情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-CE-107 No.12
2010/11/21

踊りによる プログラム処理の表現方法

山石忠弘† 林敏浩† 垂水浩幸†

プログラミング教育において，座学でプログラムの知識を身に付けた学習者が，ソースプログラムを正しく読み取れないことがある．原因として，プログラムが動く仕組み理解できていないので，プログラム処理がわからないと考える．本研究では，プログラム処理がわからない学習者に対して，踊りによるプログラム処理の可視化を行うことで，プログラム処理の理解支援を行う．

A Representation of the Process of Programs by Dance

Tadahiro Yamaishi† Toshihiro Hayashi† Hiroyuki Tarumi†

Many learners who have the programming knowledge by classroom lecture, but they cannot read source programs correctly in some cases. As for the cause, we think students cannot understand the process of programs because they do not have enough knowledge about how programs run. For such students, this research assists to understand the process of programs by visualizing them as dance.

1. はじめに

プログラミング教育において，座学によって，プログラムの命令と意味を理解した学習者は必ずしも,それら命令を含むソースプログラムを正しく読めるわけではない．座学によって，プログラムの知識を持ち，かつ文法を理解しているにもかかわらず，プログラミング教育で与えられるサンプルプログラムが理解できない学習者が存在する．プログラムが理解できなくなる原因の一つとして，プログラムの動作を正しく理解できていないということがこれまでに報告されている[1]．学習者は，身に付けた知識を基にソースプログラムを解読しようと試みるが，プログラムの動く仕組みがしっかりと身についていないため，途中でどのような処理が行われているかわからなくなり，学習意欲の低下につながる．我々は，プログラミングの学習支援として，プログラムの処理（動作）が理解できない学習者に対してソースプログラムからその動作が読み取れるようにすることを目標とする．

本稿では，プログラムの処理を踊りによって表現するシステムとその振り付けについて述べる．試作を行っているシステムは，プログラムソースから処理の過程を踊りによって動的に見せることで処理を表現する．プログラム処理を可視化することで，学習者に対してこれまでに原因とされてきた処理イメージを見せることができると考える．その見せ方として，プログラムの処理を表現するような踊りの振り付けをデザインする．学習者は，踊りによって表現されたプログラム処理のイメージを与えられることで，学んだ知識とプログラムソースを結び付けることができると考える．知識とプログラムソースが踊りによって対応付けされることで，学習者の知識レベルをプログラムソースが読める状態に向上できると考える．

2. 先行研究

プログラム処理を可視化する手法として，これまでに多く先行研究が行われてきた．本章では，これまでに行われてきたプログラム処理に対する支援について，特徴を述べ，プログラム処理を踊りで表現する手法の主張点を述べる．

プログラム動作過程の理解を支援するシステム azur [1]はプログラムのブロック構造の可視化機能を持つ．ブロック構造によって，制御文の影響を与えている範囲を階層的に表現することで，ソースコードを整理し対応付けを行う．可視化されたブロック構造上で，プログラム処理のステップ実行により，処理と変化する変数を見せる．

また，デバッカ機能に重点を置いた統合支援システムが開発されている[2]．この研究では，学習者に対してデバッカやインタプリタの必要性を論じ，初心者でも扱いや

† 香川大学
Kagawa University.

Vol.2010-CE-107 No.12
2010/11/21

すいデバッカを組み込んだ統合支援システムを開発している．編集，実行，デバッグ機能をシステム内に持つことにより，利用しやすい環境を整えている．デバックモードには，トレースモード，スローモード，シングルステップモード，関数呼び出し順序の表示,変数値の一覧機能を備え,プログラムの動作過程を様々な視点で見られる．

アルゴリズムアニメーションシステムの研究では，その適切な利用法の検討や，効果的な見せ方について論じられている[3,4,5]．これらの研究の特徴は，アルゴリズムの行う処理（ソート，探査）のモデルを作成し，ステップごとの処理に対してアニメーションモデルを利用することで，アルゴリズムの構造理解を計っている．

