コンピュータと教育　62－2
（２００１．　１２．　１４）

文科系向けプログラミング教育支援システム JPADet の設計と開発

河村一樹(東京国際大学)，斐品正照(宮城大学)，徳岡健一(日本科学技術研修所)

概要:我々は，文科系学生のためのプログラミング教育を支援するシステムである JPADet を設計し開発した。JPADet を用いると，プログラミング言語に依存することなく，日本語と構造化チャート(PAD)を使ってアルゴリズムを実装することができる。また，インタプリタ方式なので，プログラム(アルゴリズム)の検査も容易である。本稿では，JPADet の設計仕様と開発環境と操作例について取り上げる。

Design and Development of Programming Education Support System JPADet(Japanese PAD Editor and inTerpreter) for Liberal Arts Students

Kazuki Kawamura(Tokyo International University),Masateru Hishina(Miyagi University),Ken-iti Tokuoka(Institute of J.U.S E)

Abstract *: We designed and developed the JPADet that is the system that supports the programming education for liberal arts students.. The JPADet can do implementation of the algorithm by using Japanese and structure chart (PAD), without relying on programming language. Also, JPADet is the interpreter system, it is easy to inspect the program (algorithm) .In this paper, we discuss about the design specification and developmental environment and chastity model of JPADet.*

1．はじめに

情報処理を専門とする学生を対象にしたプログラミング教育は，すでに各大学の情報系学科で実施されている。その基盤となるカリキュラムとしては，情報処理学会の CS 部会が策定したカリキュラム J97[1]があげられる。この中で，プログラミング教育に関する科目として，「プログラミング入門(L-2)」「形式言語とオートマトン(U-2)」「データ構造とアルゴリズム(U-3)」「プログラミング言語論(U-4)」などが取り上げられている。それぞれの科目では，基礎理論から，抽象化あるいはモデリング，さらには実装技術に至るまでの内容が体系的に網羅されている。

これに対して，情報処理を専門としない利用者の立場である文科系学生に対するプログラミング教育については，情報処理学会の一般情報処理教育部会が策定した調査研究報告書[2]以来公開されたものがない。この報告書では，プログラミング教育に関して，「プログラミング」教育(かぎ括弧付き)とプログラミング教育(かぎ括弧なし)とに分けた。前者は，特定の実用言語の習得だけを目的とせずに，コンピュータサイエンスの基本的な概念を理解するための技能を理解するための教育である。後者は，実用言語の習得を目的としたプログラマ養成のための教育である。一般情報処理教育におけるプログラミング教育は，前者を目指すべきと提言した。ただし，そのためのカリキュラムモデルの提案はなく，実際に行われている教育事例の紹介で終わっている。

以上のことから，一般情報処理教育におけるプログラミング教育について，実用言語に依存することなく，コンピュータサイエンスの基本的な概念を理解させるための方策が必要になってくる。

一方，2003 年度からは，高等学校普通科において教科情報が新設されることになる[3]。その中の科目「情報 B」では，モデル化とシミュレーションという単元でプログラミングが扱われることになる。ここでも，「プログラミング言語の習得が目的とならないように」という留意点が明示されている。これより，上述した一般情報処理教育における「プログラミング」教育とほぼ同じような状況におかれることになる。

このようなことから，我々は，大学での一般情報処理教育および高等学校での教科情報教育を視野に入れ，文

科系学生向けのプログラミング教育支援システムを開発した。本稿では，開発の経緯，JPADet の設計と開発について述べる。

2．開発の経緯

筆者の一人である河村は，以前から PAD を用いたプログラミング教育についての研究[4)~6)]を行っていた。当初は，COBOL プログラマを育成するためのプログラミング教育において，アルゴリズムの設計技能を育成するための授業を担当していた。その際に，アルゴリズムを設計してから，COBOL でプログラミングをさせて実行結果を検証するという演習形態をとっていた。しかし，この方式では，アルゴリズムの検証(おもに，論理面)までに時間がかかること，学生はアルゴリズムよりも COBOL 言語の方に興味を示すこと，などの問題が生じた。

