UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PASCAL - XSC - PASCAL PARA COMPUTAÇÃO CIENTÍFICA: DESCRIÇÃO, INSTALAÇÃO E APLICAÇÕES

por

Tiarajú Asmuz Diverio

RP - 245 Outubro/1994

Relatório de Pesquisa

UFRGS - II CPGCC
Caixa Postal 15064 - CEP 91501-970
Porto Alegre - RS - BRASIL
Telefone: (051) 336-8399 e 339-1355
Fax: (051) 336-5576
E-Mail: PGCC@INF.UFRGS.BR

SUMÁRIO

RESUMO

PASCAL-XSC é uma linguagem de programação de propósito geral que proporciona condições especiais à implementação de algoritmos numéricos sofisticados, que verificam matematicamente os resultados. O novo sistema PASCAL-XSC tem as vantagens de ser portável a várias plataformas, estando disponível para computadores pessoais (PC), estações da trabalho do tipo Sun e Hp, mainframes e supercomputadores. A portabilidade é garantida pelo uso de um compilador que traduz para a linguagem ANSI-C.

Este sistema proporciona uma completa simulação da aritmética de ponto flutuante definida pelo padrão binário IEEE 754. Graças a isto, programas em PASCAL-XSC produzem resultados idênticos em todas as plataformas.

Pelo uso dos módulos matemáticos do PASCAL-XSC, algoritmos numéricos que providenciam alta exatidão e verificação automática de resultados podem ser facilmente programados. Além disso, a linguagem PASCAL-XSC simplifica o projeto de programas para as Engenharias e para a Computação Científica, graças à estrutura modular dos programas, à possibilidade de definição de operadores, ao *overloading* de funções, rotinas e operadores, às funções e operadores com tipos arbitrários de dados e aos *arrays* dinâmicos.

Outras características presentes nos módulos da aritmética padrão para os tipos adicionais de dados numéricos incluem operadores e funções elementares com alta exatidão e com avaliação exata de expressões.

Os programas escritos em PASCAL-XSC são de fácil leitura, mesmo se existirem as operações com tipos de dados dos espaços matemáticos avançados, onde os operadores usados para as operações obedecem à notação matemática convencional. Existe ainda, uma grande quantidade de problemas numéricos que podem ser resolvidos pelas bibliotecas de rotinas com verificação automática do resultado. O PASCAL-XSC possui grandes facilidades para o desenvolvimento de tais rotinas.

PALAVRAS-CHAVE:

Pascal; Pascal XSC; Linguagem para Computação Científica;
Matemática Intervalar; Intervalos

ABSTRACT

PASCAL-XSC is a general purpose programming language which provides special support for the implementation of sophisticated numerical algorithms with mathematically verified results. The new PASCAL-XSC system has the advantage of being portable across many platforms and is available for personal computers, workstations, mainframes and supercomputers by means of a portable compiler which translates to ANSI-C language. A complete software simulation of the arithmetic defined by the IEEE 754 binary floating-point arithmetic standard is provided. This ensures that PASCAL-XSC programmes procedure identical results om all platforms.

By using the mathematical modules of PASCAL-XSC, numerical algorithms which deliver highly accurate and automatically verified results can be easily programmed. PASCAL-XSC simplifies the design of programs in engineering and scientific computation by modular program structure, user-defined operators, overloading of functions, procedures, and operators, functions and operators with arbitrary result type and dynamic arrays. Arithmetic standard modules for additional numerical data types including operators and standard functions of high accuracy and the exact evaluation of expressions provide the main numerical tools.

Programs written in PASCAL-XSC are easily readable since all operations, even those in the higher mathematical spaces, have been performed as operators and can be used in conventional mathematical notation.

In addition, a large number of numerical problem-solving routines with automatic result verification have been developed. The PASCAL-XSC system greatly facilitates the development of such routines.

