# プログラミング教育環境の構造に関する一考察

*A Study for Programming Education Environment*

桑原 悟*

適用範囲が広く，社会に良質の教材を多く提供できるプログラミング教育環境を構築するため，プログラムの内部表現での保持と既存の言語への変換という概念を導入した．

これにより，初等教育における概念教育，中等教育及び非情報系の学生向けの導入教育に加え，C言語，Javaなど既存実用言語へのアプローチ教育が同一環境において可能な設計を実現した．

また，各教育者がこの環境上で設定したテーマセットは，移植が可能であるので，多様な教材の共有や改良を行うコミュニティが形成できる．さらに，入力支援機能により，初心者にとって障壁となる，キーボード入力及び綴り誤りに纏わる負担を軽減し，論理的な思考に傾注できる環境を目指した．

Introducing an inner expression system and conversion to a source of an existing programming language, the followings will be done with the same environment. (i) Education for general idea of programming, (ii) Programming introduction education, (iii) Approach education to existing programming languages such as a c-language and JAVA.

In addition, programming teachers can set their own materials on this environment.

Furthermore, in this environment, by the effective mouse choice input, programming barrier such as a keyboard input work load and a spelling error will be reduced to devote to a logical thought in a programming for beginners particularly.

## 1. 背景

情報処理学会の日本の情報教育・情報処理教育に関する提言2005[1]では，国民全体に対して何らかのプログラミング教育が必要であるとしている．

プログラミング教育環境に関する研究としては，初等教育でのオブジェクト指向プログラミング概念の習得の成果に関しての報告[2]がある．

大学での初心者向けプログラミング教育用の言語に関する報告としては，「若葉」[3]があり，初心者には不要な機能の排除，1種類だけの制御処理記述，単一の数値データ型及び，初心者には意味を説明し難い必須記述部分の排除という観点から，新しい言語仕様の導入と処理系の設計及び実装を行っている．

また，文系学生向けのプログラミング教育に関する報告[4][5]も行われている．

Nigari[6]は，学習者が将来Javaへ移行することを意識し，初心者には「おまじない」と説明するような部分や利用に先立ち書かなければならない変数の宣言文が不要であるなど，初心者向けのプログラミング言語として設計されている．

プログラミングの難しさを軽減するものの報告としては，PEN[7]がある．PENでは，プログラミング言語として，特別な説明なしに入試にも用いられているという理由から,DNCL[8]及びTUATLE[9]を採用し,日本語でのプログラミングを志向している．PENのプログラミング環境では，

*KUWAHARA, Satoru［情報システム学科］

ソースプログラム入力フィールドの下にプログラム入力支援ボタンを設け，条件分岐，繰返しなどの定型部分を自動生成し，キーボード操作を減らす仕組みをもっている．

## 2. 狙い

このように，プログラミング教育に関しては，さまざまな研究がなされ，その成果も報告されているが，それぞれ個別の環境であり，たとえば，初等教育においてグラフィカルなオブジェクトを扱うプログラムで興味を刺激し，慣れさせることでプログラミングの概念教育を行っても，その後の実用言語への移行教育を同じ環境で行うことのできる例は報告されていない．

