4E-1

# 教育用に拡張されたPascalとその支援環境について

齋藤明紀　林正和　辻野嘉宏　都倉信樹

(大阪大学)

## 1．はじめに

当学科（大阪大学基礎工学部情報工学科）では，大型計算機MELCOM COSMO900II上で，Pascalを使用して1年生に対するプログラミング教育を行っている．その場合計算機を使用するための手順が複雑であるという問題点がある．また，モジュールや抽象データ型などの新しい概念に関する教育も行いたいという希望がある．そこで，標準Pascal［1］を拡張した言語PascalΣXEを設計し，それを核とした初心者教育支援環境を実現した．この環境を用いることにより，初心者は，OSの操作など本質的でない複雑な操作にわずらわされることなく高度なプログラミング概念の習得に集中することができる．また，PascalΣXEを使用することにより初心者に対する各種アルゴリズムの教育が容易になる．

## 2．本環境の概略および基本方針

プログラムの作成は図1のように，エディト，コンパイル，実行を繰り返して行われる．プログラミング演習においては，このサイクルが効率よく行えることが重要である．

そこで，図2に示すように，エディタ，コンパイラ，実行系（インタプリタ）を一つに統合した環境を考えた．これにより，この環境から出ることなく，速やかに図1の状態遷移を行うことが出来る．

PascalΣXE及び本環境は以下のような方針で設計，作成されている．

・教科書，参考書の豊富な点などを考慮し，Pascalから離れ過ぎないようにする．
・利用形態としては初心者のプログラミング演習を想定する．
・言語仕様，環境の仕様ともに実現が容易であること．
・移植性に十分配慮する．

## 3．Pascalの拡張

Pascalはデータ型や制御構造が豊富な点などから教育用としては定評がある．また，多くの計算機上で利用可能であり，入門書，解説書も入手が容易である．そこで，各種アルゴリズム，概念の教育を行いやすいようにPascalを拡張した言語PascalΣXEを設計した．主な拡張点は以下のとおりである．

(1)モジュールの導入

主目的は，データとそれを取り扱う手続き（関数）を一まとめに書くこと（モジュール）を可能にし，さらにモジュールの内部を外から隠すことによってADT（抽象データ型：abstract data type）の概念を教えることである．

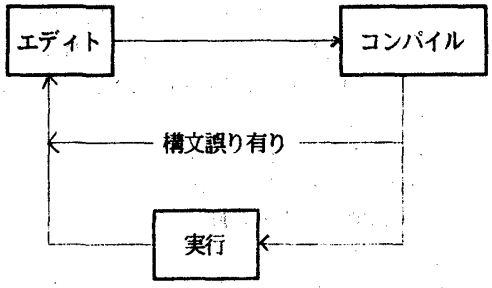


図1　プログラミングの過程（概念）

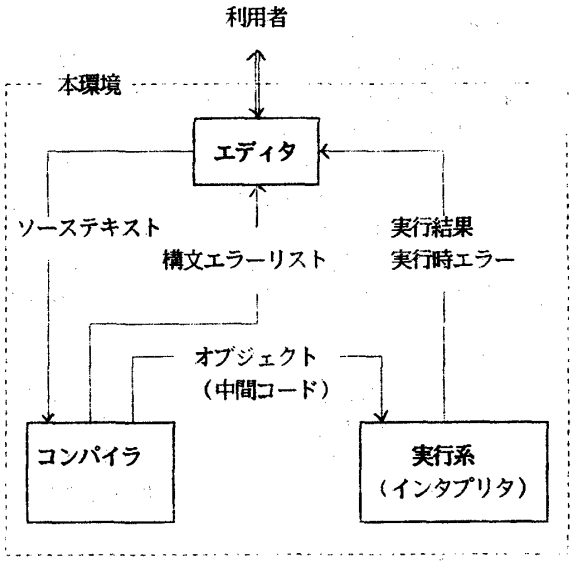


図2　本環境の内部構成

ここで言うモジュールとは，定数，型，変数，手続き，関数宣言からなるファイルのことである．module文によりモジュールファイルをソーステキスト中に取り込むことができる．ただし，言語Cのインクルード機能のように単に展開するだけではなく以下のような機能を持つ．

・モジュール内部では，宣言のレベル（スタティックレベル）は，外と同じであるが，名前のスコープが，1レベル深くなる．つまり，任意の名前の作業変数，内部手続きをつかってよい．

・モジュール内の，一番浅いレベルで，varやtypeのまえに陽に予約語exportをつけて宣言することにより，その名前をモジュールの外側でも有効となるように出来る．そのと

---

An Extended Pascal for Programming Education and its Environment
Akinori SAITOH, Masakazu HAYASHI, Yoshihiro TSUJINO, Nobuki TOKURA
OSAKA UNIVERSITY

