

プログラミング教育を目的とした言語Cパーザとインタプリタの開発

3S-8

玉木裕二, 森田雅夫†, 並木美太郎, 高橋延匡
 (東京農工大学 工学部 電子情報工学科 †現 富士ゼロックス)

1. はじめに

我々は、言語Cのプログラミング教育システムの研究、開発を行っている[1]。このシステムは、学習者が演習形式で学習を行い、システムが教師となり、適切な指導を行えることを目標としている。しかし、このシステムを実現するためには、学習環境の構築、学習者モデルの作成など、解決しなければならない問題点が多い。我々は、演習形式の学習が可能で、プログラム実行環境を構築することから着手し、まず、言語Cのパーザ、インタプリタを開発した。本稿では、このプログラム実行環境の構成と、その中枢となる言語Cパーザとインタプリタの機能について述べる。

2. プログラム実行環境の構成

プログラム実行環境は、言語Cプログラミング教育システムの学習環境として位置づけられる。この実行環境は、入力からデバッグまで、統合的な支援が可能である。

プログラム実行環境の構成を図1に示す。この図に示すように、エディタを中心に統合環境を構成する。この構成の特徴は、エディタにプラットフォームの役割をもたせ、各部から出力される様々な情報を受け取らせる仕様になっていることである。

2.1 インタフェース部

インタフェース部は、マルチウィンドウのエディタで構成され、その名の通り、学習者とのインタフェースを司る。学習者は、ここでプログラムを記述するほか、プログラム実行環境に用意した各機能を、コマンドを発行することで呼出し、使用することができる。また、各部から受け取った情報を、ウィンドウを開いてそこに出力することができる。

2.2 構文解析部

構文解析部は、編集時のプログラムを構文解析し、解析木を生成する。言語Cパーザから構成される。この言語Cパーザの機能については、後述する。

2.3 プログラム実行部

プログラム実行部は、コースウェアの例題や学習者の作成したプログラムの実行を行う。言語Cインタプリタで構成される。この言語Cインタプリタの機能については、後述する。

2.4 静的解析部

静的解析部は、構文解析部の出力した解析結果を入力とし、各処理を行うツール群により構成される。ツールの例として、プログラム静的解析ツール、仕様書自動生成ツール、プログラム整形ツールなどがある。これらのツールは、おもに保守の支援を目的としている。

2.5 動的解析部

動的解析部は、プログラム実行部が出力した実行時データを用いて、実行効率の評価支援とプログラムテストの支援を行う。

3. パーザの機能

ここでは、プログラム実行環境の構文解析部を構成する、言語Cパーザの機能について述べる。

(1) ソースプログラムとの対応が取れた解析木を生成する

ソースプログラムに対し、構文解析を行うことによって、言語Cの構文規則に対応した解析木を生成する。これを利用することにより、言語Cインタプリタなどの各アプリケーション

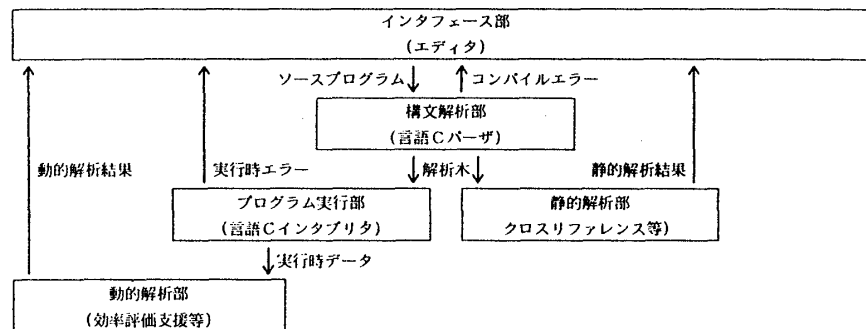


図1 プログラム実行環境の構成

ョンは、構文解析の機能を積極的に利用できる。この解析木は、ソースプログラムへの逆ポインタを持ち、ソースプログラムと1対1の対応が取れる。これにより、構文解析部の出力を入力とする各アプリケーションは、ソースプログラムの情報を得ることができる。例えば、インタプリタがこの解析木を解釈、実行したとき、実行時エラーが発生しても、ソースレベルでエラーの位置を特定できる。実行時エラーの位置をソースレベルで特定できることは、教育という観点では重要なことである。

(2) 任意の構文要素を解析の対象として指定できる

学習者のプログラミングを調査したところ、ケアレミスによるコンパイルのし直しが多かった。ソースプログラム中の一箇所を訂正しただけで、そのファイルを最初からコンパイルしては、効率的な学習は行えない。

