

プログラミング入門教育用学習環境 PEN

中村 亮太[†] 西田 知博^{††} 松浦 敏雄[†]

職業プログラマを目指すのではなく、プログラミング、および、コンピュータの仕組みを理解することを目的としたプログラミング入門教育を行う上で有効な、初学者向けのプログラミング学習環境 PEN を開発した。PEN では、大学入試センター等の入試で用いられている言語を用いているので、付加的な説明を行わなくても容易に理解できる。また、プログラムの入力補助機能を備えることで、プログラムの作成時の誤りの混入を減らすのに役立つ。また、ステップ実行機能、スロー実行機能、変数表示機能などにより、プログラムの動作を観察しやすくしている。ここでは、PEN の概要と操作ログの分析による有効性の検証について報告する。

The Programming Environment for Novice Education

RYOTA NAKAMURA,[†] TOMOHIRO NISHIDA^{††} and TOSHIO MATSUURA[†]

We developed PEN, a programming environment for novices. It is not for the training to bring up experts, but for the learning to understand what the programming is and how the computers work. The programming language used in PEN is based on DNCL, the programming language for the test carried out by National Center for University Entrance Examinations. The key words are written in Japanese, therefore users can understand the meaning without the additional explanation. We provide supporting functions for inputting a program to avoid syntax errors occurred with the type mistake. In addition, users can watch how the programs work by using the functions; step-by-step execution, speed control of execution and watching variables. In this paper, we show the outline of PEN and verify the effectivity of the system by using the operation logs.

1. はじめに

高校および大学での情報教育において、プログラミング入門教育は軽視されているように思われる。高校の教科「情報 B」でも取扱われているが、その指導要領では、あまり深入りしないように釘をさされており、実際には、プログラミング言語を用いた実習を伴うような授業はほとんど行われていない。大学においても、理系の一部の学部/学科を除いて、同様の傾向がみられる。プログラミングが取り上げられない大きな理由としては、プログラミング言語は難しいものであり、その習得には多くの時間が必要とされていることが挙げられる。しかし、プログラミングの入門教育の目標を、職業プログラマの養成を目指すのではなく、プログラミングとは何かを理解し、コンピュータの本質を理解することと定めるならば、適切なサポートツ

ルを用意することで、比較的短い時間でこの目標を達成できると思われる。

本研究は、上記のプログラミング教育を達成するためのサポートツールとして、入門教育用プログラミング環境 PEN¹⁾ (Programming Environment for Novices) を構築し、その有用性を明らかにすることを目的とする。

既存のプログラミング入門用の学習環境を提供する研究として、ドリトル²⁾ や Nigari³⁾ などがある。これらはいずれも、学習の導入段階から、オブジェクト指向の考えでプログラミングの学習を行うものである。オブジェクト指向の考え方は極めて重要ではあるが、メソッドの記述は手続き的に書かざるをえず、オブジェクト指向の考えに至る前段階で手続き的なプログラミングの習得が必要と思われる。

PEN は、手続き的なプログラミングが容易に習得できるような、プログラミング学習環境の提供を目指している。PEN は、分かりやすいプログラミング言語を備え、構文エラーなども生じにくくするための入力支援機能を備えており、また、プログラムの実行の様子を把握しやすくするための機能も備えている。

[†] 大阪市立大学大学院創造都市研究科
Graduate School for Creative Cities, Osaka City University

^{††} 大阪学院大学情報学部
Faculty of Informatics, Osaka Gakuin University

2. プログラミング教育に必要な支援機能

ここではまず、2.1 において従来のプログラミング演習で初学者が直面する問題を挙げる。2.2 では、本研究で考えるプログラミング教育の目標を明白にし、2.3 では、その教育目標を達成するためのプログラミング環境としてどのような機能が必要かについて考察する。

2.1 プログラミング演習の難しさ

初学者は、プログラミング演習の初期の段階でつまづくことが多い。演習初期段階ではタイプミスに起因する文法エラーが多く発生し、意味の理解できないコンパイルエラーメッセージを(多くの場合英語で)突き付けられ、それだけで自信をなくす者も少なくない。文法エラーの修正ばかりに気を取られていると、プログラムの構造をどのように組み立てるかといった、全体への配慮ができず、プログラミングの力がなかなか身につかないことが多い。また、コンパイルエラーがなくなっても、論理的なエラーのために予想していたのと違う結果が返ってくると対処することが難しい。論理的なエラーはエラーメッセージが表示されないのので、間違い箇所をみつけるためには、変数に何が代入されているかを逐次チェックしていかなければならない。デバッグを用いれば、このようなチェックは比較的容易に行うことができるが、多くのデバッグは初学者が使うことを考慮しておらず、使いにくいことが多い。

