A-020

実行可視化機能を備えた図形ベースプログラミング学習支援ツール

A Graphical Programming-Learning Assist Tool with Execution-Visualizing Feature

平尾 太加士† 野口 健一郎†
Takashi Hirao Kenichiro Noguchi

1. はじめに

プログラミングに必要な能力にはアルゴリズム作成能力とソースコード作成能力がある．特に，プログラミング学習の序盤においては，アルゴリズム作成能力の訓練が重要になる．

本研究では，図形の操作によりプログラムを作成することができ，またプログラム実行の可視化機能を備えたプログラミング学習支援ツールの試作を行った．プログラムの基本構造である連接，選択，反復用の図形要素をマウスで操作することで，プログラムの作成および編集が容易にできる．これにより，学習者はアルゴリズムの作成に専念でき，またプログラムの構造が理解しやすくなる．実行の可視化では，ステップ実行や巻き戻し実行も可能にした．

2. ツールの概要

ユーザはツールを用いて，画面上で図形を操作することによりプログラム図の組み立てと編集ができる．また，作成したプログラム図をいくつかのモードで実行することができる．ファイルからサンプルプログラムを読み込むこともできる．ツールの主要機能を図 1 に示す．

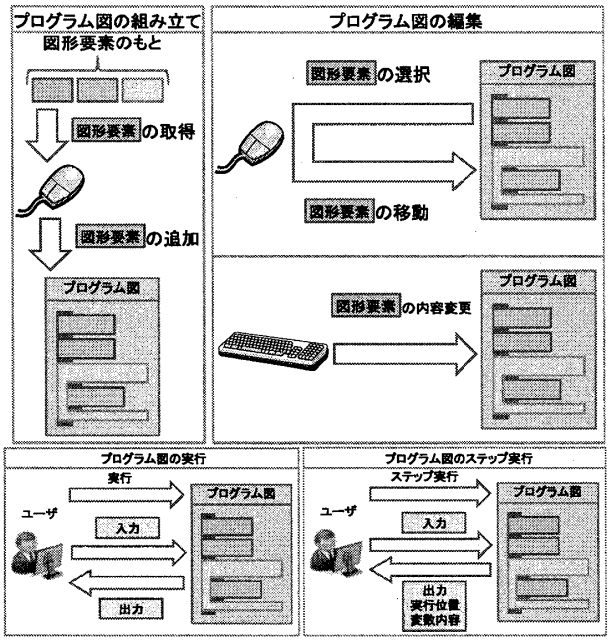

図 1 ツールの概要

プログラム図の形式を定める際には，PAD (Problem Analysis Diagram) [1]の記法を参考にした．

†神奈川大学理学部情報科学科

3. 図形操作によるプログラミング

プログラム図は基本図形要素，連接要素，および複合図形要素で構成される．

(1) 基本図形要素

次の 3 つがある．図の例は図 3 参照．

①入力要素：変数へ入力を行う．実行時に，入力ダイアログへ文字列を入力する．文字列が実数を表すなら実数値，“true”なら真値，“false”なら偽値，それ以外なら文字列値として変数へ代入される．

②代入要素：変数へ値を代入する．

③出力要素：値をコンソールへ出力する．

(2) 連接要素

ユーザは基本図形要素と複合図形要素を連接要素（コネクタ，図 2）の位置へ追加することでプログラム図を組み立てる．図 3 は図 2 のコネクタへ入力要素，代入要素，および出力要素を追加した状態を表している．コネクタへ要素を追加すると，追加した要素の上下に新たなコネクタが自動的に生成され，そこへ新たに要素を追加していくことができる．図 3 の例は，変数 x へ入力ダイアログから値が入力され，変数 y へ 1+1 の評価値が代入され，そして文字列“1+1=2”が出力される．

図 2 連接要素
(コネクタ)

