流れの可視化と自動修正提案によるプログラミング支援

市川雄介†

†明治大学大学院理工学研究科

あらまし　プログラミング教育で挫折してしまう学生が多い。そのため、プログラミングに対して苦手意識を持つ学生を支援する研究がされてきた。本研究では対象となる学生を絞り、可視化とエラー修正による支援を目的とした学習支援ツールを作成した。扱うアルゴリズムと用いる変数を固定することで、より詳細な可視化とエラー箇所の特定を行っている。

1. 研究背景

プログラミングに対して苦手意識を持つ学生を支援する研究は様々な視点で行われてきた。例として、プログラム構造の理解を与えるために学生が作成したプログラムに関する内部データの可視化を行う支援ツール、プログラミングする上で必要な発想を促すプロセスの提供などがある。

ユーザが記述したプログラムの内部データを可視化する方法はプログラム理解する与える意味で有効的である。しかし、従来の研究では変数の中身だけを可視化するものが多く、支援となる情報が不十分である。また、より詳細な情報を提供するために重要な部分だけを記述する穴埋め方式による支援ツールが研究されが、プログラムを固定しすぎると、構造的な発想への支援には繋がらない。そこで、プログラミングの自由度を残しつつ、より詳細な可視化とエラー処理を行う支援ツールを作成した。

2. 提案

前章で言った支援ツールを作成するために、本研究ではプログラミングさせるアルゴリズムを固定し、処理に用いる変数を固定した。繰り返し・分岐など具体的な処理方法は自由である。このやり方によって、より詳細な可視化とエラー箇所の特定を行うことができる。また、本研究ではプログラミング教育をトータルで支援することを目的としている。そのため、教育現場で必要な課題提供、課題説明、プログラミング補助、エラー時の補助の過程を全て網羅する支援ツールを作成した。

Programming support by visualizing the flow and automatic amended proposal

Yusuke Ichikawa†

†Department of Science and Technology, Meiji University

3. 本研究の詳細

構造的な発想は、基礎知識を習得してから必要となる。大学で初めてプログラミングする人が多いため、本研究では大学２～３年の学生を対象とする。そのため、扱うアルゴリズムは難易度が高いソーティングアルゴリズムを採用した。現時点でクイックソートとヒープソートによる支援ツールを作成した。本稿ではクイックソートを例にして説明する。

本研究の支援ツールは、学生に C 言語で記述さえる。入出力・main 関数は実装済みである。以下に記述してもらう箇所を示す。

```
quick_sort(int left, int right){
    int  i,j ;
    int  pivot、pivot_value;
}
```

この関数は main 関数から呼ばれる。説明どおり、変数並びに引数を固定している。関数内部の処理部分を記述させる。

4. 支援の流れ

支援手順を示す。

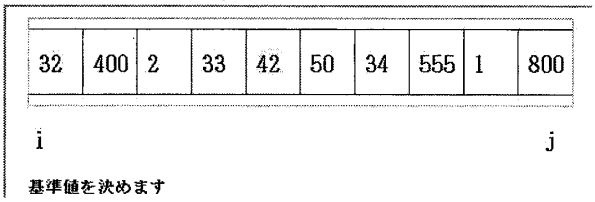
1. 記述してもらうアルゴリズムの説明
   - 一般的なクイックソートの説明
2. 使用してもらう変数とその動きの説明
   - 今回使用する i, j, pivot, pivot_value の説明をする。それを用いてクイックソートの流れを説明する
3. プログラミング
4. コンパイルし、アニメーション・可視化

難易度が高いアルゴリズムであり、変数を固定しているため、本研究に合った詳細な説明を用意した。

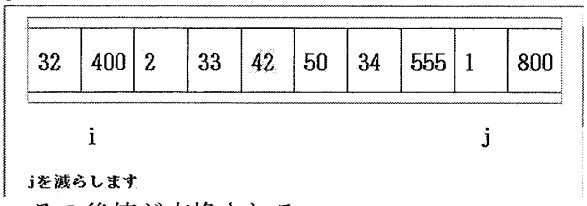
5. アニメーション・可視化について

学生に対して、自分が使用した変数が目的に合った処理を行っているのかという理解を与える目的としてアニメーション化を行った。例として、i と j を中心に注目しアニメーションを説明する。i は左端から、j は右端から始まる。ま