そこで，日本科学技術研修所(以降，日科技研と略す)が開発し製品化していた PADET/C という PAD の編集とプログラミング言語への変換機能を持つ PADET(PAD Editor and Translator)の C 言語版を用いた演習形態に変更した。しかし，PADET/C は，C 言語に変換するので，いくつかの不適合な面が明らかになった。たとえば，

- 処理箱の中には，文の区切りとしてかならず「;」を入れなければならず，COBOL プログラマにとっては「.」でないため違和感があること
- 選択文や繰り返し文で指定する条件式は，C 言語風に書かなければならず，学生が混乱すること
- COBOL の再定義(REDEFINES)が扱えないこと
- ファイルのアクセスについては，C 言語のファイルの扱いに制限があり，COBOL らしいプログラム(いろいろなファイル編成を用いたアルゴリズム)が作成できないこと
- 画面の入出力処理については，C 言語特有の記述が必要になり，COBOL プログラマにとっては理解しずらい(これについては，マクロ関数を独自に作成して対応)

このようなことから，日科技研と共同で，COBOL 版の PADET/CBL を開発することになった。当時の日科技研は，PADET/B(BASIC 版)，PADET/F(FORTRAN 版)，PADET/C(C 言語版)を製品化していた。この中の PADET/C を改変することで，その実装には C 言語を用いた。開発した PADET/CBL を実際に授業で使い評価した結果，アルゴリズムの設計能力が向上することが明らかになった[7)~10)]。

以上のような経緯があったことから，今回も日科技研と，新たに日本語と PAD を用いたプログラミング教育支援システムである JPADet(Japanese PAD editor and interpreter)を共同開発することになった。

3．JPADet の設計と開発

ここでは，JPADet に関する設計仕様と実装環境に関してまとめる。

3．1 基本仕様

(1) 前提条件

本研究では，非情報系の学生(文科系大学生あるいは高校生)を対象に，問題解決手順(アルゴリズム)の習得，論理的思考力および抽象的概念の理解力の育成，を目指すための教育支援システム JPADet の構築と教育現場での実証実験による評価を目指している。このため，JPADet の設計にあたっては，次のような前提条件を設定した。

a. 実務用ではなく教育用を前提にすること
b. 大学の一般教育課程(1・2 年生)におけるプログラミング演習科目や高等学校教科目「情報 B」での演習に適用することを想定すること
c. 実用言語に依存することなく，アルゴリズムの設計が学習できること
d. 構造化チャート(PAD)を用いることで，アルゴリズムを視覚的に記述できること
e. 日本語を使用して簡単な式表現ができること
f. トランスレータではなく，インタプリタとして実装すること

このうちの f のインタプリタ採用の理由として，開発工数の軽減，インストール時の設定が単純，トランスレータのようなコンパイラ製品毎の差を考慮する必要なし，利用者にとって言語の知識はまったく不要，学習解析データなど独自に追加可能，などがあげられる。

(2) JPADet の言語仕様

ここでは，JPADet の箱内記述の言語仕様についてまとめる。

① データ型

数型(整数と浮動小数点の区別は利用者に対して隠蔽)と文字型を用意する。

② メインと副プログラム

JPADet 上の手続きは，プログラムファイルにただ一つ存在するメインと，関数もしくは手続きとして実行される副プログラムから構成される。副プログラムは，式あるいは文が書ける箇所で，

副プログラム名(実引数，…)

と書くことで呼び出せる。実引数と仮引数の数は一致していなければならない。副プログラムを関数として用いる場合は，単一の戻り値を持つ。引数は値渡しのみで参照渡し(関数内の引数への代入が呼出し側の実引数変数へ戻ること)はしない。

③ 制御構造

制御構造はすべて PAD(処理箱，双・多岐選択箱，前判定繰返し箱，後判定繰返し箱，条件値判定繰返し箱，入力箱，出力箱)で記述されるので，箱内記述は制御構造を持たない。なお，goto 文は扱わない。