き，以下のような制約を加えた名前の参照のみを許すことで，ＡＤＴを実現している．

・export const
その値を参照すること

・export type
その型を持つ，指すあるいは含む変数，その型を引き数として取る手続き，関数の宣言および使用（ただし，型の内部構造は，モジュール外には見せないので，その型全体を代入すること，引数として渡すこと，同一かどうかの判定だけが許される）．

・export var
その変数を参照すること．

・export procedure/function
その関数，手続きを使用すること．

(2)宣言順序の自由化
const,module,type,var,procedure,functionの宣言順序を自由化する．これにより，例えば大域的に使用される型や変数のみをプログラム先頭で宣言し，本体内の作業変数は本体の直前に書くことができる．

(3)return,break文，case文のdefaultの導入とgoto文の制限
無制限なgoto文の使用をさせないため，関数，手続きから脱出するgoto文を禁止した．またアルゴリズムの記述を容易にするためreturn文などを導入した．case文にどのケースラベルにも一致しない場合の指定を追加した．

(4)手続き，関数引数のパラメタの型の宣言の追加
標準Pascalでは，手続き，関数引数のパラメタを書かなくてよいが，型チェックを行うため，手続き，関数引数のパラメタの型宣言をするようにした．

(5)文字列型の導入
テキスト処理を容易にするため文字列を専門に扱う型として，stringを追加した．たとえば，var　s:string[100];とすると，sは最大１００文字まで扱える（可変長の）文字列型となる．又，文字列型どうし，文字列型と文字配列は相互に代入可能である．

(6)集合型の拡張
集合型はその許される基底型の大きさは少なくとも文字型よりも大きいことを言語仕様で規定した．

(7)整数のビット演算の追加
and,or,notを整数型に対して用いた時は，ビット演算を行う．シフトは標準関数で持つ．

(8)ファイル処理機能の変更
標準Pascalには外部ファイル変数とその（実際の）ファイル名をプログラムの中から対応づける機能がない．そこで，プログラム中のファイル変数とファイル名を対応づける標準手続きを追加した．

## ４．プログラミング環境の整備

初心者に対するプログラミング教育においては，演習を行う環境も大きな問題となる．エディタ，コンパイラ，リンカなどの操作法が分からなくて，つまずく初心者も多い．また，例えばcase文での値が制限を越えたときと，配列の添字の値が制限を越えたときに同じエラーメッセージになっているなど，コンパイラ等のエラーメッセージが不適切であることがある．また，適切であっても，エラーメッセージに含まれる専門用語がわからないためそれが理解できないでいるという問題がある．そこで初心者を主な対象とし，PascalΣXEを核として統合的環境を作成した．

本環境の特徴としては，次のようなものがある．

(1)利用者にとっては，コマンドの増設されたエディタに見える．
コンパイル，実行等はエディタのコマンドによって起動される．利用者は，エディタ，コンパイラ等の存在を意識する必要はない．例えば，'ｒｕｎ'コマンドによって実際には，コンパイルが行われ，文法誤りが無ければ，自動的に実行系が起動される．

(2)コンパイラ，実行系と連携してエディタが動作する．
コンパイル時に構文エラーが検出された場合や，実行時エラーが起こった場合には，自動的にエディタに戻る．そして，エラーメッセージを表示して，エラー行にカーソルをセットする．

(3)コンパイル速度を重視している．
プログラミング演習では，目的プログラムの実行速度よりもコンパイルが速いことが重要となる．そこで，高速化のためこのコンパイラは１文字先読みの１パスコンパイラとなっており，エディタのテキストバッファから直接ソースプログラムを得てオブジェクト（中間コード）をメモリ上に出力している．

(4)実行系は中間コードインタプリタである．
実行時エラーを正確に検出するため，また，コンパイルを高速に行うため，さらに将来の拡張（デバッガの作成等）を考慮して実行系は中間コードインタプリタとなっている．

(5)利用者にとって親切な文法誤りの処理を行う．
文法誤りを検出したとき，正確にまた親切に，それを指摘することに重点をおいて作られている．

・エラーが検出された行でなく，エラーの原因になった行を（可能な限り）指摘する．

・一つのエラーの影響で，無意味なエラーを大量に出すことをさける．

・エラー検出後も出来る限りコンパイルを続行し，文法誤りを見つけるためのコンパイルの回数を減らす．

(6)モジュールあるいはプログラムの断片（作成途中のプログラム）に対して文法チェックを行うことができる．

(7)実行結果の内,CRT出力分は実行と並行して自動的にファイルに保存する．

## ５．まとめ

エディタ，コンパイラ，インタプリタを統合することにより，初心者が複雑なＯＳ等の操作から解放され，プログラミング演習に集中することを可能にした．また標準Pascalにくらべ，ＡＤＴ，モジュールなどのプログラミング上の概念を教えやすくなっている．今後は実際の初心者教育に使用することで，プログラミング教育を支援する機能を更に高めて行きたい．

参考文献

[１] K.Jensen,N.Wirth,PASCAL User Manual and Report, Springer-Verlag,1974.