教育用プログラミング環境について

4E－1

梶浦　文夫　　木村　宏
岡山理科大学理学部

1　はじめに

現在ではパソコンやワークステーション上の開発言語としてC言語がよく使用されている。また情報処理教育としてC言語を利用した計算機実習が行われるようになった。C言語はPascalやFORTRANに比較して自由な記述が可能である。この結果として、他のプログラミング言語での実習に比べて学生が誤り(バグ)を起こし易く、加えて誤りの訂正(デバッグ)が困難であるという問題が発生している。著者らは前にC言語実習における文法エラーの分析を行いこれらの問題を明らかにした[1][2]。効率的にC言語実習を進めるためには、学習者の理解を高めると同時に初心者向きのプログラミング環境が必要である。

著者らは、初心者が文法エラーのデバッグを行うには「誤りの種類や位置」と同時に「具体的な修正法」を示すようなシステムが必要であると考えている。本研究では、そのような環境を実現するための基礎となる文法エラー自動訂正の実験的試みを中心として、教育用プログラミング環境について検討する。

2　プログラミング環境と対策

著者らが想定しているプログラミング環境は、ソースプログラムの入力および編集のためのエディタを中心とするシステムである。利用者がソースプログラムを1行入力するたびに構文解析用のモジュールが呼び出されて解析を行い、文法エラーに関してはその場で訂正させる。実行方式はインタプリタ方式を想定している。

基本的なエラー対策を図1に示す。発生してからでは自動的なエラー訂正が難しいバグに対しては、バグの発生そのものを減少させる対策が必要である。このタイプの代表的なものは {} の対応誤りである。このタイプのバグに対しては1番目の「入力時の強制」を行う。学習者に {} の対応を常に意識させるように強制的にインデントする機能をエディタに持たせる。また、強制的に1行1ステートメントの形に変形する。

入力時の強制
◎{}による強制インデント
◎1行1ステートメント

入力時のチェック
◎行内に2重以上の()がある時の対応チェック

エラー検出時の訂正
◎プロトタイプ宣言を基準にした引数の型の強制
◎大文字／小文字の間違いチェックと訂正
◎Tokenのキーワード+その他への分割可能性チェックと訂正
◎Tokenの削除／挿入／置換と再コンパイル

図1: 基本的なエラー対策

2番目の「入力時のチェック」は、コンパイラのエラー検出位置から実際のバグ位置を特定することが難しいタイプのバグに対して行う。このチェックは特定のルールを定め、構文解析／意味解析に先だって行う。3番目の「エラー時の訂正」は、エラー検出位置(カレントトークンあるいは直前のトークン)にバグがあるタイプのバグに対して行う。

2番目および3番目のエラー訂正では、訂正の

A Study on Educational Programming Environment
Fumio Kajiura and Hiroshi Kimura
Okayama University of Science
1-1 Ridai-cho, Okayama 700, Japan

誤りが重要な問題となる。従来のエラーメッセージは構文規則などその言語に定められた規則に違反している箇所と内容を知らせるものである。従って、最初に検出したエラーに関する限り、適切とは言えないものの誤ったメッセージは出力しない。一方、訂正を指示するメッセージの場合、具体的で適切である反面、その訂正指示が間違っていた時に初心者に与える混乱は非常に大きい。現在、文法エラーを完全に自動訂正する手法が確立されていない以上、このような副作用をいかに低く抑えるかが非常に重要な問題である。

今回の研究では、変数・関数の未定義・2重定義に関して自動訂正の検討を行っていない。このタイプのバグは、検出は容易でも訂正が難しいことと、従来のエラーメッセージでも問題のトークンそのものを指摘するため比較的容易にデバッグできるからである。以上の対策および自動訂正手法の詳細は講演の際に説明する。

3 考察

バグを含む1386本のソースプログラムに提案した訂正手法を適用し、訂正可能率および誤訂正発生率を調査した。この作業は全て手作業で行った。調査したのは、各ソースプログラムで最初に現れるバグだけである。本研究ではバグを入力時に訂正させるように想定しているため、最初のバグ以降のテキストはその時点では入力されていない。それ以降に現れるバグは以前のバグからの影響を受けている場合があるため調査対象から除外した。従って、調査対象のバグは1386件である。

図2に全体的な訂正可能、訂正不可能および誤訂正の割合を帯グラフで示す。上側のグラフは総件数(1386件)に占める自動訂正の対象範囲、今回対象外にした範囲および予防的対策を施した範囲の割合を表している。下側の帯グラフは、自動訂正対象範囲を100 %として訂正可能なバグ、訂正不可能なバグおよび誤訂正に陥ったバグの割合を表している。検討した訂正手法により正しく訂正できたバグは61.15 %、訂正出来なかったバグは38.66 %であった。また、誤った訂正をしてしまったバグが2件あり、比率は0.19 %である。

4 むすび

C言語による計算機実習を改善するために、実習履歴の分析結果をもとに文法エラーの自動訂正のための幾つかの手法を提案し、その有効性を検証した。提案した文法エラーの自動訂正手法は文法エラーの分析結果から導かれた簡単なルールを用いるものである。多数発生するバグに照準を定めた手法であるため、数種類のルールのみで自動訂正対象範囲のバグの約6割を訂正でき、誤訂正を約0.2 %に抑えることができた。しかし、文法的バグを論理的にも正しく自動訂正するためにはプログラムの機能解析や目的解析までも必要となってくる。提案した手法のような機械的なアルゴリズムだけでは自動訂正可能率の向上にも限度があり、誤訂正を無くすることも非常に難しい。今後は、今回提案した対策や手法を用いたプログラミング環境を実現するとともにプログラムの意味、機能および目的にまで踏み込んだ自動訂正手法の開発に取り組んでいきたい。

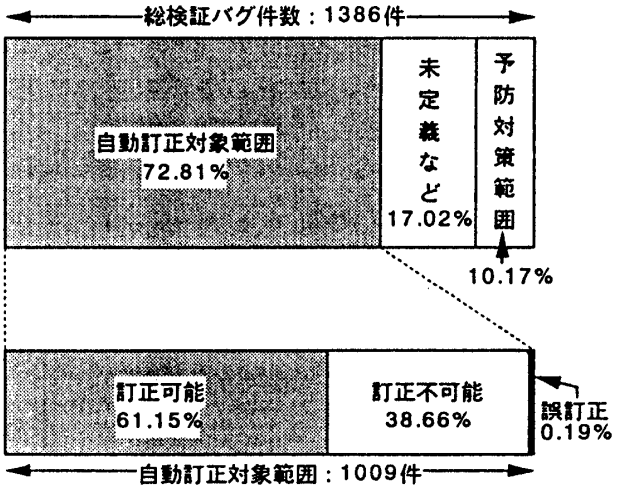

図 2: 訂正可能/不可能および誤訂正の割合

参考文献

[1] 梶浦文夫, 木村宏：”Cプログラミング実習における文法エラーの分析”, 情報処理学会47回全国大会論文集(1), pp.27-28 (1993).

[2] 梶浦文夫, 宮地功：”C言語実習における初心者の文法エラーの分析”, CAI学会誌, Vol.11, No.2, pp.96-103 (1994)