④ 変数

使用する変数は，専用の宣言ダイアログで宣言する。

メインで宣言された変数は，プログラム全域に対応するスコープを持つ。副プログラムで宣言された変数には，仮引数(呼出しプログラムから引数として受け取る変数)，一時変数(副プログラム内でのみ有効な変数)，「結果」(値を返す副プログラムでの結果を代入する変数)がある。ただし，関数が終了した時点で「結果」に設定された値が戻り値になる。このため，明示的なreturn箱は持たない。

⑤ 配列

変数は配列として使用することができる。配列変数は，

変数名［添字，…］

の形で参照および代入する。

⑥ その他のデータ構造

ポインタと構造体については，複雑になるので扱わない。ファイルに関しては，シーケンシャルアクセスだけにし，テキスト関数で処理を記述する。

(3) JPADetの文法定義

構文解析に伴う構文規則は，次のように定義する。ただし，**表1**のような(BNFに近い)記法を用いる。

表1．記法の意味

A=B	生成規則(AとはBである)
名称	終端記号か非終端記号
"....."	終端記号
X...	Xの1回以上の繰返し
[X]	省略可能なX
[X]...	Xの0回以上の繰返し
a|b	aまたはb

処理　process=statement....
命令　statement=assignment | call
代入文　assignment=variable "=" expression
変数　variable=ident ["[" expression_list "]"]
式列　expression_list=expression ["," expression]...
呼出し　call=ident ["(" expression_list ")"]
式　expression=term [op term]...
2項演算子　op="+" | "-" | "*" | "/" | "%" | "^" | "<" | ">" | "=" | "<=" | ">=" | "<>" | "&" | "|"
項　term = variable | call | number | string | uni_op term | "(" expression ")"
単項演算子　uni_op="+" | "-" | "~"
変数列　variable_list=variable ["," variable]...
条件値判定繰返し　for=variable ":" expression_list

上記の中の2項演算子の優先順位は，次の通りとする。

高い	* % ^
↑	+ −
	< > <= >= <> =
↓	&
低い	|

また，構文上のトップレベルは，箱の種類によって**表2**のように異なるものとする。

表2．箱と構文の関係

箱の種類	構文
処理箱	処理
多岐選択 前判定繰返し 後判定繰返し	式
条件値判定繰返し	条件値判定繰返し
入力	変数列
出力	式列

3．2 実装環境

(1) 動作環境

JPADetが動作するOSは，Windows95/98/MeおよびWindowsNT4/2000とする。

(2) 開発環境

JPADetの開発には，Microsoft Visual C++6.0およびクラスライブラリとしてMFC(Microsoft Foundation Class)を用いる。また，JPADetは，MDI(Multiple Document Interface)アプリケーションとして開発する。

(3) 編集系

図1に示すように，編集系はCJPadetDoc(jpdファイルの内部表現)を中心に，定義された関数・手続き(CProcedure)，関数・手続き内の変数(CVar)，PADシンボル(CAlg)を管理し，要求に応じてこれらを編集する。

① CJPadetDoc(ドキュメント)

下位の構成要素である関数群を管理する。また，MFCフレームワークによるシリアライズにより，ファイルのロードとセーブも行う。関数ダイアログ(CProcDlg)を経由して，関数の追加，削除，オープン処理を行う。

② CProcedure(関数，手続き)

下位要素として，変数と処理シンボルを持つ。関数ダイアログからの要求により，対応するCJPadetViewと結びついて画面上に描画される。また，結びついている(自身を表示している)ビューからの要求により，処理シンボルの木構造を編集する。変数一覧ダイアログを経由して，関数中の変数リストを編集する。

③ CAlg(処理シンボル)

属性として，種別とソース記述(任意のテキスト)を持つ。種別は，次の通りである。

```
enum alg_code {
  unit_alg,        先頭要素
  process_alg,     処理箱
```

