロード機能

作成した FC は任意の段階でファイルに保存することができる。また、保存したファイルをロードして、保存時の画面を再現し、編集を再開することも可能である。

(7) Web 対応

本ソフトウェアは Java で作成されているため、容易に Web 上のアプレットへ変換することができる。ただし、ファイル関連の機能が若干制限される。

４　Flash版DFC

４-１　Flash

Flashとは、アニメーションを組み合わせて Web コンテンツを作成するための開発環境、及びそれによって作成されたコンテンツのことである。Macromedia 社により開発され、近年では Safari や Internet Explorer などの Web ブラウザで標準的に対応しているため、ユーザーは特別なソフトを追加しなくても、容易にそのファイルをブラウザ上で実行できる。Flash は、図形をベクター形式（頂点とそれを結ぶ曲線式のパラメータ）で管理しており、作成されたファイルのサイズが非常に小さいという特長をもつ。インターネットの回線が低速だった時代でも実用的な速度で Web 上のアニメーションを実行できたために、今日では広く普及している。

Flashでは基本的に、オブジェクトを時系列に配置してアニメーションを作成する。これらにインタラクティブな操作を機能として追加するために、オブジェクトに対する挙動

を ActionScript と呼ばれるプロトタイプベースのオブジェクト指向処理言語で記述する。例えば、画面上のボタンが押された時の処理や、アルゴリズムに基づくオブジェクトの処理などがこれによって実現できる。

4-2　Flash版DFCの特徴

本学でも導入されているMicrosoft社のVisual Studio .NET など代表的な従来のプログラム開発環境では、アルゴリズム作成とコーディング、プログラムの実行、そしてデバッグが独立のウインドウで行われてきた。このため、初学者にとっては、アルゴリズムとソースコード、ソースコードと実行時の処理の対応関係が理解しづらいものであった。デバッガを用いれば、処理の流れを段階的に追跡することは可能だが、ソースコード自体が静的に記述されているため、分岐や繰り返し処理をイメージしにくいという本質的な問題を抱えていた。

Flash 版 DFC は、それらの問題を解決するため、可能な限り統一的な環境で、かつ直感的に理解させることを目標として設計されている。ユーザーは DFC 上でフローチャートを作成し、その同一の画面上でアルゴリズムの処理の流れを追いながら、プログラムの動作原理を学ぶことができる。さらに実行中であっても、フローチャートの構造を動的に変更したり、その中で用いられている計算式や条件文を変更したりすることができ、その変更の影響を即座にフローチャート上で調べることも可能であるため、初学者にとって理解の助けとなることが期待できる。

4-3　Flash版DFCの操作

今回開発した Flash 版 DFC の起動時の画面を図 4-1 に示す。

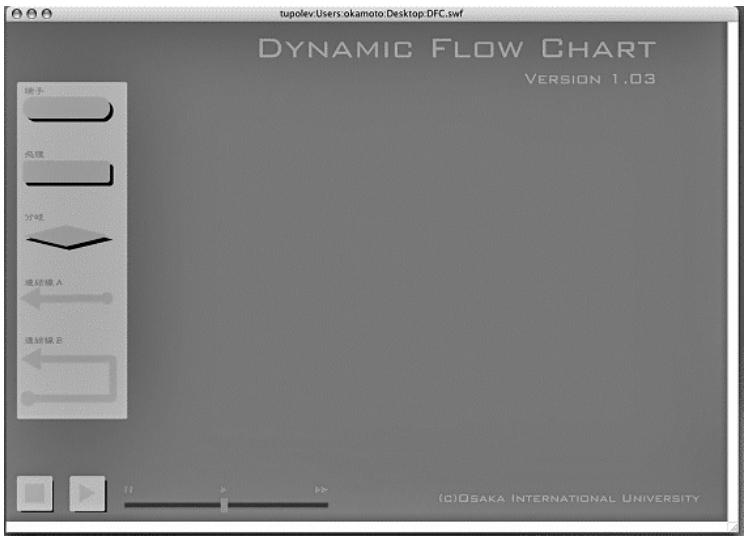

図 4-1　Flash版DFCの起動画面

画面左側にはフローチャート作成の際に用いる「部品ボタン」が配置され、それらをクリックすると、画面中央部にそれに対応した「部品」が現れる。部品は標準的なフローチャートで用いられるものに準拠した機能をもっている。個々の部品はドラッグすることで、自由に位置を変えることができ、また、一部の部品では数式も入力可能となっている。