KEY-WORDS:

Pascal; Pascal XSC; Language for Scientific Computation;
Intervalar Mathematics; Intervals

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Nas últimas décadas, muitos esforços têm sidos feito para aumentar o poder das linguagens de programação. Linguagens novas e poderosas têm sido projetadas e novas características têm sido acrescentadas às linguagens existentes, assim como o Fortran está em constante progresso. Entretanto, todas estas linguagens ainda necessitam de uma definição precisa em sua aritmética, pois um mesmo programa pode produzir diferentes resultados em diferentes processadores.

Atualmente, as operações aritméticas elementares dos computadores eletrônicos são, geralmente, operações em ponto-flutuante de alta exatidão. Em particular, isto significa que para qualquer escolha de operandos o resultado do cálculo é o resultado exato arredondado da operação com apenas um arredondamento aplicado no final, como é descrito no padrão IEEE 754 (em [IEEE754]). Esta aritmética padrão também requer as quatro operações aritméticas básicas: adição, subtração, multiplicação e divisão (+, -, * e /) com arredondamento direcionado. Um grande número de processadores já proporciona estas operações, entretanto as linguagens de programação mais usuais não permitem um fácil acesso a elas.

Por outro lado, tem sido verificada a migração da computação científica de computadores de propósitos gerais para computadores vetoriais e paralelos. Esses assim chamados supercomputadores providenciam operações aritméticas adicionais, tal como "multiplica e soma", "acumula" ou "multiplica e acumula". Estas operações de hardware podem sempre produzir um resultado de grande exatidão, mas ainda não há nenhum processador, que satisfaça estas exigências, disponível no mercado. Em alguns casos, o resultado calculado por um algoritmo numérico em computadores vetoriais é totalmente diferente do resultado calculado pelo mesmo processador no modo escalar (exemplos em [HAM90] e [RAT90]).

PASCAL-XSC é o resultado de um longo trabalho de um grupo de cientistas para produzir uma ferramenta poderosa capaz de resolver problemas científicos. A definição matemática da aritmética é uma parte intrínseca da linguagem, incluindo a otimização das operações aritméticas com arredondamento direcionado, as quais são facilmente acessadas pela linguagem. Operações aritméticas adicionais para intervalos e números complexos e, mesmo operações vetoriais e matriciais providas pelos módulos aritméticos pré-compilados são definidos com máxima exatidão, conforme as regras de semimorfismo (descrito em [KUL81]).

O desenvolvimento de programas em PASCAL-XSC é garantido pelo sistema de desenvolvimento do PASCAL-XSC. O sistema consiste do compilador e do módulo de execução, os quais são ambos escritos em ANSI C ([ANSI C]). Foi implementada uma grande quantidade de "códigos geradores nativos" para diferentes processadores e sistemas operacionais. O sistema PASCAL-XSC compila um dado código fonte

PASCAL-XSC, transformando-o em código C, o qual é passado a um compilador C, que gera o código objeto. Finalmente o código objeto e as rotinas de execução do PASCAL-XSC são conectadas (*linkadas*) para se ter o programa executável.

Por causa da variada distribuição de compiladores C, o sistema PASCAL-XSC está disponível em vários computadores . Tanto o código fonte do PASCAL-XSC como o código C gerado são portáveis.

Do ponto de vista dos matemáticos, é de fundamental importância que resultados de algoritmos implementados sejam reproduzidos, independente das diferentes capacidades computacionais. Infelizmente, as capacidades aritméticas dos computadores são diferentes quanto a representação de números em ponto flutuante e na maneira como as operações aritméticas são processadas. Portanto, uma mesma aritmética básica exata precisa ser, no mínimo, garantida por uma linguagem de programação.

O sistema de execução do PASCAL-XSC garante um conjunto completo de rotinas, as quais são baseadas no padrão binário de ponto flutuante do IEEE 754. Todas as operações aritméticas são implementadas em software e não dependem das operações reais dos processadores e nem do sistema C em uso. Para se obter um melhor desempenho, o sistema de execução pode ser configurado de tal forma que se adapte à unidade aritmética de hardware do processador em uso.