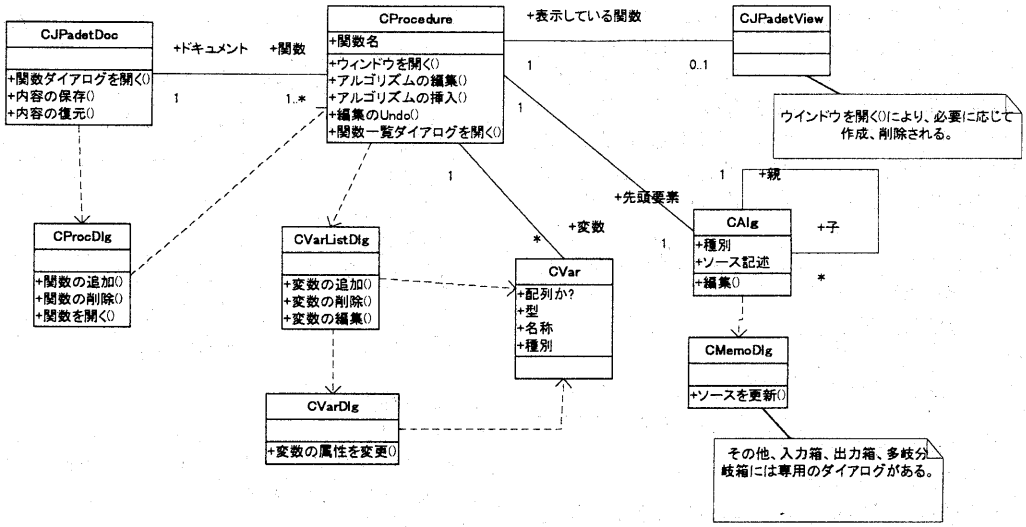

図1. 編集系のUML

for_alg,	条件値判定繰返し
while_alg,	前判定繰返し
repeat_alg,	後判定繰返し
switch_alg,	分岐処理
case_alg,	分岐処理中の個々の分岐
input_alg,	入力箱
output_alg	出力箱

};

任意個の子を持つことができ，unit_alg を先頭とする木構造を構成する。unit，for，while，repeat，case は，unit，case 以外の子を持つことができる。switch は，case のみを子として持つことができる。

④ CVar(関数，手続き毎の変数)

属性として，名称，配列型か否かのフラグ，型(数値か文字列か)，種別(大域変数，ローカル変数，引数，結果変数)を持つ。

(4) 表示系

図2 に示すように，表示系は，CJPadetView(MDI の関数表示画面に相当)が中心になる。CJPadetView は画面上の表示要素オブジェクトを管理し，利用者からのマウス操作やメニュー操作を翻訳し，CProcedure を経て編集操作を支援する。

① CJPadetView(MDI の関数表示画面に対応)

表示している関数(CProcedure)と一対一に対応していて，CProcedure 下のアルゴリズム木から画面上の表示要素を作成して維持する。表示要素群は，CProcedure が編集される度に再編成される。また，利用者のマウス操作やキーボード操作によって，シンボルの選択動作を管理し，編集指示に応じて CProcedure へ編集を要求する。

② CFigure(画面上の表示要素の基底クラス)

画面上での表示領域の座標を維持する。また，要求に応じて，画面上に自分自身を描画するメソッドと選択表示を行うメソッドがある。

③ CLine(画面上の横線)

たんに表示されるだけで，選択はできない。

④ CSelectFig(画面上の選択可能な要素)

CAlg への参照と，その CAlg との相対的な関係を維持している。関係はシンボル自身(CAlgFig)，シンボルの下方(CTerminalAlg，CGlueAlg)，シンボルの子の先頭(CTerminalAlg)がある。これにより，どのシンボルがどのように選択されているかが決定されるので，適切な編集を行うことができる。

⑤ CAlgFig(多岐選択箱以外の箱)

参照している CAlg により表示形式が異なる。箱の内部には，CAlg のソーステキストがそのまま表示される。

⑥ CSwitchAlg(多岐選択箱)