2.2 「プログラミング」入門教育の目標

はじめにでも述べた通り、ここで想定している「プログラミング入門」は、大岩レポート⁴⁾でいうところの「プログラミング」を目指すものであり、職業技能としてのプログラミング言語の習得を目的とするものではない。「プログラミング」習得の目的は、文献⁵⁾に示されている通りであり、要約すると次の通りである。

プログラミングを、決まった正解を求める機械的な作業のようにとらえるのは、まったくの間違いである。プログラミングは、無限の解法が存在する、まさに創造的な作業というべきであり、その過程そのものが教育的に有益である。自分が考えたことを決められたルールに従って、きちんと記述するという訓練は、自分の知識状態や考え方を明確にする力を養える。また、プログラムのデバッグは、間違い発見のプロセスであり、自ら発見するという学習そのものであり、自立的な思考力を養える。このような効果を得るためには、プロ

グラムを書いて、実際に動かしてみるということから自ら体験する必要がある。

さらに、プログラミングを体験することで、コンピュータというものの本質を理解できるようになる。すなわち、コンピュータは、プログラムに書かれたことを忠実に実行するだけの機械であって、それ以上でもそれ以下でもないことを実感できる。また、考えたことをプログラムに書き表すのは容易ではないことや、コンピュータのプログラムからバグを取り去るのは容易でないことなども自分の経験として理解できるようになる⁶⁾。

このように「プログラミング」の習得は、上記のような体験にもとづいて、コンピュータの本質を理解することを目標としている。

2.3 初学者用プログラミング環境として何が必要か

2.2 で示したプログラミング入門を、なるべく短時間に習得するために必要なプログラミング環境について考える。

まず、プログラミング言語としてどのようなものを選ぶかが重要な問題である。読んで分かりやすい言語であることが第一の条件であろう。そこで、我々は、大学入試センターの「情報関係基礎」で用いられている手順記述言語 DNCL⁷⁾、および、DNCL をもとに作られた東京農工大の入試用手順記述言語 TUATLE⁸⁾ に注目した。これらの言語は日本語をベースとした記述言語であり、特別な説明なしで入試で用いられているということからも、分かりやすさの点では問題ないと思われる。

DNCL および TUATLE は、読みやすさの点では、初学者に十分受け入れられるレベルであるが、書くのは必ずしも容易ではない。日本語をベースとしていることから、キーワードなども長めであり、入力において、文法エラーが入り込む余地は大きい。そこでプログラムを書く際の何らかの入力支援機能が必要となる。

また、2.2 で述べたように、プログラミングは自分で書いて実際に動かしてみることが重要である。そのためには、言語処理系が必要であり、さらに、論理的な間違いを発見しやすいように、初学者がプログラムの実行状況を観察できる機能等が必要である。

3. 初学者向けプログラミング学習環境 PEN

図 1 は PEN の実行時のスナップショットである。PEN はプログラムの入力/編集を行うためのエディタ機能、プログラムの実行・一時停止・一行実行などの実行制御機能、実行結果とその履歴を表示するコンソール機能、実行中の各変数の値を表示する機能等をもつ。

