Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

PASTALK
Uma Extensão Orientada a Objetos do Turbo Pascal ™

por

Luiz Henrique de Araujo Vecchio
Ivan Moura Campos
Roberto da Silva Bigonha

Relatorio Técnico RT030/89

Caixa Postal 702
30.161 - Belo Horizonte - MG
1989

SINOPSE

A fim de permitir que a Programação Orientada a Objetos em TURBO Pascal [1] seja feita de forma mais natural, é necessário eliminar detalhes de implementação que não fazem parte da aplicação. Este artigo apresenta o PASTALK, uma extensão orientada a objetos do TURBO Pascal, que possibilita a utilização do PPOO em uma linguagem procedimental - neste caso, o TURBO Pascal 5.0.

Já os identificadores dos procedimentos Chama_Classe seguem o padrão descrito abaixo:

CLASSH	Chama_Classe
1 Documento	Chama_Documento
2 Periodico	Chama_Periodico
...	...
1 Escaninho	Chama_Escaninho

Os nomes das variáveis ADDR utilizadas pela rotina SEND - arquivo ADDRVAR.PAS - para a ativação dos procedimentos Chama_Classe, possuem o seguinte formato:

CLASSH	ADDRClasse
1 Documento	ADDRDocumento
2 Periodico	ADDRPeriodico
...	...
1 Escaninho	ADDREscaninho

Os arquivos complementares gerados pelo Macro Expansor são de 2 tipos:

- UNITs, que por este motivo devem ser incorporados à cláusula USES do módulo que os referencia;
- INCLUDEs, arquivos que serão incluídos em um determinado módulo como parte complementar da declaração dos tipos, constantes e variáveis do programa.

Além das UNITs que correspondem às classes da aplicação, outras duas são produzidas como resultado final da "compilação" de um programa PASTALK:

USEND.PAS
USTACK.PAS

A fim de que a troca de mensagens entre as UNITs seja feita, a incorporação desses módulos ao programa-fonte TURBO Pascal é feita através do comando USES de cada uma das classes existentes na aplicação. Por exemplo, o comando USES para a classe PERIODICO é:

```
USES UDOCTEXT, USEND, USTACK;
```

Onde, DOCTEXTO é a superclasse de PERIODICO.

Os arquivos OBJMSG.PAS, METHODS.PAS e ADDRVAR.PAS referem-se, respectivamente, à declaração de tipos, constantes e variáveis. Por decisão de projeto, estes arquivos devem ser incorporados ao módulo que contém as variáveis globais da aplicação e que faz parte do comando USES de todos os módulos e/ou classes da aplicação.

Para a classe PERIODICO, o comando USES correspondente é:

```
USES UDOCTEXT, USEND, USTACK, UGLOBAIS;
```

Os módulos complementares necessários à ativação de uma classe devem ser incluídos posteriormente.

Para que não seja atingido um nível de granularidade de objetos semelhante ao da linguagem SMALLTALK, apenas ações que estejam relacionadas diretamente à solução do problema devem ser colocadas em classes (CLASS). Os chamados "Procedimentos de Serviço" devem continuar sendo utilizados como na linguagem Pascal, ou seja, sendo declarados em UNITs e acionados da maneira tradicional.

A fim de facilitar a identificação e, consequentemente, a tradução para o formato TURBO Pascal, as linhas de programa que são comandos PASTALK têm na primeira coluna o símbolo %. Desta forma, o processamento do programa de entrada consiste das seguintes ações:

```
"leia linha do texto";
Se "tem % na primeira coluna"
  entao "processa linha";
  senao "imprime linha";
fim se;
```