多岐分岐の区切りの位置を維持していて，多岐選択箱を描画する。表示されるソースも対応している switch_alg ではなく，下位の case_alg 群のソースを表示する。

⑦ CGlueFig(画面上の縦線)

⑧ CTerminalFig(シンボルの上下端の選択領域に対応)

選択表示の時には反転されるが，通常では表示されず，位置情報だけが維持される。

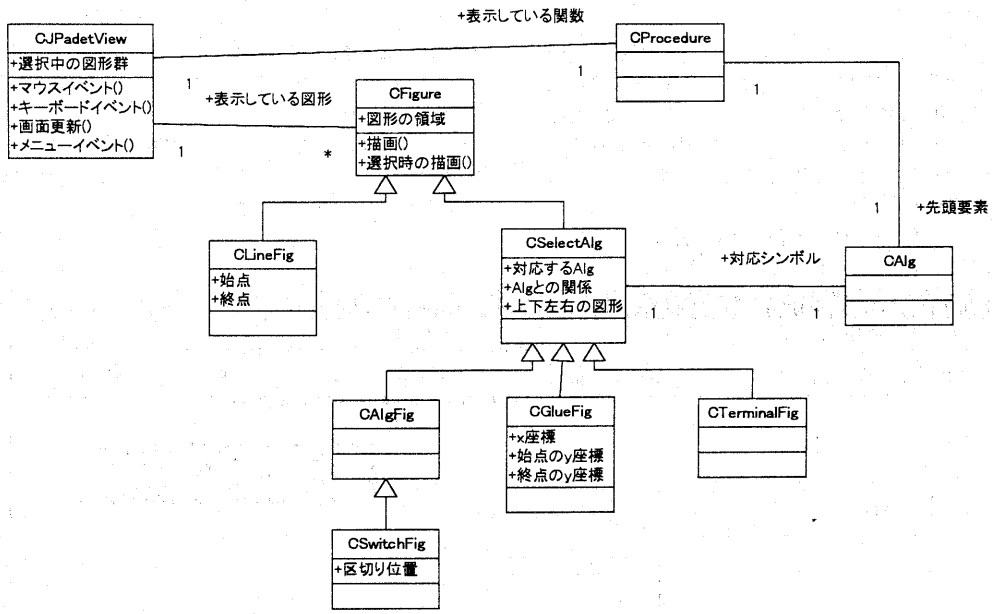

図2. 表示系のUML

(5) 実行系

実行系は，CContextが中心になる。CContextが一つの関数を実行するための環境(変数テーブル，現在実行中の関数(CProcedure)，シンボル(CAlg))を維持している。関数呼出しが行われると，新たにコンテキストが作成される。

① 字句解析

字句解析は，一文字先読みで処理する。その際に，マルチバイト文字は，対応するシングルバイト文字があればシングルバイト文字に変換して処理する。たとえば，×は*へ，÷は/へ，≠は<>へ，≦は<=へ，≧は>=へ，それぞれ変換する。字句としては，識別子(ident)，文字，文字列(string)，数値(numeric)とする。

② 構文解析

上述した構文規則にしたがい，単純な再起下降型パーサーで処理する。

③ 実行

実行は，CProcedure上のCAlgを先頭から解析結果に応じてtree-walkすることで行う。CAlgのソーステキスト解析と同時にコンテキストの変数参照を行い，同時に式評価も行うことから，中間言語は生成しない。

3．3 操作手順

ここでは，JPADetの操作手順を通して，JPADetのシステム構成や表示デザインについて取り上げる。

(1) 起動

JPADetの実行ファイルへのショートカットアイコンをダブルクリックする。

(2) 初期画面の表示

JPADetが起動すると，**図3**の中で，タイトルバー，メニューバー，ツールバー，シートウィンドウ(mainの箱のみ表示)，ステータスバーから構成される初期画面が表示される。