Neste trabalho é descrita a linguagem Pascal XSC no capítulo 2 através da apresentação de suas principais características.

No capítulo 3 são abordados o ambiente e a instalação do Pascal XSC. A versão tratada aqui se refere ao ambiente do computador do tipo IBM-PC, para o qual foram organizados três disquetes de instalação. Dois para o compilador C e um para o Pascal XSC propriamente dito. Nesse capítulo são abordadas questões de configuração para uma correta instalação e uso.

No capítulo 4 é feita uma breve discussão sobre as versões do Pascal XSC, onde é tratado o estado atual da implementação, o processo de compilação, opções de compilação e criação de módulos e bibliotecas.

No capítulo cinco são apresentados alguns exemplos de aplicações, resolvidos através de programas desenvolvidos em Pascal XSC.

2 DESCRIÇÃO DA LINGUAGEM PASCAL-XSC

PASCAL-XSC é uma extensão da linguagem de programação PASCAL para computação científica. Seu nome vem do inglês, **PASCAL** *e***X***tension for* **S***cientific* **C***omputation*. Ele contém as características do Pascal Padrão; o conceito de operador universal (operador definido pelo usuário); funções e operadores com tipo de resultado arbitrário; *overloading* de rotinas, funções e operadores; *overloading* de atribuição de operadores; *overloading* de rotinas de entrada e saída (*read* e *write* genéricos); conceito de módulos; *arrays* dinâmicos; acesso a *subarrays*; conceito de *string*; arredondamento controlado; produto escalar ótimo (exato); tipo padrão ***dotprecision*** (um formato de ponto fixo abrangendo todo o intervalo do produto de pontos flutuantes); aritmética especial para tipos padrões de complexos e intervalos; aritmética de alta exatidão para todos os padrões; alta exatidão para funções elementares e avaliação exata de expressões (*#-expressions*).

Uma completa descrição da linguagem PASCAL-XSC e dos módulos aritméticos, assim como uma grande variedade de exemplos são dados em [KLA91] e [KLA92]. As novas características desta linguagem são discutidas a seguir.

2.1 Tipos padrões de dados, operadores pré-definidos e funções

Além dos tipos de dados inteiro (*integer*) e real (*real*) do Pascal Padrão, existem ainda os tipos de dados numéricos da figura 2.1 disponíveis no PASCAL-XSC e, como se pode observar os prefixos `r`, `i` e `c` são abreviaturas de real, intervalo e complexo. Os tipos vetores e matrizes são definidos por *arrays* dinâmicos e podem ser usados como intervalos de índices arbitrários.

rvector	vetor de reais;
rmatrix	matriz real;
complex	tipo complexo;
cvector	vetor de complexos;
cmatrix	matriz complexa;
interval	tipo intervalo;
ivector	vetor de intervalos;
imatrix	matrix de intervalos;
cinterval	tipo intervalo complexo;
civector	vetor de intervalos complexos;
cimatrix	matrix de intervalos complexos;

Fig. 2.1 Tipos de dados numéricos do PASCAL-XSC

Um grande número de operadores são pré-definidos para estes tipos de dados nos módulos aritméticos do PASCAL-XSC (veja na seção 2.8). Todos estes operadores proporcionam resultados com máxima exatidão.

Comparando com o Pascal Padrão, existem 11 novos símbolos de operadores. Estes são os operadores $^{\circ}<$ e $^{\circ}>$, onde $^{\circ} \in \{+, -, *, /\}$ para as operações com arredondamento direcionado para baixo e para cima e, os operadores $**$, $+^*$ e $><$ necessários em cálculos intervalares para a intersecção, união e para o teste da desconectividade (intersecção vazia).