O formato do arquivo UPERIODI.PTK contendo a classe PERIODICO, antes de ser traduzido para o formato TURBO Pascal 5.0, pode ser visto a seguir:

```
%CLASS UPeriodi;
%SUPERCLASS UDocumen;
%INSTVAR Titulo, Assunto: String;
%METHOD Novo(var Obj: APontObj);
%BEGIN
     new(Obj);
    "inicializa TAGs";
% Inic(Obj);
%END;
%METHOD edita(var Obj: APontObj; IN Tmp: St; OUT Titulo: St);
%BEGIN
     ...
%END;
%METHOD cabec(...);
%BEGIN
%END;
%METHOD inic(var Obj: ApontObj);
%BEGIN
 with Obj^ do
 begin
  Titulo := '';  Assunto := '';
  DataChegada := Hoje;
 end;
%END;
%END.
```

Embora não seja a solução ideal, visto que o texto de entrada para o Macro Expansor possui caracteres estranhos à linguagem PASTALK, tem se mostrado satisfatória na obtenção dos seguintes objetivos:

- eliminação dos detalhes de implementação dos conceitos relativos ao PPOO;
- compreensão do ambiente de RUN-TIME das Linguagens Orientadas a Objetos;
- Manutenção do "conforto" sintático existente quando da programação na linguagem Pascal.

A saída produzida pelo macro expansor, após a análise do trecho de programa escrito em PASTALK - classe PERIODICO - pode ser vista a seguir. As palavras e trechos de programa escritos em letras maiúsculas representam o código gerado pelo FRONT-END.

```
UNIT UPERIODI;
INTERFACE
USES UDOCTEXT, USEND, USTACK, UGLOBAIS;
PROCEDURE CHAMA_PERIODICO(VAR OBJ: APONTOBJ);
IMPLEMENTATION
PROCEDURE CHAMA_PERIODICO;
VAR MSG: MSGREG; PARAM: PARAMREG;
PROCEDURE MNOVO(var Obj: ApontObj);
begin
 new(Obj);
"Inicializa Tags";
MSG.SELETOR := INIC;
SEND(MSG,OBJ);
end;
PROCEDURE MEDITA(var Obj:ApontObj);
VAR TMP: STR; TITULO: STR;
begin
POPPARAM(PARAM);
TMP := PARAM.STR;
 ...
 PARAM.ST := TITULO;
 PUSHPARAM(PARAM);
end;
PROCEDURE MCABEC( ... );
begin
...
end;
PROCEDURE MINIC(var Obj: ApontObj);
begin
 with Obj^ do begin
  Titulo := '';
  Artigo := '';
  DataChegada := Hoje;
 end;
end;
BEGIN
 POP(MSG);
 CASE MSG.SELETOR OF
  NOVO: MNOVO(OBJ);
```

```
  EDITA: MEDITA(OBJ);
  CABEC: MCABEC(OBJ);
  INIC:  MMINIC(OBJ);
  ELSE BEGIN
      PUSH(MSG);
      CHAMA_DOCUMENTO(OBJ);
     END;
 END;
END;
BEGIN
 ADDRPERIODICO := ADDR(CHAMA_PERIODICO);
END.
```

A primeira transformação feita pelo Macro Expansor é a substituição dos comandos CLASS e SUPERCLASS pelos correspondentes UNIT e USES. A substituição do comando CLASS é feita diretamente:

```
CLASS PERIODICO      UNIT UPERIODI
```

Ao analisar as variáveis declaradas pelo comando CLASSVAR, o Macro Expansor transforma-as em variáveis globais à UNIT, permitindo que todos os métodos da classe PERIODICO tenham acesso às mesmas. Já as variáveis associadas ao comando INSTVAR são colocadas no registro base - OBJETO (arquivo OBJMSG.PAS).

A transformação do comando INSTVAR em uma sequência de comandos válidos no TURBO Pascal requer 2 etapas:

- agrupamento de todas as variáveis declaradas por este comando;
- geração do arquivo OBJMSG.PAS, tomando-se como base o arquivo CLASSH.