このうちのツールバーは，通常よく使われるメニューコマンドとシンボルを配置するコマンドをアイコンとして登録している。

これより，JPADetでは，PADシンボルとして，左から，処理箱，前判定繰返し箱，後判定繰返し箱，条件値判定繰返し箱，分岐処理箱，入力箱，出力箱を採用している。

(3) 変数の作成

使用する変数は，あらかじめ変数一覧ダイアログを用いて宣言する。「追加」を選ぶと，[変数の編集]ダイアログが表示され，ここで変数を作成する。変数名には，日本語を指定できる。変数の型には，数値(整数か浮動小数点)と文字[列](単一文字か文字列)と配列(添字の開始番号は0から)を指定できる。

(4) PADシンボルの記入

シンボルを配置するには，シート上のシンボルを配置したい位置にカーソルを移動し，シンボルバーの中から配置したいシンボルのアイコンをクリックする。または，[命令の追加]メニューから配置したいシンボルを選択する。

① 位置の指定

シンボルを配置したい位置は，マウスまたはキーボードの方向キーで選択できる。実際にシンボルを配置する箇所は，短形(■)で表示される。シンボルが配置可能な場合は，ツールバーの各シンボルボタン(図4の四角で囲んだ部分)，および，[命令の追加] メニューの各シンボルがアクティブになる。