まず、部品ボタンの種類と機能について以下に述べる。

(1) 端子

フローチャートの開始と終了を指定するために用いる。実行時、開始の端子からステータスバルーンが表示され、それが終了の端子へ移動したときにそのアルゴリズムの実行は終了する。

(2) 処理

様々な計算式を入力するために用いる。計算式中で使える変数や演算子は、C言語に準拠している。ステータスバルーンがこの部品の位置へ来たとき、そこで指定された演算が行われ、その結果はステータスバルーン内に瞬時に表示される。

(3) 分岐

条件式を入力し、それに応じて処理を分岐させるために用いる。ステータスバルーンがこの位置へ来たとき、条件式が真の場合は右へ、偽の場合は下へ分岐する。

(4) 連結線A・B

(1)から(3)までの部品を連結させるために用いる。連結線Aは垂直方向に伸長させることができる。また、連結線Bは垂直・水平方向に形状を変化させることができ、主として分岐によって生じたループを表現するために用いる。

次に、画面下部に配置されているボタンは次の機能をもつ。

(1) 停止

実行を停止するときに用いる。一時停止の状態でこのボタンを押すと、ステータスバルーンが消え、実行はキャンセルされる。

(2) 実行

実行を開始するときに用いる。このボタンを押すと、開始の端子にステータスバルーンが現れ、速度調節バーに応じた速度で、ステータスバルーンが移動していく。

(3) 速度調節バー

バーをスライドさせることにより、ステータスバルーンの移動速度を調節できる。最も左へスライドさせると一時停止となる。

ユーザーは、以上述べてきた部品をマウスでドラッグしながら、フローチャートを構成し、処理や分岐部品に対して計算式や条件式を入力する。フローチャートの完成後、「実行」ボタンをクリックすると、ステータスバルーンが表示されて実行が開始される。ステータスバルーンにはその時点の変数値が表示され、処理に応じて内容が更新されていく(図4-2)。実行中も、計算式や条件式は自由に変更が可能となっており、ステータスバルーンがその部品の位置へ来た際にその変更は即座に反映される。