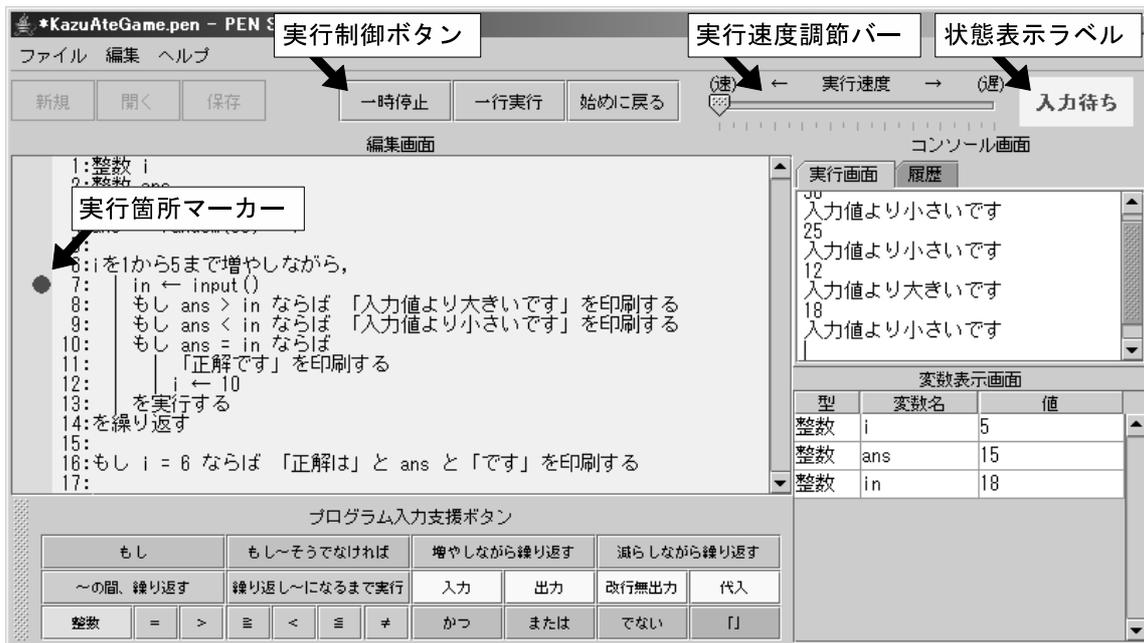


図 1 PEN の実行時の表示例

PEN で動作するプログラム言語は、DNCL および TUATLE を拡張した言語を用いた。これを xDNCL とよぶ。図 2 に xDNCL によるプログラムの記述例を示す。言語仕様は DNCL に、変数宣言の追加と演算子の定義で若干の変更を行っているが、基本的にはほとんど同じである。図 2 からわかるようにプログラムを読むのは容易である。

3.1 プログラム入力支援機能

3.1.1 プログラム入力支援ボタン

3 で述べたように、xDNCL で記述されたプログラムは読みやすいが、日本語で記述されているので、プログラム作成時は文法通りに間違わず入力することは必ずしも容易ではない。また、キーボードに慣れていない初学者にとっては、かな漢字変換の操作も煩雑である。

キー入力の操作を減らし、補助するための機能として、図 1 の下部のようなプログラム入力支援ボタンを用意した。

以下に、プログラム入力支援ボタンの使用例を示す(図 3)。

(1) 図 3 (a) の状態で入力支援ボタン [~の間、繰り返す] をクリックすると、図 3 (b) のように制御構造のテンプレートが編集画面に挿入される。(2) 次に条件式 など、と で囲まれた部分(この部分全体で一つの文字として扱われる)をマウスでクリックするか、その位置にカーソルを移動することにより、

```

整数 a, x, i
i を 1 から 3 まで 1 ずつ増やしながらか、
i と「回目の勝負です」を印刷する
「手を選んでください」を印刷する
「グー(0)、チョキ(1)、パー(2)」を印刷する
a ← input()
x ← random(2)
もし x = 0 ならば
  「私の手はグーです」を印刷する
を実行し、そうでなければ
  もし x = 1 ならば
    「私の手はチョキです」を印刷する
を実行し、そうでなければ
    「私の手はパーです」を印刷する
を実行する
を実行する
もし a = x ならば
  「あいこです」を印刷する
を実行し、そうでなければ
  もし (a + 1) % 3 = x ならば
    「あなたの勝ちです」を印刷する
を実行し、そうでなければ
    「あなたの負けです」を印刷する
を実行する
を繰り返す

```

図 2 xDNCL の記述例(ジャンケンゲーム)

図 3 (c) のように と で囲まれた部分が選択される。(3) 選択された部分を実際の条件式等へ書き換えて、さらに 4 行目以降に命令を挿入していくことにより、プログラムを作成することが可能である。