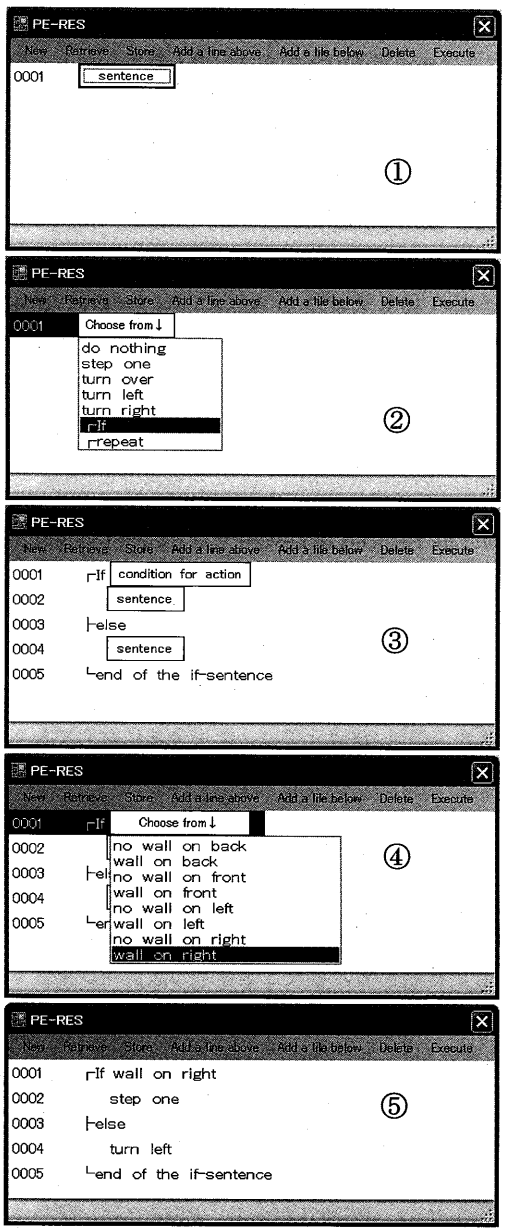


図 1　入力支援機能の動作例

Figure 1　Action sample of Input Support Function

また，複数の実用言語の教育を行う場合は，それぞれ別の環境を渡り歩くことになるので，そのそれぞれの環境は優れた面をもっていても，プログラミング環境の変化という負担を学習者に強いることになる．

さらに，初心者向けに提案されているプログラミング言語も最初から完全無欠のものを創出することは難しく，変更や改良案は出るが，そのためのシステム変更の費用や負荷は相当なものとなり，また，短時間でこれを実現することは難しいといわざるを得ない．

加えて，プログラミング教育の構想や初心者向けプログラミング言語の新しい構想をもつ教育者が，必ずしも実行環境を自ら作成できるわけではないので，良質の教材を豊富に社会に提供できる状況とはなっていない．

本研究では，興味を刺激するグラフィカルな教育言語も，実用言語への将来の移行を意識した教育用言語も，同じプログラミング学習環境で実現すること及び，教育者が考案した教育用プログラミング言語を専用の環境をそれぞれに構築しなくてもよいように，比較的簡単に新しい教育用言語を実装できる構造を実現するための設計について述べる．

また，特に初心者のため，より積極的にキーボードの使用を減らした入力支援機能 [10] を実現する．

## 3. 主要な仕様と設計

### 3.1 入力支援機能

この環境では，特に初心者にとって障壁となる

プログラミング時のキーボード入力作業負荷及び綴り誤りから起こるエラーに関し，マウス選択入力を充実されることでこれを排除し，プログラミングにおいて学習者が，論理的な思考に傾注できる環境を目指して設計した．

ここでは，ウィンドウに描画された迷路からコマを動かして脱出させるという問題解決型課題のための教育用言語をこの環境に実装した例を用いて説明する．

図１①の画面は，入力の画面の初期状態である．表示されている文（画面では“sentence”）を入力するためのフィールドを左クリックすることで，そこに入力できる選択肢が現れる（図１②参照）．そこで，if 文を左クリックすると，図１③の画面に示すように，if 文の構成が自動的に入力される．

if 文の構成中には，判定式（画面では“condition for action”）と二つの文を記述すべきフィールドが表示され，判定式のフィールドを左クリックすると判定式としてこの言語で記述できる選択肢が表示される（図１④参照）．

この中から，“wall on right” を選び，さらに if 文中の二つの文を同様にマウス操作で選択していき，図１⑤で，if 文が完成する。このように，プログラム作成者は，キーボードによる文字入力操作をせずに，画面表示から情報を得て，プログラムを作成していくことができる．

### 3.2 外部翻訳・実行系へのリンク

学習者は，環境に設定された教育用プログラミング言語でプログラムを作成するが，この環境は実行機能をもたず，学習者が作成したプログラムの実行は，自動的にそのソースコードを既存のプログラミング言語のソースコードに変換し，外部のその言語の既存の翻訳・実行系へリンクすることで実現する．

このようにすることで，既存実用言語へのアプローチ教育にその言語のサブセット又は若干の構文変更を行った言語を用いる場合に無理なく対応でき，今後登場する新しいプログラミング言語にも，その翻訳・実行系が入手できれば容易に対応できる．

さらに，すでに充実したライブラリをもつ既存言語が利用できるので，たとえば，初等教育などにおいて，学習者の興味をひく新たなグラフィカルな教育用言語なども比較意的容易に定義することができる．