Tabela 2.1 Tabela dos operadores aritméticos pré-definidos

\right left \	integer real complex	interval cinterval	rvector cvector	ivector civector	rmatrix cmatrix	imatrix cimatrix
monadic	+, -	+, -	+, -	+, -	+, -	+, -
interger real complex	°, °<, °>, +*	+, -, *, / +*	*, *<, *>	*	*, *<, *>	*
interval cinterval	+, -, *, / +*	+, -, *, / +*, **	*	*	*	*
rvector cvector	*, *<, *> /, /<, />	*, /	°, °<, °>, +*	+, -, *, +*		
ivector civector	*, /	*, /	+, -, *, +*	+, -, *, +*, **		
rmatrix cmatrix	*, *<, *> /, /<, />	*, /	*, *<, *>	*	°, °<, °>, +*	+* +, -, *
imatrix cimatrix	*, /	*, /	*	*	+* +, -, *	+*, **, +, -, *,

OBS:

- As operações da linha monadic não tem operando esquerdo;
- A operação ° pertence ao conjunto {+,-,*,/} na linha dos tipos inteiro, real e complexo;
- A operação ° pertence ao conjunto {+,-,*} na linha dos tipos rvector, cvector, rmatrix e cmatrix.
- A operação * denota produto escalar ou de matrizes na linha de vetores e matrizes;
- A operação +* indica união;
- A operação ** indica intersecção.

Nas tabelas 2.1 e 2.2 mostram todos os operadores aritméticos e relacionais pré-definidos em relação com todas as possíveis combinações de tipos de operandos.

Tabela 2.2 Tabela dos operadores relacionais pré-definidos

\right left \	integer real complex	interval cinterval	rvector cvector	ivector civector	rmatrix cmatrix	imatrix cimatrix
interger real complex	=, <> <=, <, >=, >	in =, <>				
interval cinterval	=, <>	in, ><, =, <> <=, <, >=, >				
rvector cvector			=, <> <=, <, >=, >	in, =, <>		
ivector civector			=,<>	in, ><, =, <> <=, <, >=, >		
rmatrix cmatrix					=, <> <=, <, >=, >	in =, <>
imatrix cimatrix					=,<>	in, >< =, <> <=, <, >=, >

OBS:

- Os operadores <= e < denotam a relação de subconjunto (contido) , e os operadores >= e > denotam a relação de superconjunto (conter).
- O operador >< indica o teste de desconectividade de intervalos;
- O operador in indica o teste se um ponto pertence ao intervalo ou o teste se um intervalo está estritamente contido no interior do intervalo.

Comparado com o Pascal Padrão, o PASCAL-XSC proporciona um largo conjunto de funções elementares matemáticas (veja a tabela 2.3). Estas funções estão disponíveis para os tipos *real, complex, interval* e *cinterval* com um nome genérico e proporciona um resultado com máxima exatidão. As funções para os tipos *complex, interval* e *cinterval* são providenciados em módulos aritméticos do PASCAL-XSC.

Tabela 2.3 Tabela das Funções matemáticas pré-definidas

	Função	Nome genérico
1	Valor Absoluto	abs
2	Arc Cosseno	arccos
3	Arc Cotangente	arccot
4	Inverso Cosseno Hiperbólico	arcosh
5	Inverso Cotangente Hiperbólico	arcoth
6	Arc Seno	arcsin
7	Arc Tangente	arctan
8	Inverso Seno Hiperbólico	arsinh
9	Inverso Tangente Hiperbólico	artanh
10	Cosseno	cos
11	Cotangente	cot
12	Cosseno Hiperbólico	cosh
13	Cotangente Hiperbólico	coth
14	Função Exponencial	exp
15	Função Potência (Base 2)	exp2
16	Função Potência (Base 10)	exp10
17	Logaritmo Natural (Base e)	ln
18	Logaritmo (Base 2)	log2
19	Logaritmo (Base 10)	log10
20	Seno	sin
21	Seno Hiperbólico	sinh
22	Quadrado	sqr
23	Raiz Quadrada	sqrt
24	Tangente	tan
25	Tangente Hiperbólico	tanh