3.1.2 インデントの自動挿入

DNCL 等では、インデントを縦棒記号 (|) で表して

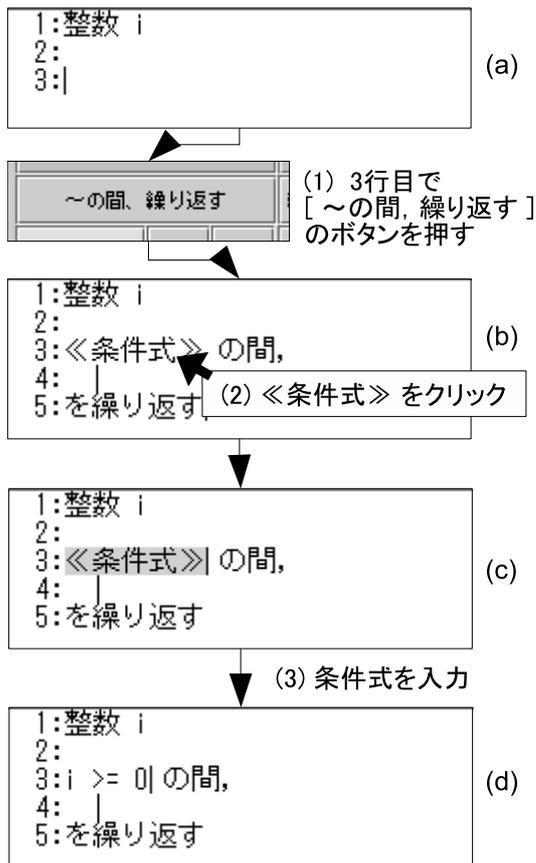


図 3 入力支援ボタンによるプログラミング例

いる。PEN では、インデントの縦棒記号を自動で挿入するようにした (図 4)。

図 4 (a) の状態で、[もし~そうでなければ] の入力支援ボタンを押すと、インデントを付加した制御構造のテンプレートが図 4 (b) のように挿入される。インデントを自動的に付加することにより、プログラムの記述を助けるだけでなく、プログラムの構造を明確に意識する手助けにもなる。

3.2 プログラムの実行状態表示機能

プログラムの実行は図 1 上部中央にある実行制御ボタン群の実行ボタンにより行う。また、プログラム実行の流れを理解できるよう、1 行ずつ実行できるステップ実行や、実行速度を調節し実行できるスロー実行の機能を持つ。

実行時には、どの行が実行されているかを実行箇所マーカーで常に明示している。変数表示画面では、実行中のプログラムで用いているすべての変数の値を常に表示している。これらの情報を提示することで、プログラムがどのように実行されているかなどの状況を把握しやすくしている。また、実行状態表示ラベルで

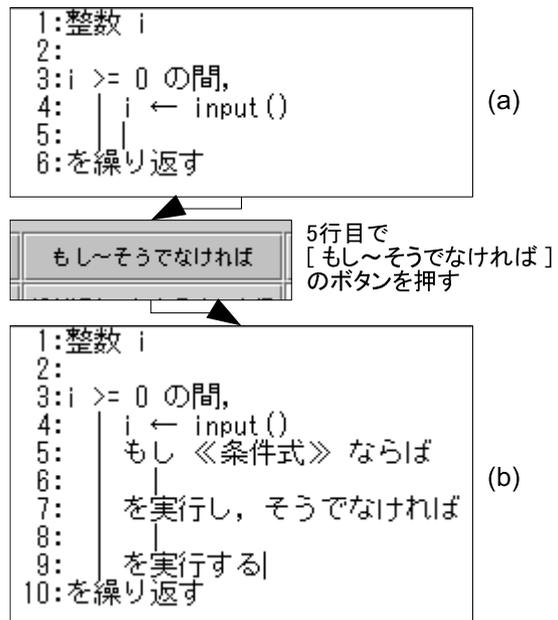


図 4 インデントの自動挿入

プログラムの実行時の状態を明示している。

3.2.1 実行箇所マーカー

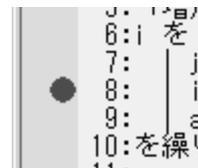


図 5 実行箇所マーカー

図 5 にある「●」は、次に実行するプログラムの行を指し示した実行箇所マーカーである。また、スロー実行やステップ実行と組み合わせて用いると、トレース作業を容易に行える。

3.2.2 実行状態表示ラベル

図 1 の右上に実行状態表示ラベルがあり、プログラムの実行状態を把握できるようにしている。プログラムの状態としては、表 1 に示すように、5 つの状態がある。プログラムの実行状態を表示することにより、プログラムが途中で停止しているのか動いているのか、入力を求められて停止しているかなどが、一目でわかる。