Após "compilar" todas as classes da aplicação, o programa tradutor gera o registro base OBJETO com o seguinte formato:

```
OBJETO = record
...
case Classe: TipoClasse of
  Documento: (case SubClasse: TipoClasse of
    DocTexto: (...);
    Periodico: (titulo, assunto: String););
  Escaninho: (...);
end;
```

Onde,

- "título" e "assunto" são variáveis declaradas pelo comando INSTVAR da classe PERIODICO.

Os corpos dos métodos no comando CASE dos procedimentos Chama_Classe foram substituídos por chamadas aos métodos correspondentes. Assim, para diferenciar os identificadores dos procedimentos (Métodos), dos "Labels" do comando CASE (seletores), o padrão de formação dos identificadores passa a ser:

MMetodo

A substituição dos parâmetros de um método por chamadas às rotinas POPPARAM e PUSHPARAM permitem a padronização da declaração e, consequentemente, da chamada de todos os métodos:

```
MMetodo (var Obj: ApontObj);
```

Deve-se ressaltar ainda a inclusão de trechos que substituem a declaração dos parâmetros formais do método EDITA. Os parâmetros pertencentes à cláusula IN são substituídos por comandos do tipo POPPARAM, e os associados à cláusula OUT por comandos PUSHPARAM. Além disto, os parâmetros formais são transformados em variáveis locais ao procedimento, fazendo com que as referências aos parâmetros dentro dos procedimentos MMétodo permaneçam inalteradas.

Finalmente, ao encontrar a sequência que indica fim da declaração de uma classe - "END." - o Macro Expansor gera automaticamente:

- o comando CASE com as entradas para cada um dos métodos encontrados;
- o trecho de inicialização da variável ADDRClasse (utilizada pela rotina SEND).

REFERÊNCIAS

[1] Vecchio, L. H. A., Campos, I. M., Bigonha, R. S., "*O Paradigma de Programação Orientada a Objetos em Turbo Pascal*", Monografia do Departamento de Ciência da Computação, 1989.

[2] Goldberg, A., Robson, D., "*SmallTalk-80: The Language and its Implementation*", Addison-Wesley Publishing Company, Reading Massachusetts, 1983.

[3] Jacky, J. P., Kalet, Ira J., "*An Object-Oriented Programming Discipline for Standard Pascal*", Communications of the ACM, v. 30, n. 09, Setembro 1987.

[4] Bigonha, M. A. S. e Bigonha, R. S., "*SIC: Sistema de Implementação de Compiladores*", Monografia do Departamento de Ciência da Computação, 1986.

[5] Bergin, J., GreenField, S., "*What does Modula-2 Need to Fully Support Object-Oriented Programming ?*", Division of Computer Science and Mathematics, Marist College, PoughKeepsie, NY.

[6] "*TURBO Pascal 5.0 - Owner's HandBook*", Califórnia, Borland International, Inc., 1987.

Indice

INTRODUÇÃO

A Programação Orientada a Objetos em linguagens procedimentais, notadamente em C, Modula e Pascal, deve ser vista como uma alternativa para a manutenção do conforto sintático proporcionado por estas linguagens, utilizando ao mesmo tempo os benefícios advindos do Paradigma de Programação Orientada a Objetos (PPOO).

Com PASTALK não se objetiva a criação de uma nova linguagem de programação, mas sim a possibilidade de se estudar o PPOO dentro de um ambiente familiar.

O PASTALK pode ser considerado uma extensão do TURBO Pascal, ou seja, além dos comandos já existentes foram incorporadas outras funções que facilitam a programação orientada a objetos. As alterações sintáticas existentes em PASTALK, notadamente na definição das classes e uma comparação entre as linguagens SMALLTALK, TURBO Pascal e PASTALK são feitas a seguir.

SINTAXE DO PASTALK

A nível de sintaxe, as modificações introduzidas no TURBO Pascal 5.0 restringem-se à criação de uma nova estrutura de controle - CLASS. A estrutura CLASS permite a declaração das classes existentes em uma aplicação e dos métodos que manipulam suas variáveis. Além de facilitar a programação dentro do paradigma de programação orientada a objetos, a estrutura CLASS torna transparente vários detalhes de implementação descritos em [1]. Pode-se dividir a nova estrutura em 3 partes:

- identificação da classe;
- declaração das variáveis de classe e de instância;
- declaração dos métodos que realizam operações sobre as variáveis dos objetos criados a partir da classe em questão.