Além das funções elementares matemáticas, o PASCAL-XSC providência funções de transferência, enumeradas na figura.2.2, entre tipos de dados necessários para a conversão de tipos de dados numéricos como intervalos e complexos, incluindo tipos escalares e *arrays*.

```
intval (ra,rb: real): interval;
inf (a: interval): real;
sup (a: interval): real;
compl (ra,rb:real): complex;
re (a: complex): real;
im (a: complex): real;
```

Fig. 2.2 Funções de transferência

2.2 O Conceito de operador geral

Por um simples exemplo de uma soma de intervalos, as vantagens do conceito de operador geral podem ser demonstradas. Na concepção da definição de operador pelo usuário, existem duas formas de implementar operações entre intervalos como por exemplo a adição de duas variáveis do tipo intervalo. As variáveis intervalares são declaradas como do tipo intervalo, ou seja,

```
type interval = record inf, sup: real;
```

Estas formas seriam a implementação por rotina ou por função. Na implementação por rotina, ou seja, pela declaração de uma rotina, onde os operadores com arredondamento direcionado (+< e +>, os quais não são disponíveis no Pascal padrão) têm de ser programados por uma rotina (procedure) como na figura 2.3.

```
procedure intadd (a,b: interval; var c: interval);
begin
  c.inf := a.inf +< b.inf;;
  c.sup := a.sup +> b.sup;
end;
```

Fig. 2.3 Adição de intervalos por rotinas

Neste caso a expressão matemática z := a+b+c+d, seria implementada por três comandos de programação, ou seja, três chamadas da rotina da figura 2.3: intadd(a,b,z); intadd(z,c,z); intadd(z,d,z).

No caso da implementação através da declaração de uma função (somente é possível em Pascal XSC, pois este tipo de declaração não é possível no Pascal padrão), como mostra a figura 2.4, a mesma expressão matemática z:= a+b+c+d, seria expressa no uso recursivo da função, ou seja: z:= intadd(intadd(intadd(a,b),c),d);.

```
function intadd(a,b: interval): interval;
begin
  intadd.inf:= a.inf +< b.inf;
  intadd.sup:= a.sup +> b.sup;
end;
```

Fig. 2.4 Adição de intervalos por função

Em outros casos, a transcrição da fórmula matemática ficou um pouco complicada. Em comparação, se um operador é implementado em Pascal XSC, como mostra a figura 2.5, a expressão matemática seria programada pelo comando z:=a+b+c+d; ou seja, a notação matemática usual.

```
operator + (a,b:interval) intadd: interval;
begin
  intadd.inf:= a.inf +< b.inf;
  intadd.sup:= a.sup +> b.sup;
end;
```

Fig. 2.5 Adição de intervalos pela definição de operador

Além da possibilidade da sobrecarga dos símbolos de operadores, esta forma admite o uso do mesmo nome de operador. A declaração de tais tipos de operadores necessita ser precedida por uma declaração de prioridade. Existem quatro níveis diferentes de prioridade, cada uma representada por um símbolo como mostra a figura 2.6.

unária (monadic)	↑	nível 3 (prioridade mais alta)
multiplicativa	*	nível 2
aditiva	+	nível 1
relacional	=	nível 0

Fig. 2.6 Níveis de prioridade

Por exemplo, um operador para o cálculo do coeficiente binomial $\binom{n}{k}$ pode ser definido como na figura 2.7, onde a avaliação matemática corresponderia ao comando c:=n choose k.

```
priority choose = *; {priority declaration}
operator choose (n,k: integer) binomial: integer;
var i,r: integer;
begin
  if k > n div 2 then k := n-k;
  r := 1;
  for i:=1 to k do
        r := r* (n-i +1) div i;
  binomial:= r;
end;
```

Fig. 2.7 Definição do operador para cálculo do coeficiente binomial

O conceito de operador desenvolvido em Pascal XSC oferece a possibilidade da definição de um número arbitrário de operadores, ou a sobrecarga de símbolos de operadores ou nomes arbitrários para operadores e a implementação recursiva da definição de operadores.