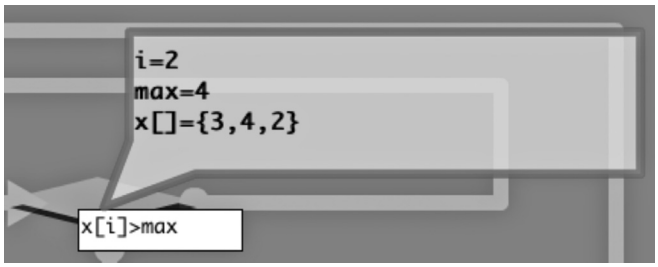

図4-2　ステータスバルーン

図4-3に、最大値を求めるアルゴリズムのフローチャートをDFC で表現した例を示す。

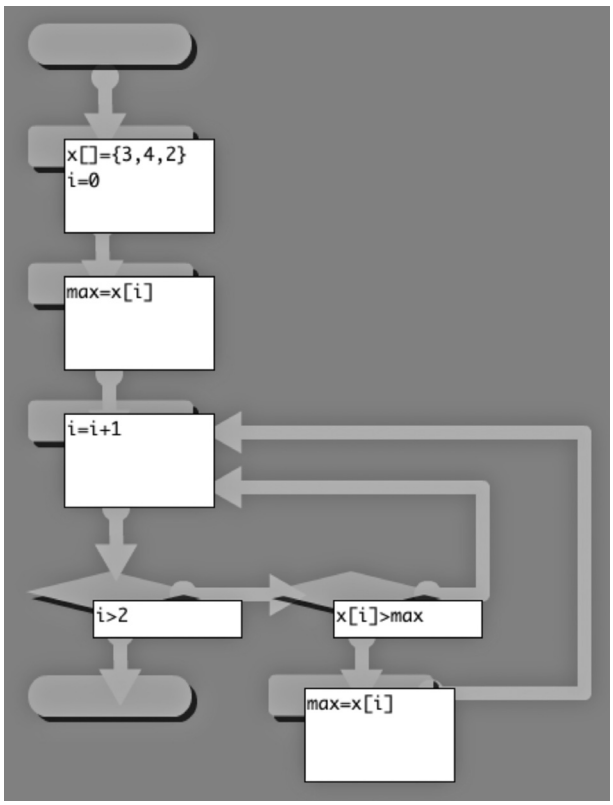

図4-3　最大値を求めるアルゴリズム

4-4　Flash版DFCの課題

最後に現在の Flash 版 DFC が抱える問題点を述べる。

まず、表現できるフローチャートは現在のウインドウの画面サイズ内に収まるものに限られるため、複雑なものを描くことができない。これは仮想画面方式を採用し、画面全体の拡大縮小を自由に行える機能を追加することにより解決できると考えている。ユーザー

は画面が縮小された状態でフローチャート全体の構成を考え、画面の一部が拡大された状態でその詳細を作成できるようにすれば、より複雑なフローチャートについても扱うことが可能となるだろう。

次に、ステータスバルーンで現在の変数値を即時的に知ることができるのはFlash版DFC の特長の一つであるが、ステータスバルーンのサイズも固定されているため、その中に含められる情報には限りがある。つまり変数の数が多くなったり、配列に多くの値を代入したりすると、その一部しか表示されないために視認性が低下してしまう。これについては、ステータスバルーンのサイズが表示すべき情報量に応じて自在に変化するようにすれば解決できる。

5　おわりに

5-1　既知の問題点

現在の Java 版、Flash 版、両実行プログラムはいずれもベータバージョン相当のものであり、学習環境としての実用のためのテスト、細かいバグの修正、マニュアル・ドキュメントの整備などについてはなお作業中である。各システムにおける個々の問題点については、それらを説明した各章に任せるとして、ここではもう少し大きな視点からの問題点を挙げることにする。

まず、このようなアルゴリズム教育環境の実際の授業の中での有効性についての検証が得られていない点を挙げる。先行して開発していた Java 版はすでに一部の授業でアルゴリズム教育に利用しているが、なお教育現場での利用を重ねて有効性についてのデータを収集しなくてはならない。Flash 版についても同様に教育の場での利用を始める必要がある。

次に、DFC という学習環境をネットワーク上の異なるプラットフォームの上で利用するためのシステムモデルの考案という課題もある。DFC が Java ならびに Flash という言語によって開発された所以である、遠隔自己学習のための機能が未整備であり、ネットワーク利用の学習モデルが明確に描かれていない。アルゴリズムの学習は、論理的・数理的思考の訓練でもあり、学習速度の個人差が激しいと考えられる教育領域である。そこで、自習環境によって必要なだけ繰り返して学習できるような機能と環境を整備し、その学習モデルを作成しておくことが必要である。

最後に、プログラミング言語の学習用教材として利用する以上、DFC を使用して学習した学生が、最終的になんらかのプログラミング言語を習得するまでの道筋を用意しなくてはならない。しかし現行の DFC の機能は、初歩のアルゴリズムを学習するところまでにとどまっており、テキストワークによるプログラミング言語の学習までの間にギャップが存在している。これを埋めるアイディアとシステムを考案しなくてはならない。