Um exemplo da estrutura CLASS é mostrado a seguir:

```
CLASS ClasseX;
SUPERCLASS ClasseY;
CLASSVAR Variavel3: Tipo2;
INSTVAR Variavel1, Variavel2: Tipo1;
METHOD Metodo1(var Obj: ApontObj);
begin
 ...
end;
METHOD Metodo2(var Obj: ApontObj; IN Param1, Param2: Tipo3);
begin
 ...
 Metodo1(Obj);
end;
METHOD Metodo3(var Obj: ApontObj; OUT Y: Tipo3);
var X: Tipo3;
```

```
begin
 Metodo2(Obj, IN X,Y);
 ...
end;
END.
```

Onde,

- CLASS especifica o nome da classe;
- SUPERCLASS define a superclasse de ClasseX;
- INSTVAR e CLASSVAR declaram, respectivamente, as variáveis de instância e variáveis de classe de ClasseX;
- METHOD define os métodos que pertencem a classe ClasseX.

Resumidamente, as principais modificações introduzidas na linguagem base, que resultaram na criação do PASTALK, são as seguintes:

- substituição do comando UNIT pelo comando CLASS para a definição de classes;
- declaração dos métodos implementados por uma classe, feita de maneira similar a do SMALLTALK:

```
Method Metodo1(...);
begin
 ...
end;
```

- acionamento de um método, feito como se fosse um procedimento;
- os parâmetros de entrada são precedidos pela palavra IN, e os de saída pela palavra OUT;
- eliminaçao do comando CASE, que se constituía na principal estrutura de controle dos procedimentos Chama_Classe;
- eliminação do trecho de programa que realiza a pesquisa de métodos na superclasse associada;
- estruturação do registro base, do qual todos os objetos são instâncias, feita automaticamente a partir das variáveis declaradas pelos comandos CLASSVAR e INSTVAR;
- declaração da rotina SEND e do trecho de programa que realiza a procura de um método na superclasse, feitos através de rotinas semânticas associadas às produções da nova gramática.

Os exemplos apresentados nas seções a seguir referem-se a uma aplicação de Automação de Escritórios [1].

COMPARAÇÃO ENTRE SMALLTALK, PASTALK E TURBO PASCAL

Para facilitar o entendimento das modificações introduzidas, apresenta-se a seguir a UNIT PERIODICO escrita em três versões: SMALLTALK, TURBO Pascal e PASTALK.

SMALLTALK

```
Documento subClass: #Periodico
instanceVariableNames: 'titulo assunto'
classVariableNames: ''
poolDictionaries: '' !

| Periodico Class methods |
novo
 | obj |
 obj := super new inic.
 ^obj !

| Periodico methods |
edita
 titulo := Prompter prompt: '----Titulo----' default: titulo.
 assunto := Prompter prompt: '---Assunto---' default: assunto.
cabec
 ^'Periodico: ',dataChegada formPrint.
inic
 titulo := ''.
 assunto := ''.
 dataChegada := Date Today.
```

PASCAL

```
UNIT Periodico;
INTERFACE
USES Documento;
Procedure Chama_Periodico(var Obj: ApontObj);
IMPLEMENTATION
Procedure Chama_Periodico;
var Msg: MsgReg; Param: ParamReg;
begin
 pop(Msg);
 case Msg.Seletor of
  mNovo: begin
       new(Obj);
       {inicializa TAGs};
       Msg.Seletor := Inic;
       Send(Obj,Msg);
       end;
  edita: ...
  cabec: ...
  inic: with Obj^ do
     begin
      Titulo := '';
```

```
        Assunto := '';
        DataChegada := Hoje;
       end;
    else  begin
       (* realiza pesquisa superclasse *)
       push(Msg);
       Chama_Documento(Obj);
       end;
  end; (* case *)
end;
begin
 AddrPeriodico := Addr(Chama_Periodico);
END. (* Unit *)
```