3.2.3 スロー実行

プログラムを実行しながら観察できるよう、実行速度を調節できる機能を実装した。実行速度の調節は図 6 の実行速度調節バーにて行う。実行速度調節バーの

表 1 実行状態の一覧

状態表	意味
実行待ち	プログラムを実行していない状態。プログラムの編集が可能。
実行中	通常にプログラムを実行している状態。
一時停止中	「一時停止」や「一行実行」によって実行が停止されている状態。
入力待ち	プログラム内の入力文による入力待ち状態。入力はコンソール画面に行う。
実行終了	プログラムの実行が終了した状態。再度、プログラムを実行する場合は「始めから実行」で行う。また状態を「実行待ち」に初期化するには「始めに戻る」で行う。

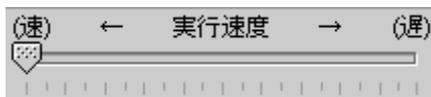


図 6 実行速度調節バー

ツマミをドラッグ&ドロップし、右へ動かすと実行速度が遅くなり、逆に左に動かすと速くなる。実行速度の調節はプログラムの実行中でも変更できるので、注目してみたい箇所で行実行速度を遅くし、じっくりと観察するといった使い方も可能である。

3.2.4 ステップ実行

プログラムの実行過程を詳しく観察できるよう、ステップ実行を用意している。ステップ実行は、図 1 上部中央にある実行制御ボタン群の一行実行ボタンで実行できる。一行実行ボタンを押すと、実行箇所マーカーの指し示している行を実行し、次の命令に実行箇所マーカーが移り、プログラム実行状態が一時停止状態になる。

3.2.5 変数表示画面

プログラムの実行過程を観察する際、変数にどのような値が代入されているかなどの情報が必要になる。変数に関する情報は、図 1 の右下にある変数表示画面に表示される。表示される項目は「データ型」、「変数名」、および「変数の値」である。

3.3 プログラム入力支援ボタンのカスタマイズ

プログラム入力支援ボタンは外部ファイルによって定義されており、ボタンの表示名やサイズ、色、押した時に入力される文字列をカスタマイズできるようにしている。カスタマイズすることにより、教員が授業の進行状況に合わせて、必要なボタンのみを表示させておくことも可能である。また、個々の入力支援ボタンには任意の文字列を対応付けておくことができるので、不等号など入力しづらい文字をボタンに割り当てることができる。

ボタンの定義方法は、入力支援ボタン定義ファイルに、1 行につきボタン 1 つを記述する。フォーマットは「ボタン名@ボタンサイズ@ボタン色 (R,G,B)[@挿入文字列]」である。ボタン名とは、ボタンに表示される文字列で、ボタン名は最大 48 文字表示できる。ボタンサイズはボタンの横幅で、例えばボタン名 10 文字分の幅をとりたい場合は「10」と記述する。挿入文字列は、ボタンを押したときに編集画面に入力される文字列である。また、挿入文字列を記述しなかった場合、ボタンではなくラベルとして表示される。図 7 に、初学者向けの入力支援ボタンの定義ファイルと、その入力支援ボタンの表示例を示す。

4. 評価

4.1 操作記録の解析と評価

プログラミング教育の入門用として、PEN の有効性を検証するために、利用者の操作記録を記録再生できる機能をもつ SOLAR-CATS[5] システムを利用した。SOLAR-CATS の作者である鹿児島大学の山之上先生の協力を得て、SOLAR-CATS に PEN を組み込んで頂いた。これによって、PEN の利用者のすべてのマウスおよびキーボード入力を時刻とともに記録することができ、利用者の挙動を把握できるようになった。

今回の実験は、PEN の有効性を検証するための予備実験の位置づけであり、プログラミング、および、キーボード入力に習熟した被験者 3 人に対して、プログラムソースを与えて、入力支援ボタンを使った場合と、使わなかった場合で、プログラムの入力時間、打鍵数などを比較した。

表 2 解析結果の平均値

プログラム	Prog1	Prog1	Prog2	Prog2
支援ボタンの使用の有無	無し	有り	無し	有り
入力間違い箇所	2.67	1.33	0.33	0.33
全入力時間 (s)	285	302	264	293
マウス操作時間 (s)	3	55	0	33
キー操作時間 (s)	216	120	200	148
キー入力総数	1,004	709	964	549
カーソル打鍵数	20	207	7	145
消去打鍵数	58	72	52	43
モディファイア打鍵数	38	26	39	26
[半角/全角] の切替数	55	26	49	19
漢字変換中の打鍵数	791	446	720	232
入力支援ボタン押回数	0	21	0	21