シンボルの上下へ挿入する場合は，左隅やシンボルをつなぐ線をクリックする。

各 [繰返し] シンボルや [分岐] シンボルで，アルゴリズムを右に展開する場合は，シンボルの右端をクリックする。以上の関係を，**図3**に示す。

② シンボルの配置

シンボルが配置可能な状態で，[命令の追加] メニューから配置したいシンボルを選択すると，短形で表示されていた箇所にシンボルが配置される。

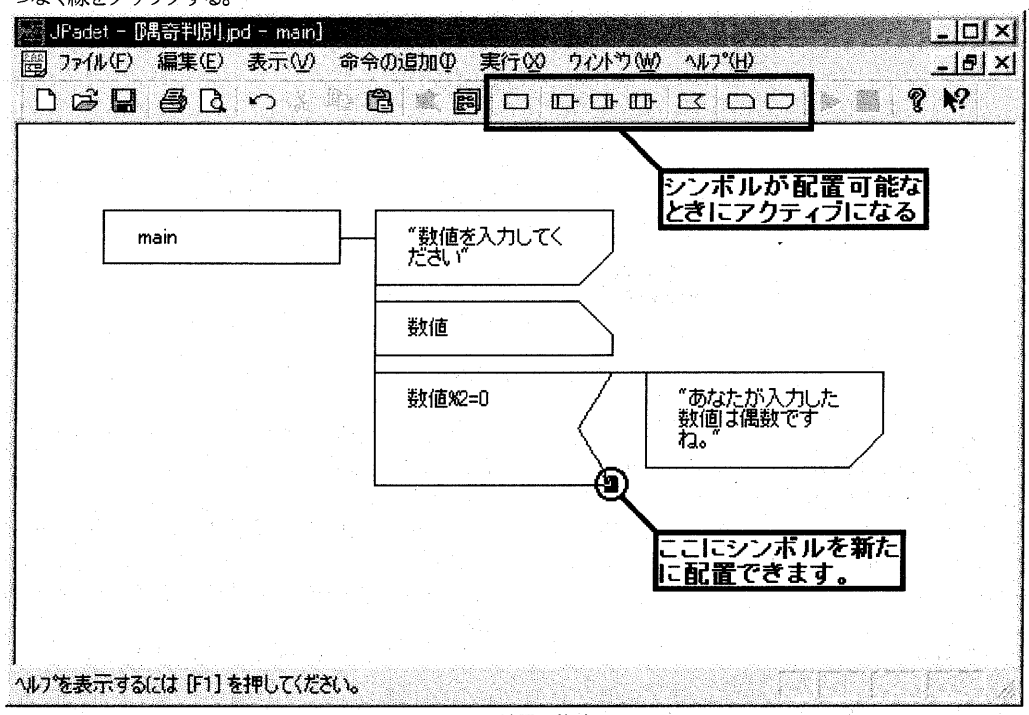

図3. 位置の指定

(5) PAD シンボルの編集

PAD シンボルの編集には，次のようにそれぞれ該当するダイアログが表示され，その中に記述する。

- [処理] シンボル：[式] ダイアログ(代入式や計算式を記入)
- [前/後判定繰返し] シンボル：[式] ダイアログ(繰返し条件と，条件が真のときの処理を記入)
- [条件値繰返し] シンボル：[式] ダイアログ(繰返し制御変数名：開始値，終了値，増減分値，，および，条件は真のときの処理をそれぞれ記入)
- [分岐] シンボル：[分岐] ダイアログ(省略時は分岐が1個だか，多岐選択のため分岐を増やすことも可能。選択条件を記入)
- [出力] シンボル：[出力] ダイアログ(画面かファイルかを選択した上で，出力したい変数や式を記入)
- [入力] シンボル：[入力] ダイアログ(キーボードかファイルかを選択した上で，入力する変数を記入)

(6) プログラムの実行

作成したプログラムの実行を行う場合は，[実行] メニューの [実行] を選択する。また，ツールバーの [実行] ボタンをクリックしてもよい。これによって，実行ウィンドウが開き，プログラムが実行される。

また，プログラムを実行した際に，プログラムに記述エラーが含まれている場合は，エラーメッセージが表示され，エラー箇所のあるシートを開きシンボルを選択状態にする。その後，エラーメッセージにしたがい，プログラムのエラー箇所を修正する。

(7) プログラムの印刷

プログラムの印刷については，印刷するシートのウィンドウをアクティブにして，[ファイル] メニューの [印刷...] を選択する。これにより，[印刷] ダイアログが表示されるので，各項目を設定し，[OK] ボタンをクリックする。なお，印刷プレビューも可能である。

(8) プログラムの保存

プログラムの保存については，[ファイル]メニューの[名前を付けて保存...]を選択し，[ファイル名]ダイアログボックスにシートの名前を入力し，[保存]ボタンをクリックする。

3．4　サンプルプログラムの例

ここでは，等差数列のサンプルプログラムを JPADet で作成した例を示す。

(1)　プログラム課題

等差数列の初項(a)，公差(d)，項数(n)を入力して，初項から第n項(nは項数)までの各項の値(a_n)を求める。具体的には，$a_n=a+(n-1)*d$ という関係になる。