### 3.3 内部表現によるプログラムの保持

学習者は前述の入力支援機能を使ってプログラムを作成するが，環境内では，内部表現でこれを保持する．

学習者には意識させずに，入力支援環境が，内部表現を作成し，環境がこれを保持する仕組みである．

この内部表現は，プログラムの各記述要素の前後を，HTML で用いられ記述性が検証されている，開始と終了のタグで挟む形式をとる．これは，階層構造も表現できる．

Ｃ言語と同じ書式の代入文を定義した教育用言語の表示と内部表現は，表１のようになる．

表１　表示と内部表現の例

Table 1 An Outer Expression and The Inner Expression.

| 表示 | 内部表現 |
|---|---|
| na = na / 10 ; | <代入文> <変数> na </変数+1> = <式> <変数> na </変数+1> <除算演算子> / </除算演算子+1> <定数> 10 </定数+1> </式+1> ; </代入文+1> |

この形式では，記述要素の階層構造と名称が表現されているので，前述のプログラミング画面のマウス選択エリアの表示に必要な情報もここに揃っていることになる．

また，タグによって記述要素の開始終了を容易に判定できるので，後述のプログラム変換のアルゴリズムは比較的容易なものとなる．

タグの文字に日本語を用いているのは，表現の自由度を示すためであり，“<“ と “>” の間に名称を書き，終了タグ側は，名称の前に “/” を書くという規則が守られれば，名前の文字列は任意である．

表１の内部表現にもあるように，終了タグ中の名称の後に，“+” に続けて，プログラミングにおけるその記述要素の状態を記述する形式を考案した．この状態は，プログラム記述要素の次の形式上の３状態を表す．

- 未入力 (st = 0)：
  たとえば，図１中段での判定式が入る部分のように，内容入力の前であることを意味する．
- 完成 (st =1)：
  たとえば，図１中段の文は，判定式は未入力 (st=0) であるが，if 文は，完成 (st=1) となる．
- 未完成 (st = 2)：
  すべてマウス選択入力でプログラミングを行えば，状態は，未入力と完成だけで表現できるが，テキストエディタなど，別の環境で作成されたものの入力を許した場合に起こり得る，綴り誤りなどで，その記述要素が完成していないことを意味する．

すべての記述要素で，この状態が st=1 である場合だけ，プログラミング言語変換に進むことができる．

### 3.4 言語の定義とテーマセット

この環境では，専用の教育用プログラミング言語を定義し用いるが，ここでの定義とは，プログラムの内部表現の BNF（Backus-Naur Form）を決定することである．

例として，図２に，ウィンドウに描画された迷路からコマを動かして脱出させるという問題解決型課題のための言語の内部表現 BNF の主要部分を示す．

この例では，英語を用いているが，BNF で日本語での表記を用いた定義を行えば，日本語を用いることができる．

step one は，迷路の中のコマを１マス前進させ，turn right，turn left は，それぞれ，コマの向きをその場で 90 度右及び 90 度左に回転する．wall on right は，コマの右側に壁がある場合に真を返し，の no wall on right は，コマの右側に壁がある場合に真を返す．

中級以上のプログラマであれば，真偽の逆転するこの二つを用意する必要を認識しないが，ここでは，教育用にあえてこの二つを用意している例を示した。また，if 文と repeat 文では，入れ子にすることも当然可能であるが，入れ子構造は特に初心者には分かり難い．入れ子の階層の識別のため，この例では，┌if，├else，└end of the if-sentence，┌repeat，└end of the repetition のように括弧様の記号文字を導入し，字下げ機能と組み合わせて，入れ子構造の目視による認識に工夫をしている。