以上のような問題点を、順次解決に導いていかねばならない。

5-2　最終的な目標

DFC のそもそもの発想は、近年の本学部でのプログラミング学習において、さまざま

なポイントで学生達が学習意欲を減退させ、あるいはプログラミングを習得するということをあきらめていく、そんな局面に頻繁に出会うようになったからである。そのポイントとは、たとえば二重ループであり、あるいは配列変数であり、時にはアルゴリズムや論理的思考そのものが理解困難なポイントであるように思われることも少なくない。これまで情報科学の世界では当たり前の前提となっていた、テキストワークによるプログラミング教育が当たり前でなくなりつつあるという現象につきあたったことが、開発のきっかけとなった。考えてみれば現在のような、パソコンの学習の当初からウィンドウアプリケーションとマウスやアイコンを操作しなれた新しい世代のコンピュータ利用者にとって、テキストによるコマンドとパラメータでコンピュータを操作するやり方は、その世界に入り込むだけで多大なストレスと発想の転換を強いられる作業なのかも知れない。このような状況で、しかも数理的思考の訓練をさほど受けていない学生達が悪戦苦闘するというのも、これは必然のような気もするのである。

そこで、今回発表した DFC の開発に至るのであるが、しかし現状 DFC は、それだけでアプリケーションが開発できるようなものではもちろんなく、今のところアルゴリズム学習のための教材というものでしかない。高々数個の変数と数種類のコマンドしか持たず、数ステップのプログラムが組めてデバッグ環境の中だけで実行できるものに過ぎない。しかし、仮にこのようなvisualなプログラミング環境が、現在のウィンドウシステムに慣れた学生達のアルゴリズム学習に効果的であるとすれば、次に必要なものは、visual な開発環境からテキストワークの開発環境への移行をスムーズにする環境であろう。この両者のボーダーを取り払うアイディアは、すでに Java 版 DFC で実現されているソースプログラムの出力と、同じく Java 版の拡張機能として検討されていた、サブルーチンの生成機能である。部品の開発、または逆にメイン関数の開発に DFC を使用し、他の部分は他言語で開発し、両者をリンクして実行プログラムを生成できるようになれば、既存の言語システムとの融合が実用レベルで可能になると考えられる。無論、現段階でこれがそのままアプリケーションなどの開発に直結することは考えられないが、いわゆる第 4 世代言語の一種としてより本格的なプログラム言語教育教材と捉えることも可能ではないかと思っている。いわばこれが DFC の最終目標とでもいえるだろう。ユーザレベルの知識や習熟で、アプリケーション開発が可能となる程度に機能やライブラリが拡張されれば、社会科学系の学生に対して本来教育するべき、エンドユーザコンピューティングの目的が果たされるということになるのが、本システムに関わった者の見る最良の夢であろう。

なお、DFC の Java 版と Flash 版は以下のサイトで公開しており、自由にダウンロードすることができる。また、これらのソフトは今後も改良を続けていく予定で、以下のサイトでは常に最新版を掲載していく。

http://silicon.mis.oiu.ac.jp/DFC/

参考文献

中井、岡本、下條、“Flashによるアルゴリズム学習教材の開発”、平成17年度大学情報化全国大会, A-10, 2005-09, pp.148-149

中井、岡本、下條、“プログラミングにおける補助教材の開発”、平成16年度大学情報化全国大会, C-3, 2004-09, pp.146-147

高見　他、“画像プログラミング演習によるＣ言語導入授業の試み”、平成14年度大学情報化全国大会, C-12, 2002-09, pp.182-183

新開、大森、“インタプリタ教育のための Web 教材の開発”、情報処理教育研究集会講演論文集, 2001-10, pp.217-220

「Flashユーザーガイド」、マクロメディア株式会社、2004年６月

「ActionScriptリファレンスガイド」、マクロメディア株式会社、2004年６月