(2)　変数一覧ダイアログ

本サンプルプログラムの変数一覧ダイアログは，**図4**のようになる。

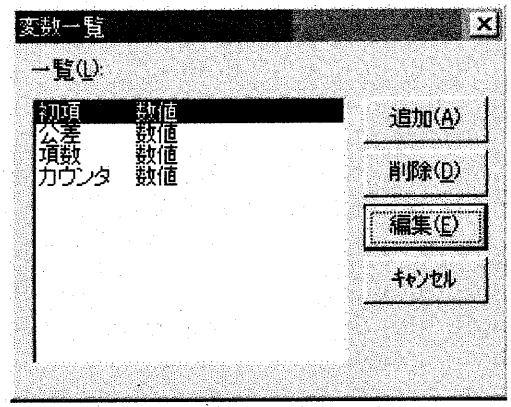

図4．変数一覧ダイアログ

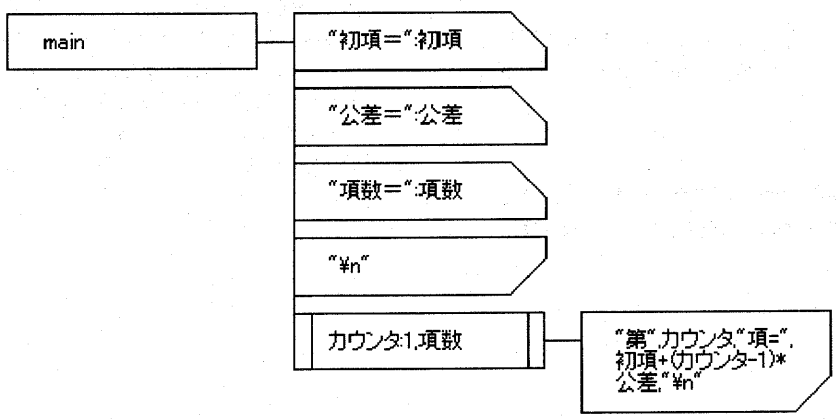

図5．プログラム課題のPAD

(3)　JPADet のメインシート

変数一覧をもとに，実際のPADは**図5**のようになり，これをメインシート上に記入する。このように，main箱の右側に，PADシンボルを，上から下へ(順次処理)，右から左へ(選択処理か繰返し処理)，展開していくことになる。

(4)　実行結果

[実行]メニューから実行すると，実行ウィンドウが開き，初項の問合せとなる。初項・公差・項数を入力すると，右のような結果が表示される。

```
初項=5
公差=3
項数=6

第1項=5
第2項=8
第3項=11
第4項=14
第5項=17
第6項=20
実行を終了しました
```

4．おわりに

以上，JPADetに関する設計仕様と開発環境，および，操作手順と事例について述べてきた。

JPADetの特徴としては，実用言語に依存することなく日本語とチャート(PADを採用)を用いてアルゴリズムを記述することができるとともに，インタプリタ方式により簡便にプログラムの翻訳実行ができプログラムの動作確認が短時間でできること，などがあげられる。これより，文科系の学生や高校生にとっても，問題解決手順習得のための学習支援システムとして適しているといえる。

JPADetは，すでに開発を終え，その実行プログラム(exeファイル)ができあがった状況にある。また，我々の方では，JPADetに関連したWBE(Web Based Education)も別途開発している。これらを用いることで，ウェブ上での自学自習教材としても利用できるような試みを進めている。

3．1の基本仕様の前提条件で述べたように，JPADetは大学の一般教育課程(1・2年生)の情報処理関連の科目や高等学校教科情報での適用を想定している。そこで，これらの教育現場でJPADetを適用し，その教育効果を実証的に評価する必要がある。これらについては，現在準備段階であり，今後の研究課題としたい。

謝辞 本研究にあたり，共同研究の機会を与えて頂いた(株)日本科学技術研修所営業本部長の高田克美氏に感謝致します。

参考文献

1) 情報処理学会：大学の理工系学部情報系学科のためのコンピュータサイエンス教育カリキュラムJ97，情報処理学会，1997年
2) 情報処理学会：大学等における一般情報処理教育の在り方に関する調査研究，情報処理学会，1993年
3) 文部科学省編：高等学校学習指導要領，文部科学省，1999年
4) 河村一樹：PADによる構造化プログラミング教育について，情報処理学会第35回全国大会予稿集，1987年
5) 河村一樹：PADによる構造化プログラミング，啓学出版，1988年
6) 河村一樹：PADによる構造化プログラミング習得のための技術教育について，CAI学会誌，Vol.6，No.1，pp.26-31，1989年
7) 河村 一樹：COBOL版PADによる構造化プログラミング，啓学出版，1989年
8) 新田雅道，河村一樹：教育用構造化プログラミングエディタ(PADET/CBL)の設計・開発，情報処理学会第39回全国大会予稿集，1989年
9) 河村一樹：構造化プログラミングエディタPADET/CBLの開発と評価，尚美学園短期大学研究紀要，第6号，pp.77-94，1992年
10) 河村一樹：構造化プログラミングエディタPADET/CBLの開発と教育での適用，教育システム情報学会誌，Vol.14，No.4，pp.151-160，1997年
11) 二村良彦：構造化プログラム図式，コンピュータソフトウェア，Vol.1，No.1，pp.64-77，1984年
12) 二村良彦：プログラム技法—PADによる構造化プログラミング—，オーム社，1984年
13) 川合敏雄：PADプログラミング，岩波書店，1985年