消去打鍵数：BackSpace や Delete キーの打鍵数

モディファイ打鍵数：Shift, Ctrl, Alt の打鍵数

漢字キー打鍵数：半角/全角切り替えキーの打鍵数

支援ボタン定義ファイル：

```
初学者向け入力支援ボタン@48@204, 204, 204
もし～そうでなければ@12@140, 220, 200@もし <<条件式>> ならば<br>
| <br>を実行し, そうでなければ<br> | <br>を実行する
～の間, 繰り返す@12@180, 220, 200@<<条件式>> の間, <br> | <br>を繰り返す
入力@6@255, 255, 204@<<変数>> ← input()
出力@6@255, 255, 204@<<出力文>> を印刷する
代入@6@255, 255, 204@<<変数>> ← <<式>>
整数@6@255, 204, 255@整数 <<変数>>
```

表示例：

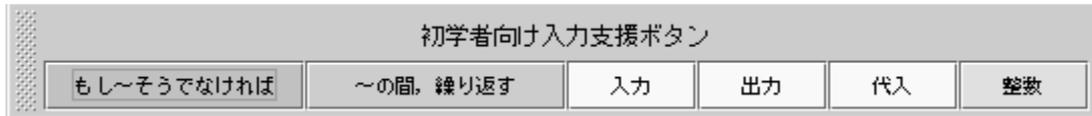


図 7 プログラム入力支援ボタンのカスタマイズ例

評価で使用したプログラムは、図 2 のプログラム (Prog1 と呼ぶ) と、図 2 と同じステップ数の別のプログラム (Prog2) の 2 種類である。この 2 種類のプログラムを、入力支援ボタンを使用した場合と、使用しなかった場合で、被験者に入力してもらい、その操作を記録した。公平をきすため、入力支援ボタンを使用した場合と使用しなかった場合について、Prog1 と Prog2 で実験の順序を入れ替えて、以下の順で実験を行った。(1)Prog1-ボタン使用、(2)Prog1-ボタン使用せず、(3)Prog2-ボタン使用せず、(4)Prog2-ボタン使用。

表 2 は実験時の操作記録を解析したものである。被験者全員がキーボードの入力に習熟していることもあり、入力支援ボタンを使用した時より使用しない方が、プログラムの記述を早く行える結果となった。しかし、キー入力総数は、入力支援ボタンを使うことにより、Prog1 で 29.38%、Prog2 で 43.05% 減少した。

入力支援ボタンの利用によって、利用者のキー入力数を大幅に減らすことができることが確認できたことから、初学者がプログラムを入力する際にタイプミスによる文法エラーの発生を抑えることができる可能性があることを示唆している。

4.2 授業での利用

4.2.1 大阪学院大学での使用例

大阪学院大学情報学部において、PEN を 2005 年度前期 1 年生向けのプログラミング入門の講義 (座学) および、演習 (全 13 回, 90 分/回) で利用した。PEN は 5 月の各 4 回の講義、演習で利用し、C 言語を学習する前に、プログラミングの基本として順次、選択、反復の各制御構造と配列の考え方を理解してもらうことを目的として授業を行った。教材は、22 の例題と

それに対応して各 1 問の演習問題を提供し、講義では例題の解説と演習問題を考えるためのヒントとなるような説明などを行った。演習ではそれを受けて PEN を実際に用いた例題の入力、検証と演習問題のプログラミングを行った。

講義最終日に学生から意見を得るため、アンケートによる調査を行った。アンケートには、「理解できる場所は簡単に理解できました。」「初めて使いましたがなかなか面白いものだと思います。」「一部わからなかった部分もありましたが、その次の授業で大体は分かりました。」など、プログラミングというものを理解してもらえているような感想があった。また、「日本語環境だったので、C 言語よりわかりやすかった。」「日本語なのでわかりやすかった。」と、言語が理解しやすかったという感想もあった。しかし、一部の学生からは「C 言語をメインにやると思っていたため、つまらなかった」という意見があった。