PASTALK

```
CLASS Periodico;
SUPERCLASS Documento;
INSTVAR Titulo, Assunto: String;
METHOD Nove(var Obj: ApontObj);
begin
 new(Obj);
 "inicializa TAGs";
 Inie(Obj);
end;
METHOD edita(var Obj: ApontObj; IN Tmp: St; OUT Titulo: St);
begin
 ...
end;
METHOD cabec(...);
begin
 ...
end;
METHOD inie(var Obj: ApontObj);
begin
 with Obj^ do
 begin
  Titulo := '';
  Assunto := '';
  DataChegada := Hoje;
 end;
end;
END. (* Unit *)
```

Analisando os trechos de programa escritos em PASTALK e SMALLTALK, as seguintes correspondências podem ser feitas:

PASTALK	SMALLTALK
CLASS	subClass:
INSTVAR	instanceVariableNames:
CLASSVAR	classVariableNames:

Um dos pontos que devem ser analisados mais detalhadamente é a passagem de parâmetros entre métodos. Em SMALLTALK, a mensagem Idade at: 'Joao' put: 33,

possui como seletor "at:put:" e como parâmetros de entrada "Joao" e "33". O objeto que recebe a resposta é "Idade".

Em PASTALK, o comando correspondente é:

```
atput(Idade; IN 'Joao',33);
```

A declaração do método "atput" em PASTALK é:

```
METHOD atput(var Obj: ApontObj; IN Nome: String; Id: Integer);
```

Caso um parâmetro de saída seja necessário, este pode ser incluído na declaração do método através do comando OUT.

```
METHOD atput(var Obj: ApontObj; IN Nome:String; Id: Integer; OUT Resp: Boolean);
```

Com relação aos trechos escritos em PASTALK e Pascal, as correspondências são:

PASTALK	PASCAL
CLASS	UNIT
SUPERCLASS	USES
INSTVAR	Objeto = record
	...
	classe: (variáveis)
	...
	end;

No caso específico da passagem de parâmetros - seletor at:put: - a tradução do trecho de programa PASTALK

```
atput(Idade; IN 'Joao',33; OUT Resp);
```

para PASCAL é:

```
Param.Str := 'Joao';
PushParam(Param);
Param.Int := 33;
PushParam(Param);
Msg.Seletor := atput;
Send(Msg,Idade);
PopParam(Param);
Resp := Param.Boo;
```

Em PASTALK, a troca de mensagens entre os objetos é feita como se fosse uma chamada de procedimento em Pascal. As duas exceções ficam por conta da obrigatoriedade de se colocar como parâmetro o objeto a que se destina a mensagem, e de se distinguir os parâmetros de entrada, precedidos pela palavra IN, e os de saída, precedidos pela palavra OUT.

UM FRONT-END PARA A POO EM TURBO PASCAL

A modificação da sintaxe do TURBO Pascal requer a existência de um compilador que aceite como entrada programas escritos em PASTALK e produza o código-objeto executável. Entretanto, como não faz parte dos objetivos imediatos dos trabalhos a construção de um compilador, optamos pela criação de um Macro Expansor que a partir de um programa PASTALK gere o código correspondente em TURBO Pascal. Resumidamente, o esquema de tradução de um programa PASTALK para TURBO Pascal pode ser visto na Figura a seguir:

Para que o Macro Expansor traduza o programa PASTALK, é necessário que a hierarquia de classes da aplicação seja também fornecida. De posse destas informações, é possível gerar os seguintes arquivos complementares:

USEND.PAS METHOD.PAS
USTACK.PAS ADDRVAR.PAS
CLASSH.PAS OBJMSG.PAS

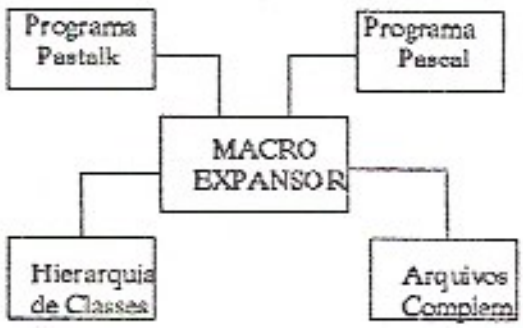

Figura 1

A descrição do conteúdo e da função de cada um dos arquivos é feita abaixo:

- O programa-fonte PASTALK é constituído do arquivo que representa o programa principal - SOURCE.PAS - e dos arquivos Classe.Ptk, descritos posteriormente;
- CLASSH é um arquivo fornecido pelo usuário e contém a hierarquia de classes. Para o exemplo "Automação de Escritórios", o conteúdo deste arquivo é:

```
1 Documento
2 Periodico
2 DocTexto
3 Carta
3 Telefonema
1 Escaninho
```

onde,

- 1, 2 e 3 representam o nível de cada classe;
- Documento, Periódico, DocTexto, Carta, Telefonema e Escaninho representam os nomes de cada uma das classes;

- USEND.PAS contém a rotina responsável pela troca de mensagens entre as classes definidas em CLASSH;
- USTACK.PAS contém as estruturas de dados e os procedimentos necessários à manipulação das pilhas de mensagens e de parâmetros;
- CLASSH.PAS representa a hierarquia de classes no formato Pascal;

- CLASSH.PAS representa a hierarquia de classes no formato Pascal;
- OBJMSG.PAS contém a declaração do registro base - OBJETO - e do registro de mensagens. A organização do registro base segue o estabelecido no arquivo de hierarquia (arquivo de entrada para o Macro Expansor);
- METHODS.PAS representa o dicionário de métodos da aplicação;
- ADDRVAR.PAS contém as variáveis utilizadas pelo módulo USEND.PAS para transferir o controle de uma classe para a outra.

ESTRATÉGIA DE IMPLEMENTAÇÃO

A implementação do Macro Expansor está dividida em 2 partes:

- tradução dos comandos PASTALK para TURBO Pascal 5.0;
- incorporação dos arquivos adicionais ao programa Pascal correspondente.

Como os arquivos adicionais USEND, OBJMSG e ADDRVAR referenciam todas as classes definidas na aplicação, é necessário que os identificadores dessas classes sejam únicos. Por esta razão, o arquivo de entrada CLASSH, além de definir a hierarquia de classe para a construção do registro base OBJETO, estabelece o padrão a ser empregado na formação dos nomes dos procedimentos Chama_Classe, dos "Labels" para o comando CASE da rotina SEND, dos identificadores das UNITs, e na definição de quais UNITs (Classes) serão traduzidas de PASTALK para TURBO Pascal.

Os nomes das UNITs geradas pelo programa tradutor seguem o seguinte padrão:

UNomeClasse.Pas,

Onde, NomeClasse representa o nome da classe, descrita no arquivo CLASSH, limitado a 7 caracteres ou dígitos - restrição imposta pelo sistema operacional utilizado.

Para o arquivo CLASSH fornecido, os sequintes arquivos são utilizados na compilação do programa-fonte PASTALK:

UDOCUMEN.PTK	UDOCTEXT.PTK
UPERIODI.PTK	UCARTA.PTK
UESCANIN.PTK	UTELEFON.PTK

Como as classes devem ter seus identificadores iguais aos nomes dos seus respectivos arquivos - restrição do TURBO Pascal 5.0 - a declaração dos comandos CLASS deve ter a seguinte correspondência:

CLASSH	Classes
1 Documento	Class UDocumen
2 Periodico	Class UPeriodi
.....	
1 Escaninho	Class UEscanin