ソースプログラムと流れ図を対応付けることでプログラム処理の流れを理解する支援を試みる研究もある[6]．この研究では，サンプルソースに合わせた流れ図を作成し，処理によって変化する変数の値やその手順を，流れ図とソースプログラムに結果を反映させ，ソースと処理の対応付けを行っている．

これまでの先行研究では，プログラムの処理を見せる方法として，変数の動きをアニメーションで表現,変数が変化していく様子を見せる,モデルを利用した動作理解,流れ図を利用した手法などが試みられている．我々は，変数の変化やフローチャートモデルを利用した処理過程を見せるのではなく，処理の動作イメージを見せる手法として踊りによる表現を試みる．踊りによって処理を表現することで，専門知識を必要とするフローチャートや変化する変数を見せるのではなく，イメージによってプログラム処理を動的に表現する．プログラム処理を，踊りの振り付けに置き換え動きとして見せることで，プログラムの動的な動きを表現できると考える．

3. 踊るシステム

本章では，プログラム処理を表現するシステムの概要と実行方法について述べる．また，実行時に生成される踊りを行うキャラクタとその役割について述べる．

3.1 システムの概要

本研究では，C 言語ソースプログラムを扱い，踊りを生成する．ソースプログラムを読み込むとキャラクタを生成し,プログラムの処理の手順を踊りによって表現する．システムの使用方法を示す．

(1) 実行可能ソースプログラムを読み込む．
(2) ソースプログラムを解析する
(3) ソースプログラムの処理を踊りで表現する

(1)について，学習者または教員が用意した実行可能ソースプログラムを用意する．対象とする実行可能ソースプログラムは，あらかじめコンパイルを行いエラーなく動作が確認されたものである．ソースプログラムを，図 1 のシステムにドラックアンドドロップすることでシステムに読み込ませる．

(2)について，システムは(1)で用意した実行可能ソースを読み込むことで，ソース解析を行い解析結果から踊りを生成する．

(3)について，踊りが生成されると図 1 のシステム下部にキャラクタが現れ踊る．学習者は踊りの振り付けによって表現されるプログラム処理のイメージが見られる．

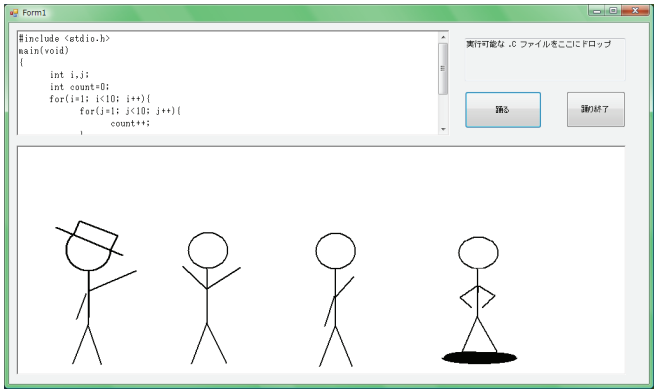

図 1 システムのインタフェース

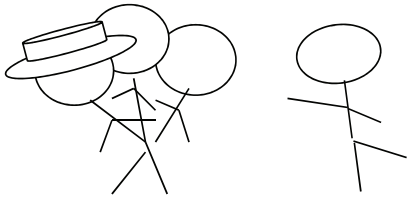

図 2 キャラクタ

3.2 関数キャラクタ

図 2 のキャラクタの中で帽子をかぶっているのが関数キャラクタである．関数キャラクタは次の特徴を持つ．

(1) main 関数およびユーザ定義関数の処理を踊る
(2) キャラクタは関数の数だけ表れ，１体につき１個の関数を表現する

(3) 関数内部で行われている処理がある場合に踊る

(1)について，関数キャラクタは，プログラム処理で扱われる関数の処理を踊りで表現する．このキャラクタの役割は，main 関数とユーザ定義関数で行われる処理を踊りによって表現する．ただし，他の関数（標準ライブラリ，数値演算，文字列操作）については，キャラクタ数の増加による学習者の負荷増加のため表現しない．