この種の工夫を文法構成に取り込むか，表示系の機能として独立させるかは，議論の必要な部分であるが，ここでは，この環境の柔軟な機能の利用例として示している。

くり返しの条件についても，任意の回数とするのではなく，課題となる迷路の大きさに合わせるように，２回から最大８回の選択しとして用意した例を示している。

```
文 ::= if   else 文
文 ::= repeat 文
文 ::= step one 文
文 ::= turn right 文
文 ::= turn left 文
文 ::= turn overt 文
文 ::= do nothing 文
文 ::= 文 文
if   else 文 ::= <if else 文> ┌if 実行条件) 文 ├else 文 └end of the if-sentence</if   else 文+ st >
repeat 文 ::= <repeat 文> ┌repeat くり返し条件) 文 └end of the repetition</repeat 文+ st >
step one 文 ::= <step one 文> step one </step one 文+ st >
turn right 文 ::= <turn_right 文> turn_right ; </turn_right 文+ st >
turn left 文 ::= <turn_left 文> turn_left ; </turn_left 文+ st >
実行条件 ::= < 実行条件> wall on right < / 実行条件+1>
実行条件 ::= < 実行条件> wall on   left < / 実行条件+1>
実行条件 ::= < 実行条件> wall on front < / 実行条件+1>
実行条件 ::= < 実行条件> wall on back < / 実行条件+1>
実行条件 ::= < 実行条件> no wall on right < / 実行条件+1>
実行条件 ::= < 実行条件> no wall on left < / 実行条件+1>
実行条件 ::= < 実行条件> no wall on front < / 実行条件+1>
実行条件 ::= < 実行条件> no wall on back < / 実行条件+1>
くり返し条件 ::= < くり返し条件> 2 times < / くり返し条件+1>
くり返し条件 ::= < くり返し条件> 3 times < / くり返し条件+1>
くり返し条件 ::= < くり返し条件> 4 times < / くり返し条件+1>
                                          ・
                                          ・
                                          ・
くり返し条件 ::= < くり返し条件> 8 times < / くり返し条件+1>
くり返し条件 ::= < くり返し条件> until getting goal < / くり返し条件+1>
```

図 2　迷路脱出課題の内部表現 BNF

Figure 2　An Inner Expression BNF of “Maze Escape”.

これらは，教育しようとする側の考えで，課題とそこでの教育目標などに応じて構成する．そして，その構成に必要な作業が比較的容易に，言語定義で構成できる点がこの環境の特徴の一つである．

これらのプログラミング言語構成要素は，予め設定した変換規則に従い，プログラム変換機能で，既存言語の関数名に変換され，予めプログラミングした対応する関数定義，たとえばＣ言語でいうならば，include 及び main 関数関連の記述も追加した形で，既存言語ソースコードとして変換出力される．

この，内部状態の BNF 定義，変換規則，既存言語関数定義の三つを一組として，“テーマセット”と呼ぶ．

テーマセットの作成は，専用システム全体の構築よりもはるかに容易であり，他の利用者の環境へも移植可能であるので，多様なプログラミング教育言語教材の出現と，優良な教材の社会への供給に貢献できる．

### 3.5　プログラム言語変換の変換規則

変換規則は，記述追加情報と記述対応情報に分けられる．前者は，前述の既存言語の関数をどこに追加挿入するかという単純な情報であり，後者は，記述の変換のための情報である．後者はさらに固

```
display ( "その数の 1/3 は " nb:5.1f " です。" ) ;
                ↓
printf ( "その数の 1/3 は %5.1f です。" , nb ) ;
```

図３　display の printf への変換

Figure 3　Translation of 'display' to 'printf'

定情報と変動情報に分けられる．このうち，固定情報は単純であるが，変動情報は教育用プログラミング言語側での記述内容で変わる．

ここでは，プログラム変換の柔軟性を示すため，C言語の printf と同じ機能を果たし，初心者に分かりやすい構文をもつ display を設定した図３に示す簡単なプログラムを例に，記述対応情報を用いた変換について述べる．

display の主要な内部表現 BNF を図４に示す．

***display* 文**::= <display 文> ***display*** ( <表示内容> ***表示内容*** </ 表示内容+st>) </display 文+ st >
***表示内容*** ::=< 表示要素 > ***表示要素*** </ 表示要素+ st >
***表示内容*** ::=< 表示要素 > ***表示要素*** </ 表示要素+ st ><表示内容>***表示内容*** </ 表示内容+st>
***表示要素*** ::=" ***文字列*** "
***表示要素*** ::=<書式付き変数> ***書式付き変数*** </ 書式付き変数+ st >
***書式付き変数*** ::=<変数> ***変数*** </ 変数>:<書式> ***書式*** </ 書式>
***変数*** ::= （C 言語に準じるので，ここでは省略）
***書式*** ::= （C 言語に準じるので，ここでは省略）