4.2.2 大阪市立大学での使用例

大阪市立大学の 2 部の共通教育科目「情報処理 I」(2005 年度前期) の中で、PEN を試用した。この授業は、いわゆるコンピュータリテラシーの習得を目的とするものであり、全 13 回 (180 分/回) の授業のうち、最後の 3 回 (ただし、最後の 1 回は自習) の授業として実施した。

全授業終了後のアンケート調査として「この授業で何を学んだか。もっとも興味深かったものは何か」という質問をおこなった。そこでの回答の一部を以下に紹介する。

昨年度までは、この 3 回で、定められたルールに従って論理的に正しく記述する作業を体験することを目的として、LaTeX の使い方について演習していた。

- 一番興味深かったのは、やはりプログラミング演習である。プログラミングをすることで、自分の都合のいいように問題を処理することができる楽しさを学んだ。
- パソコンについて無知だったけれど、少しはわかるようになりました。特に、プログラミングが全く出来なかったのに先週メールで提出したプログラムの課題や、今日のテストのプログラムが一人で出来て、エラーもなく実行できたので嬉しかったです。
- 興味深かったものは、プログラミングです。センター試験の数学 IA の問題が解けるのかなと思えてきたからです。
- PEN を使った作業は新鮮で面白かった。PEN は本当におもしろかったと思う。パズルみたいだった。最も興味深かったものとして、プログラミングを挙げたものが、全 34 名中 12 名いたことは特筆に値する。上記の意見の多くは、プログラミングそのものに対する興味を示すものであり、PEN を使ったことが、直接的に影響しているとは必ずしも言えないが、プログラミングに興味を引き付ける役割りの一部を果たしているものと推測している。

5. おわりに

プログラム入力支援機能やプログラムの実行状況の表示機能などを備えた入門教育用プログラミング環境 PEN を作成した。SOLAR-CATS に組込んだ PEN を使用し、利用者の操作履歴を収集・解析し、PEN の有効性についての評価実験を行った。今回の被験者は、いずれもプログラミング、および、キーボード入力に習熟しており、実際のプログラミング教育においての有効性を検証するまでには至っていない。今後、プログラミング初学者に対して、同様の実験を行い、PEN の有効性を検証していく予定である。また、実際のプログラミング教育での使用経験を積み重ねて、PEN の改良に努めたい。

謝辞 SOLAR-CATS への PEN の組み込みに御協力頂いた鹿児島大学総合情報処理センターの山之上卓教授に感謝する。

参 考 文 献

- 1) 西田 知博, 中村 亮太, 松浦 敏雄: “入力支援機能を備えた初学者用プログラミング環境 PEN”, IPSJ SIGCE SSS2005, pp.155-156 (2005-08).
- 2) 兼宗進, 中谷多哉子, 御手洗理英, 福井真吾, 久野靖: “初中等教育におけるオブジェクト指向プロ

- グラミングの実践と評価”, 情報処理学会論文誌, Vol.44, No.SIG13, pp58-71 (2003).
- 3) 長慎也, 甲斐宗徳, 川合晶, 日野孝昭, 前島真一, 箕捷彦: “Nigari - Java 言語へも移行しやすい初心者向けプログラミング言語”, 情報処理学会研究報告「コンピュータと教育」, No.071, pp.13-20 (2003).
 - 4) “一般情報処理教育の実態に関する調査研究”, (文部省委嘱調査研究) 情報処理学会, 一般情報処理教育の実態に関する調査研究委員会 (1992-03).
 - 5) 原田 悦子: “文科系大学・学部における情報教育～その目的と問題～”, 情報処理, Vol.41, No.3, pp.227-233 (2000-03).
 - 6) “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”, (文部科学省委嘱調査研究) 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委員会 (2002-03).
 - 7) 大学入試センター: “センター試験用手順記述標準言語 —DNCL—”, 平成 15 年度センター試験試験問題評価委員会報告書, pp.258-259 (2003).
 - 8) 中森真理雄, 中條拓伯, 小谷善行, 辰己丈夫, 金子敬一, 並木美太郎, 品野勇治: “平成 18 年度入試に向けての「情報」試行試験の実施報告 (2)”, 情報処理学会第 46 回プログラミングシンポジウム報告集, pp.173-180 (2005-01).
 - 9) 山之上 卓: “多数の端末上のアプリケーション操作の記録再生を行う教育支援システム”, IPSJ SIGCE SSS2005, pp.61-68 (2005-08).