(2)について，このキャラクタはプログラムソース内部で扱われている関数の数だけ，キャラクタが増える．関数 1 個につき，キャラクタ 1 体を割り当てる．例えば，図 2 では main 関数しか扱っていないため関数キャラクタは 1 体．図 3 では，main と keisan 関数がプログラムソース中に出てきているため，キャラクタは 2 体となる．

(3)について，関数キャラクタの役割は，現在実行されている関数を表現することである．キャラクタが担当する関数内部で処理が行われている時は，担当のキャラクタが内部で処理されている処理イメージの踊りを踊る．処理の行われていない関数はその場で待機し，出番に備える．例えば，図 3 では，keisan キャラクタが動作を行っているため，keisan が動作を行い，main が待機している状態である．

このように，1 個の関数に対して 1 体のキャラクタを割り当てて表現することで，処理を行っている現在の関数とその処理状態を表現できると考える．

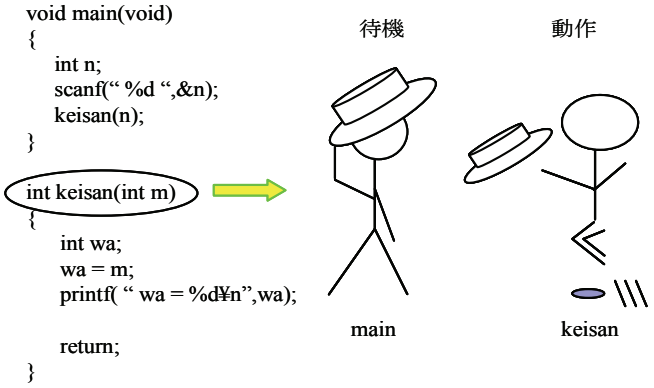

図 3　関数キャラクタの動作

3.3 変数キャラクタ

関数キャラクタと違い，図 2 で帽子をかぶっていないキャラクタが変数キャラクタである．変数キャラクタは次の特徴を持つ．

(1) 変数キャラクタは変数の表現を行う

(2) キャラクタ数は変数の数によって変化する

(3) 処理に使用されていない変数はその場で待機する

(4) 変数の値に変化があるとキャラクタの大きさが変わる

(1)について，変数キャラクタは，変数に関わる処理が実行されると踊りを踊る．これにより変数の状態を表現する．変数の増減や制御文の条件式がこれに該当する．

(2)について，変数キャラクタは関数キャラクタ同様に，1 個の変数に対して 1 体のキャラクタが存在する．図 2 では，変数キャラクタは 3 体存在している．このことから実際のプログラムソース中には変数が 3 つ記述されていることになる．

(3)について，処理を行われているキャラクタは踊りを行うが，処理の中で利用されていない変数・処理を終えた変数は待機し，処理が行われていないことを表現する．

(4)について，演算や代入によって，変数の値が変化した場合，図 4 のように変化の対象となる変数キャラクタの大きさが変化する．例えば，変数の値が減少すればキャラクタの体が小さくなり，増加すればキャラクタが大きくなる．

このように，変数の役割を分けて表現を行うことで，学習者に対してプログラム処理の過程で，変数の動きや変化を見せることができると考える．

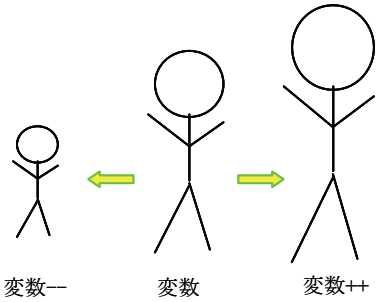

図 4　変数キャラクタの変化

4. プログラム処理の表現方法

本章では，学習者に対してわかりやすく処理のイメージを表現するための振り付け方法と，その見せ方について述べる．処理と踊りの対応を表 1 に示す．

踊りは，学習者にとりわかりやすいように，シンプルな振り付けにする．また，踊りによって表現するのはプログラムの処理であるため，プログラムの動きに近いイメージを持つ振り付けにする必要がある．次節では，学習者にとってプログラム処理がわかりにくいと考える繰り返しや判断に対する処理の振り付けを述べる．