図４　display に必要な主要な内部表現 BNF

Figure 4　An Inner Expression BNF for . 'display'

表２ printf のための記述対応情報

Table 2 Corresponding for 'printf'

| 内容 | 種別 |
|---|---|
| printf | 固定情報 |
| ( | 固定情報 |
| " | 固定情報 |
| ***文字列*** 及び ***書式*** を出現順に連結する. | 変動情報 |
| " | 固定情報 |
| "," の後に***変数名*** を連結したものを，***変数名*** の出現順に連結する. | 変動情報 |
| ) | 固定情報 |
| ; | 固定情報 |

また，printf のための記述対応情報を，表２のように定義する．

ここでは，必ずしもC言語の printf の完全な構文を記述する必要はない．設定する教育用言語は，通常，既存実用言語のサブセットであると考えられるので，ここで導入した display を定義している内部表現 BNF の範囲で対応できる表現であればよい．

表２中の固定情報は，そのままの対応する文字列を用いればよい．変動情報については，内容の欄に示す文字列操作で得ることになる．このときの，記述要素の切り出しには，図４で示した内部表現 BNF の情報を用いる．

この BNF は，タグを含んでいるので，目的とする記述要素の切り出しにも有効であることが分かる．

## 4.　まとめと今後の展開

本論文では，プログラミング教育の環境について，適用範囲の広さと教材作成の容易さを実現するためのシステム構造について述べた．

適用範囲の広さとして，キーボードからの文字入力をせずにプログラミングできるので，初等教育や，キーボードに馴染めない学習者を対象とすることができることを示した。また，この機能は，選択肢からの選択を行っていくことで，プログラミングを行うので，コンパイラ又はインタプリタの難解なエラーメッセージで学習者の意欲を削ぐ危険を回避できる機能を示した。

さらに，文法定義を分離したので，教育しようとする側が，目的に合わせて言語仕様を設定できる機能を示した．このことは，既存の実用言語の環境での初心者教育での避けがたい難点を克服でき，また，学習者がプログラミング学習のどの局面で，あるいは，文法上のどの構成要素でつまずいているのかを探るための試験環境を構築できることを意味している．

今後は，ここで報告した仕様と設計に関する独自用語の整理とシステム実装を進め，適用の第一段階として，テレビゲームの例でグラフィックイメージに興味をもち，マウス操作には問題がないと考えられる初等教育の対象者にたいして，動機付けと課題解決による達成感を背景としたプログラミング教育への適用シナリオを構成する。

また，情報系の学生向けの言語定義などの教育への適用拡大についても考察し，さらに，キーボード操作なしにプログラミングを行う機能が，バリアフリーのプログラミング環境へ展開できるかの検討も進める計画である．

## 参考文献


1) 情報処理学会情報処理教育委員会：日本の情報教育・情報処理教育に関する提言 2005, 情報処理学会 (2005).
2) 兼宗進，中谷多哉子，御手洗理英，福井眞吾，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 ( トランザクション )，Vol.44, No.SIG13, pp.58-71(2003).
3) 吉良智樹，並木美太郎，岩崎英哉：初心者入門用言語「若葉」の言語仕様と処理系の実装，情報処理学会論文誌 ( トランザクション ),Vol.40,No.SIG10, pp.28-38(1999).
4) 河村一樹，斐品正照，徳岡健一：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，情報処理学会コンピュータと教育研究会，Vol.2001,No.122, pp.9-16(2001).
5) 佐野洋：大学の文科系学部におけるプログラミング教育の試み，情報処理学会コンピュータと教育研究会，Vol.1998 No.102, pp.41-48(1998).
6) 長慎也，甲斐宗徳，川合晶，日野孝昭，前島真一，筧捷彦：プログラミング環境 Nigari －初学者が Java を習うまでの案内役，情報処理学会論文誌 ( トランザクション ) ,Vol.45,No.SIG09,pp.25-46(2004).
7) 中村亮太，西田知博，松浦敏雄：プログラミング入門教育用学習環境 PEN, 情報処理学会コンピュータと教育研究会，Vol.2005,No.104, pp.65-71(2005).
8) 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).
9) 中森眞理夫，中條拓伯，小谷善行，辰巳丈夫，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).
10) 桑原悟：初心者教育用プログラミング環境に関する一考察，FIT2007(2007)