Em Pascal XSC, também é permitida a possibilidade da sobrecarga de atribuição := para permitir a notação natural para atribuições.

Exemplo 1: Uso da definição de atribuição para complexos.

```
var
  c: complex;
  r: real;

operator := (var c:complex; r:real);
begin
  c.re:= r;
  c.im:= 0;
end;

r := 1.5;
c:= r; {número complexo < 1.5 , 0.0>}
```

2.3 *Overloading* de subrotinas

O Pascal Padrão providencia as funções matemáticas elementares *sin, cos, arctan, exp, ln, sqr* e *sqrt* somente para números reais. De forma a implementar a função seno para argumentos intervalos, um novo nome de função como *isin(..)* necessita ser usado, porque o *overloading* ou a sobrecarga do nome *sin* da função elementar seno não é permitida no Pascal Padrão. Entretanto, Pascal XSC admite sobrecarga de nomes de funções e rotinas, através da introdução do conceito de símbolo genérico na linguagem.

Assim os símbolos *sin, cos, arctan, exp, ln, sqr* e *sqrt* podem ser utilizados não somente para argumentos do tipo real, mas também para intervalos, números complexos e outros tipos. Para distinguir entre diversas funções e rotinas com o mesmo nome, o número e o tipo dos argumentos utilizados são o que determina o método, pois a rotina utilizada será do tipo do argumento. O tipo do resultado, entretanto, não é usado. Por exemplo, a rotina de rotação *rotate* pode ser definida para os tipos real, complexo e intervalo.

```
procedure rotate (var a,b: real);
procedure rotate (var a,b,c: complex);
procedure rotate (var a,b,c: interval);
```

O conceito de overloading também é aplicado as rotinas padrões de leitura (*read*) e escrita (*write*) de uma forma levemente modificada. O primeiro parâmetro da nova declaração da rotina de entrada ou saída necessita ser um parâmetro modificável (precedido pela palavra **var**) do tipo arquivo e, o segundo parâmetro representa a quantidade que deve entrar ou sair. Todos os demais parâmetros são interpretados como formato de especificação. Um exemplo é dado na figura 2.8, onde se tem uma rotina de escrita de complexo.

```
procedure write (var f: text; c: complex; w: integer);
begin
  write (f, '(', c.re:w, ',' , c.im:w, ')');
end;
```

Fig. 2.8 Rotina de impressão de complexos

Quando se chama a carga de uma rotina de entrada e saída, o parâmetro de arquivo pode ser omitido, pois corresponde a uma chamada com os arquivos padrões de input ou output. O formato dos parâmetros precisa ser introduzido e separado por vírgulas. Entretanto, vários comandos de entrada ou saída podem ser combinados em um único comando, assim como no Pascal Padrão, como no exemplo a seguir, onde r é um real, c um complexo e o comando é: write (r:10, c:5, r/5);.

2.4 O conceito de módulo

O Pascal Padrão assume basicamente que um programa consiste em um simples programa texto, o qual precisa ser preparado completamente antes que ele possa ser compilado e executado. Em muitos casos, é mais conveniente preparar um programa em várias partes, chamadas módulos, os quais podem então ser desenvolvidos e compilados independentes uns dos outros. Ainda mais, vários outros programas podem usar os componentes de um módulo sem ter que copiá-lo para dentro de seu código fonte ou recompilá-lo.

Para este propósito, um conceito de módulo foi introduzido no Pascal XSC. Este novo conceito oferece as possibilidades da programação modular, a verificação sintática e análise semântica além dos limites dos módulos e a implementação de pacotes aritméticos como módulos padrões.