このパーザは、再度構文解析を行う場合、一度生成した解析木を参照して、ソースプログラムの訂正による影響がない部分は解析は行わず、訂正が行われた部分だけを解析し、解析木を再構成することができる。

(3) 参照関係を管理している

プログラム中の1箇所を修正すると、他の複数の箇所に影響が及ぶことが多い。例えば、関数の引数の仕様を変更した場合、プログラム中でその関数を呼び出している箇所が影響を受ける。プログラムのサイズが大きくなればなるほど、参照関係を把握することは困難になってくる。

パーザは、プログラム中の識別子、タイプデフ名、マクロ名など名前に関する参照関係をすべて管理し、ソースプログラムの修正による影響が出た場合、これを学習者に指摘し、注意を促すことができる。

(4) コンパイルエラーを適切に指摘できる

我々は、上記のプログラム実行環境中で、学習者がコンパイルエラーを修正するとき、次のような利用モデルを想定している。すなわち、パーザが編集時のプログラム中にコンパイルエラーを検出したとき、構文解析を中断し、エラーの発生した位置にカーソルを移動する。学習者はそのエラーの訂正を行うというように、システムの指摘するエラーを一つずつ、インタラクティブに取り除いていくモデルである。一つのエラーが、その後の複数のエラーの原因になっていることが多く、また、多くのエラーメッセージは、学習者に嫌悪感を与えたと考えたからである。

教育という用途を考慮すると、行単位だけでなく、カラム単位でも指摘できることが望ましい。なぜなら、言語Cのソースプログラムでは、1行の中にも細かい構造を持っているからである。上記のような利用モデルを考慮すると、コンパイルエラーを検出した箇所だけでなく、ソースプログラムを直すべき箇所を指摘できなければならない。

4. インタプリタの機能

ここでは、プログラム実行環境のプログラム実行部を構成する、言語Cインタプリタの機能について述べる。

(1) エラーチェックを重視する

このシステムでは、学習者の作成したプログラムに誤りがあれば、その原因を解説し、誤りを正しい方向へ導けるようにしたい。そのためには、システムが誤りを的確に捕らえられなければならない。また、誤りのあるプログラムを実行す

ると、何が起こるか、実際の動作を見せることで解説を行いたい。

このような観点で、このインタプリタはエラーチェック、特にポインタのエラーチェックに重点を置いて実現を行った。学習者にとって、ポインタが最も理解しにくい内容であり、ポインタに関する間違いが最も多かったからである [1]。例えば、次に示す事柄を事前にチェックすることができる。

- (a) 関数コール時の、引数の個数、型が正しいか
- (b) ポインタ変数で、領域の確保を行う前に値を代入した
- (c) 配列の大きさを越えてアクセスした
- (d) すでに解放された領域をアクセスした

(2) 内部構造の動的な変化をアニメーション表示できる

この言語Cインタプリタでは、ソースプログラムに出現する変数や、システムが用いるスタックなどを、内部でメモリ管理セル [1] を用いて管理している。このメモリ管理セルを監視し、変数の値が書き変わったときや、スタックに値が積まれたときなど、データ構造の動的な変化をアニメーションで表示することができる。これを用いて、プログラムの動作を、視覚的に説明することができる。

(3) デバッガの機能を持っている

学習者が作成したプログラムを実行することを考慮すると、デバッグを支援するために、デバッガの機能は必要である。そして、学習者の欲しい情報がいつでも得られるように、システムが持つすべての情報を、学習者に公開する方式をとった。これにより、プログラムの間違いを、学習者自身が容易に調べることができる。

(4) 実行時データの計測を行う

プログラムの動的解析を行い、プログラムの性能の測定を行う。具体的には、変数のアクセス回数、演算の実行回数、トレースデータなどのデータを計測する。

(5) オブジェクトモジュールを追加リンクする

インタプリタは実行速度が遅いという欠点を持っている。この実行環境では、実行速度を向上させるために、オブジェクトモジュールを後からリンクし、実行することが可能である。我々のOSは実記憶系のOSであるが [2]、この実記憶系のOSで、動的リンクを行うことに相当する。我々は、この方式を追加リンクと呼んでいる。

5. おわりに

上記の機能を持つ言語Cパーザとインタプリタを、OS/omicon第2版上で開発した。これにより、教育用のプログラム実行環境の中核が実現できた。今後は、プログラム実行環境の他の部分を実現し、実用する予定である。

参考文献

- [1] 森田雅夫他：“言語CのCAIシステムの設計とそのプログラム実行環境の開発”，情処学会プログラミングシンポジウム報告集，情報処理学会，1991.1
- [2] 高橋延匡：“研究プロジェクト総説OS/omiconの開発”，情処学会オペレーティングシステム研究会報告39-5，1988