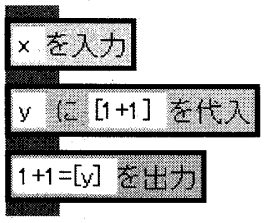

図 3 連接への要素の追加例

(3) 複合図形要素

複合図形要素には選択要素と繰返し要素がある．これらは自身の中に連接を含んでおり，プログラム図を入れ子構造で組み立てることができる．

①選択要素：指定された条件を判定し，入れ子の要素を選択実行する．図 4 の例では変数 x に符号をつけて出力している．

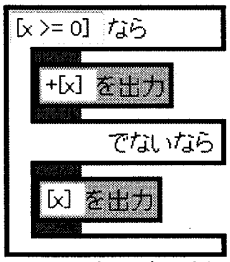

図 4 選択要素の例

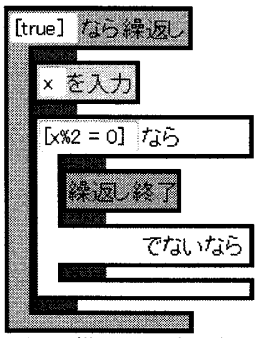

図 5 繰返し要素の例

②繰返し要素：指定された条件を判定し，その値が真なら入れ子の要素を実行し，偽なら実行せずに次へ進む．値が真で入れ子の要素を実行し終えたときには再び条件の判定を行う．図 5 に例を示す．
③繰返し終了要素：この要素を入れ子に含んでいる親要素で最も近い繰返し要素の繰返し実行を終了する．図 5 の繰返し要素では変数 x に 2 の倍数が入力されたら繰返しを終了する．
(4) 変数と値
ユーザは 1 つの変数に対して 1 つの値を格納することができる．明示的な変数宣言は存在せず，ある変数に対して入力または代入がされた時点で宣言される．
値は文字列で指定するが，式の評価値を表す場合は式を[ ]でくくる．また式に変数を含むことができる．変数 x に値 1 が格納されている場合，[x+2]は値 3 と同じ意味になる．また，式として数式，比較式，論理式を扱うことができる．
入力，代入，出力，選択，および繰返し要素の図形の白い部分は，ユーザが内容を設定する部分であり，連接要素への追加直後には空になっている．ユーザは変数における値の書式に従って内容を設定する．
(5) 例
図 6 にユークリッドの互除法を行うプログラム図を示す．

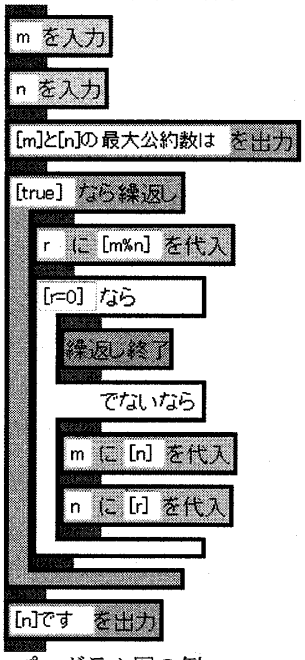

図 6　プログラム図の例

4. プログラム実行の可視化

ユーザが作成したプログラム図を次の 3 種類の方法で実行することができるようにした．
(1) 実行
プログラム図の先頭から末尾まで自動で実行する．これが本来のプログラム図の動作となる．
(2) ステップ実行
プログラム図を指定した N ステップずつ実行できる．N ステップ実行後の一時停止時に，そのときの実行位置と使用変数の内容を確認することができる．
(3) 巻き戻し実行
実行を進めるステップ N に負の数を指定することで，ステップ実行を巻き戻せるようにした．つまり，100 ステップ進めて問題が発生したら 50 ステップ戻ってみるということが可能になる．

5. 実装

実装には HTML，CSS，JavaScript を使用した．
(1) ツールの実行環境
本ツールは Web サーバに置き，ブラウザ上で利用できるようにした．プログラム図の編集は，HTML 文書内の要素のスタイルを JavaScript で変化させた．入力時に入力ダイアログがポップアップする．出力のためのコンソールはページ内に表示される．
(2) プログラム図の実行制御
(a) 連想配列を用いて変数操作を実現した．連想配列とは，配列の添え字に文字列を指定することができるオブジェクトである．ユーザの使用する変数は連想配列 variableList に記録されている．例えば変数 x の値は内部では variableList["x"]に記録されている．
(b) eval 関数を用いて式の評価を実現した．eval 関数とは，JavaScript に用意されている関数で，引数に与えた JavaScript 文を評価し，その最後の評価値を返す関数である．例えば a[x+1]は内部では"a"+(variableList["x"]+1)+ ""と置き換えられ評価される．
(c) クロージャを用いてプログラム図実行を制御するようにした．クロージャとは，変数を実行時の環境でなく自身が定義された環境において解決する関数で，関数内の変数を保持することができる．今回は，基本図形要素，連接要素，複合図形要素を実行するクロージャを生成する関数をそれぞれ定義した．生成されるクロージャには，ユーザが設定した内容，要素の状態，および巻き戻しに必要なログを保持させた．連接要素や複合図形要素は中に他の要素を含んでいるため，それらのクロージャも同様に，入れ子になっている要素のクロージャを持ち，実行時にそれらを呼び出している．また，連想配列に各クロージャ生成関数を格納することで，図形要素ごとのクロージャ生成がスムーズに行えるようにした．
(3) プログラム図の読み込み
ファイルにあるサンプルプログラムは XML 形式で記述するようにした．

6. おわりに

本研究は，情報関係の学生がプログラミングの学習を行う際に利用できる支援ツールの実現を意図した．子供を含めた広い層を対象とした図形ベースのプログラミングシステムである Scratch [2]とは，目的に違いがある．
学習支援ツールとして，作成したプログラム図のアルゴリズムを視覚的に見ることができるが，プログラム図で表すアルゴリズムを作成する支援機能などが不足しており，今後の課題である．

参考文献

[1] 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.
[2] M. Resnick, et al., Scratch: Programming for All, *Commun. ACM* 52, 11 (Nov. 2009), 60-67.