Um módulo é introduzido pela palavra-chave **module** seguida por um nome e um ponto e vírgula. Seu corpo é bem similar à estrutura de um programa normal, com excessão que a palavra símbolo **global** pode ser usada diretamente na frente das palavras-chave de declaração de constante (**const**), tipo (**type**), variável (**var**), rotina (**procedure**), função (**function**) e operador (**operator**) e diretamente depois da palavra símbolo **use** e do sinal de igual na declaração de tipo.

Portanto, é possível declarar tipos privados assim como tipos não privados. A estrutura interna de um tipo privado não é conhecida fora da declaração do módulo. Objetos deste tipo privado podem somente ser usados e manipulados através de rotinas, funções e operadores definidos pela declaração do módulo.

Para módulos importados com **use** ou **use global**, as seguintes regras de transitividade são verdadeiras: Se M1 **use** M2 e M2 **use global** M3, então M1 **use** M3. Mas se M1 **use** M2 e M2 **use** M3, não implica que M1 **use** M3.

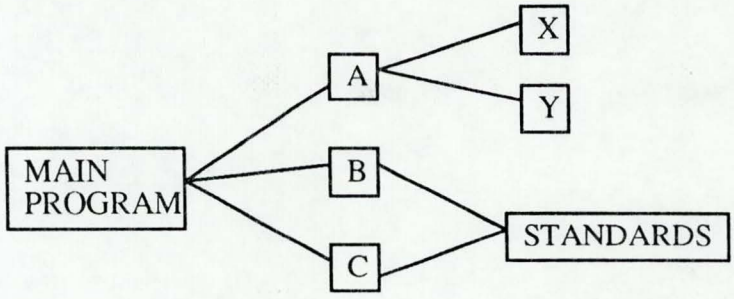

Fig. 2.9 Exemplo de Hierarquia de módulos

No exemplo da figura 2.9, todos objetos globais dos módulos A, B e C são visíveis no programa principal, mas ele não tem acesso aos objetos globais dos módulos X, Y e STANDARDS.

2.5 *Arrays* dinâmicos

No Pascal Padrão não há maneira de declarar tipos ou variáveis dinâmicas. A única maneira de gerenciar dinamicamente a memória no Pascal Padrão é através da alocação e liberação de objetos de tamanho fixo, os quais são referenciados por ponteiros (*pointers*).

Por exemplo, pacotes com operações de vetores e matrizes são tipicamente implementadas com dimensões fixas (máxima). Por esta razão, somente parte da memória alocada é usada, se o usuário estiver resolvendo um problema de dimensão menor do que a definida. O conceito de *array* dinâmico acaba com esta limitação.

Este novo conceito pode ser descrito pelas características de ser dinâmico em rotinas e funções, ter alocação automática e liberação das variáveis locais dinâmicas, utilizar economicamente o espaço de armazenamento, ter acesso a linhas e colunas de *arrays* dinâmicos e ter compatibilidade entre os tipos estáticos e dinâmicos de *arrays*.

Arrays dinâmicos precisam ser marcados com a palavra-chave **dynamic**. A grande desvantagem de esquemas de *arrays* "pré-formados", disponíveis no Pascal Padrão, é que eles podem ser acessados somente por parâmetros que sejam índices, não sendo permitidos parâmetros que sejam variáveis ou resultados de funções. Portanto, esta característica não é plenamente dinâmica.

No Pascal XSC, *arrays* dinâmicos e estáticos podem ser usados da mesma forma. Entretanto, *arrays* dinâmicos não podem ter componentes de outras estruturas de dados, por exemplo *records*.