表 1　処理と踊りの対応表

処理	踊り
入力	入力を促すような手拍子を行う
変数の増減	キャラクタの大きさが変わる
演算	関係するキャラクタが肩を組み踊る．出力先キャラクタにタッチ
繰り返し	キャラクタが上半身を回転させ回る
判断	体を斜めに倒す
戻り値	関数キャラクタが帽子を使って戻り値を呼び出す
出力	利用者に向かって，指さし，物を投げる

4.1 繰り返し

繰り返しは，キャラクタが上半身の回転を繰り返すことで表現する．トレースする際の目線の動きや繰り返すイメージから円運動の振り付けとする．繰り返しの踊りと一連の踊りの表現方法について図 5 に例を示す．以下にその詳細を説明する．

(1)　main キャラクタが登場する
(2)　変数宣言により，変数キャラクタ i , a が登場する
(3)　繰り返し処理を実行し，初めに main が回転する
(4)　main の後に続いて終了条件である i が回転を始める
(5)　main , i の繰り返しに同期を取るように a が回転する
(6)　１回転行うごとに, i , a はインクリメントすることで大きさが変わる
(7)　数回繰り返しの動きを数回行い終了する
(8)　キャラクタは処理の踊りをやめる

(1)について，踊りを実行させると，初めに関数キャラクタが登場する．今回使用するソースプログラムで扱うのは main 関数のみのため，main キャラクタが登場する．

(2)について，変数宣言が行われると，それに対応した変数キャラクタが登場する．ここで宣言されているのは，変数 i, a なので，i , a キャラクタが登場する．その後，for 文の処理を踊りで表現する．

(3)について，for の処理になると初めに，main が上半身を回転させ繰り返しをイメージした踊りを踊り始める．

(4)について，main が上半身を回転させ始めると，繰り返しの終了条件に利用されている変数キャラクタ i が関数の後ろで上半身を回転させる．main の後ろに終了条件で使われている i が移動することで繰り返しに i が関わっていることを表現する．

(5)について，繰り返し命令が実行されたことからその内部で，インクリメントされる変数キャラクタ a も動作を開始する．a は，for 文の内部で扱われ繰り返しもされることから，main, i と同期をとるように上半身の回転を始める．

(6)について，a はインクリメントされ変数の値が増加することから，a のキャラクタが大きくなる．また，for の処理が一通り終了すると，i がインクリメントされることで i も大きくなる．

(7)について，数回(3)～(6)の踊りを数回行い繰り返しの踊りを終了する．実行例で行う繰り返しの数が数回となっているのは，アニメーションを用いる先行研究で学習が進むとアニメーションが冗漫に感じたという報告[3]があり，実際の繰り返し回数を省略することで踊りを冗漫に感じさせない狙いがある．

(8)について，全ての処理を行ったことから全ての踊りを終える．なお，return については，値が 0 または，ない場合踊りを踊らない．

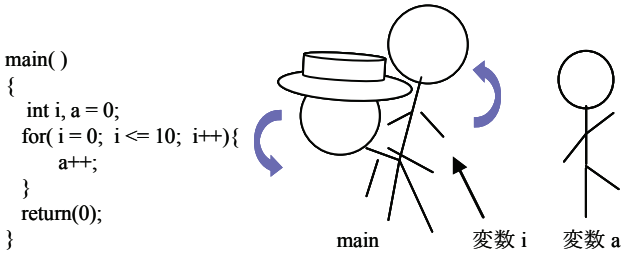

図 5　繰り返しの振り付け

4.2 判断

判断の踊りは，キャラクタが体を斜めに傾け静止する表現する．踊りは，悩んで決断する動作や日常生活のスイッチを参考にしている．判断の振り付けを図 6 に示す．以下に詳細を説明する．