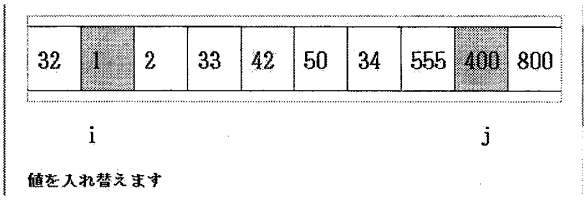
た、基準値（下図で1つだけ色がついている値）が設定される。



上記の初期状態から、iは増加し、jは減少する。そのとき、iは基準値が大きい値で止まり、jは小さい値で止まる。



その後値が交換される。



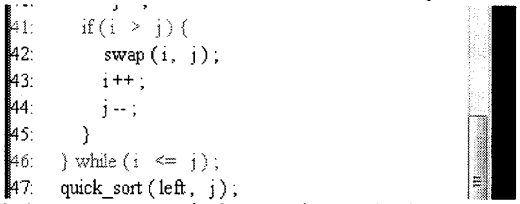
これをiとjが交差するまで行う。以上のようにソーティングの進行を色の変化や動きでアニメーション化を行っている。また、「値を入れ替えます」といった文章をそのつど表示させている。

6. エラー発生時の対応

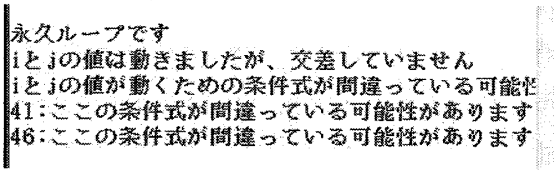
間違った記述をした際、正しい記述への発想を導くことが大切である。特に無限ループのような、コンパイルは通るが、結果的に誤りとなるバグの原因を突き止めることは初心者にとって困難な事である。本研究ではアルゴリズムと変数を固定しているので、これらのようなバグも発見し、特定できる。以下より、エラー発生時の対応の手順を示す。

1. エラー検出
2. エラーが発生した状況でもう一度実行（変数の中身を同じにし、エラーが起きた関数のみを実行）
3. 変数ごとで処理内容を監視して、エラーの原因を調べる
4. エラー箇所を特定し、提示

無限ループを例にあげ、補足する。説明した通り固定的なツールであるが、無限ループになるという断定的な判断をするのは難しい。よって、処理における命令回数の上限を決め、超えるとエラーとしている。無限ループ後、全ての変数の値を調べる。また、システム内で変数の値の関係から、処理終了後どのような値になるかを予想する。その予想と実際の値と異なっている変数を見つける。本来iとjは交差するが、交差せずに処理が終了した場合、iとjの値が動くための条件式が間違っていると判断できる。その変数の値が異なる値になるうる原因箇所の予想を立てる。本研究の支援ツールのブラウザにソースリストが表示してあり、間違っていると想定した箇所の行の色を変えている。



また、エラーの内容と間違いと想定した箇所の説明を行う。



無限ループ以外では、無限再帰・誤ったソーティング・変数へ誤った値の代入などが検出できる。

7. 考察

プログラミング教育の現場で、先生やTAに対して学生からの質問が多々ある。その際、学生に対して1から10まで教えてしまうことが多い。そのため、学生が自分の力でプログラミングを行った事にならない。また、一人一人丁寧に説明していると、全ての学生を見るには時間が掛かってしまう。本研究の支援ツールは、課題提供、課題説明、プログラミング、エラー特定までトータルで支援し、できるだけ学生自身の力で課題進めるようにサポートしている。よって、先生・TAの負担解消、学生に対して自分自身の力でプログラミングすることができる能力の向上が期待できると考えられる。

8. 参考文献

・黒川優介, 三浦悠太, 藤枝崇史, 清水智公, 服部隆志, 萩野達也：内部データの視覚化によるプログラミング支援ツール
第70回平成20年　全国大会講演論文集 1Q-7