(1)　関数キャラクタが体を斜めに倒す．
(2)　判断に変数が利用されている場合，それに関わる変数キャラクタも体を倒す．
(3)　判断条件にそぐわない場合，キャラクタは体を元の位置へ戻す．
(4)　判断条件を満たせば，一歩踏み出す．

(1)について，if や switch などの判断処理が行われる場合，内部で処理の行われている関数キャラクタが体を斜めに倒し静止する．

(2)について，判断に変数が利用されていれば，利用されている変数キャラクタも関数キャラクタと同じように体を斜めに倒し静止する．

(3)について，判断条件を満たさない場合は，元の位置へ体を戻す．else や case など処理が続く場合は，(3)の状態から，(1),(2)とは逆方向へ体を倒す．それでも満たさない場合は条件を満たすまで，同じ動作を繰り返す． else など追加の判断が存在しない

Vol.2010-CE-107 No.12
2010/11/21

場合，(3)の状態から次の処理へと移行する．

(4)について，条件を満たしている場合，体を倒した状態から 1 歩前へ踏み出すことで，条件を満たしたことを表現する．

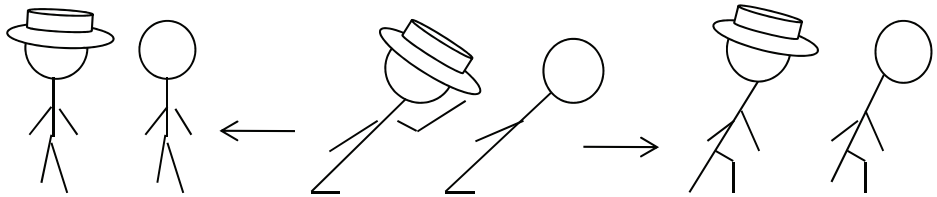

図 6 判断の振り付け

5. おわりに

本研究は，プログラム処理の分からない学習者に対して，ソースプログラムから処理を理解させることが目的である．踊りによるプログラム処理の可視化により支援を行う手法について述べた．踊りによって，プログラム処理を見せることにより，ソースプログラムと座学で学んだ知識を結びつけることができると考える．

今後の課題として，振り付けのデザインについて妥当性の検討を行う必要がある．また，ソースプログラムから踊りを自動生成できるシステムを開発する．

参考文献

[1] 今泉俊幸, 橋浦弘明, 松浦佐江子, 古宮誠一: ブロック構造の可視化によるプログラミング学習支援環境 azur ～関数動作の可視化～, http://is.sayo.se.shibaura-it.ac.jp/SIG-SE-Imaizumi.pdf
[2] 大島邦夫, 古市茂, 岡崎世雄, 平澤京子: C 言語教育用プログラミング支援システムとその統計的評価, 教育システム情報学会誌, Vol.15, No.1, pp.21-28 (1998).
[3] 井上勝行, 古川勝康, 五十嵐丈也, 魚井宏高, 首藤勝: 制御の流れに重点をおいたアルゴリズムアニメーションを実現するシステムの構築, 教育システム情報学会誌, Vol.15, No.2, pp.55-64 (1998).
[4] 井上勝行, 上和田徹, 魚井宏高, 首藤勝: 可視化技術を援用したアルゴリズム自習支援環境の構築, 教育システム情報学会誌, Vol.15, No.2, pp.65-74 (1998).
[5] 税所幹幸, 池田直光, 中村良三: 再帰アルゴリズムを可視化する学習支援システムの設計と開発, 教育システム情報学会誌, Vol.22, No.1, pp.36-41 (2005).
[6] 山本芳人: 流れ図とソースプログラムを対応させたプログラミング学習支援システムの利用, 教育システム情報学会誌, Vol.20, No.4, pp380-384 (2003).