A declaração do tipo *array* dinâmico bi-dimensional pode ser feita por:

```
type matrix = dynamic array [*,*] of real;
```

Também é possivel definir diferentes tipos de dinâmicos, com correspondentes estruturas sintáticas. Por exemplo, pode ser útil, em algumas situações, identificar os coeficientes de um polinômio como componentes de um vetor ou vice-versa. Como Pascal é uma linguagem estritamente orientada a tipos de dados, esta equivalência de estrutura de *arrays* necessita ser combinada através de uma adaptação prévia. O exemplo da figura 2.10 mostra a definição de um polinômio e de um vetor, onde as funções de conversão de tipo (*polynomial(..) e vector(..)*) são definidas implicitamente. Acessar o menor e o maior índice de cada dimensão é possível pela função *lbound(..)* e *ubound(..)* ou suas abreviaturas *lb(..)* e *ub(..)* respectivamente.

```
type vector = dynamic array[*] of real;
type polynomial = dynamic array[*] of real;
operator + (a,b: vector) res:vector[lb(a)..ub(a)];
var i:integer;
begin
  for i:=lb(a) to ub(a) do
        res:= a[i] + b[lb(b) +i -lb(b)];
end;

var v: vector [1..n];
    p: polynomial [0..n-1];

v:=vector(p);
p:= polynomial(v);
v:=v+v;
v:=vector(p) + v;  {mas não v:= p+v;}
```

Fig. 2.10 Exemplo de dinâmicos com estruturas correspondentes

Além de acessar cada componente da variável, Pascal XSC oferece a possibilidade de acesso a *subarrays*. Se uma componente da variável contém um * ou um intervalo de uma expressão de índice, refere-se ao *subarray* com um todo ou um específico intervalo de índice na correspondente dimensão. Por exemplo, M[1..2,j] é o *array* constituído do primeiro e segundo da j-ésima coluna do *array* bi-dimensional M. Este exemplo demonstra o acesso a linhas e colunas do *array* dinâmico.

```
type vector = dynamic array[*] of real;
type matrix = dynamic array[*] of vector;
```

```
var v: vector [1..n];
    m: matrix[1..n, 1..n];

 v:= m[i];
 m[i]:= vector(m[*,j]);
```

Na primeira atribuição não é necessário usar a função de conversão de tipos, tanto do lado direito como do lado esquerdo do conhecido tipo dinâmico. Um caso diferente é mostrado na segunda atribuição. O lado esquerdo é um tipo dinâmico conhecido, mas do lado direito é um tipo dinâmico anônimo, então é necessário o uso da função intrínseca de conversão *vector(..)*.

2.6 Expressões exatas

A teoria da aritmética computacional requer a implementação do produto escalar com somente um arredondamento de acordo com a seguinte definição, dada originalmente em [BOH93].

> "Dados dois vetores x e y com n componentes cada, em ponto flutuante, e um pré-estabelecido modo de arredondamento $\square$, o resultado s da operação produto escalar em ponto flutuante (aplicado em x e y) é definido por :
>
> $$s:= \square(\bar{s}):= \square(x.y) = \square\left(\sum_{i=1}^{n} xi * yi\right), \quad n \geq 1$$
>
> onde todas operações aritméticas são matematicamente exatas. Então s pode ser calculado como se fosse um resultado correto intermediário $\bar{s}$ em precisão infinita e com intervalo do expoente não limitado na primeira rotina e, então arredondado para o formato desejado de ponto flutuante de acordo com o modo de arredondamento $\square$."

Portanto o resultado da operação precisa ser o resultado correto do produto escalar com somente uma aplicação do arredondamento no final.

A implementação de algoritmos de inclusão com verificação automática do resultado ou validação (ver [KAU87, KUL89, KUL88, ULL90]) faz um uso extensivo da avaliação exata do produto escalar. Para avaliar este tipo de expressão foi introduzido um novo tipo de dado ***dotprecision***. Variáveis do tipo ***dotprecision*** podem assumir qualquer valor possível resultante da avaliação expressões de produto escalar sem perda de exatidão (como é mostrado em [KUL81, KUL89]). Baseado neste tipo de dados, as assim chamadas **expressões exatas** (*#-expressions*), podem ser formuladas por um símbolo exato (#, #*, #<, #>, ou ##) seguido por uma expressão exata entre parentesis. A expressão exata precisa ter a forma de expressão do produto escalar na estrutura escalar, vetorial ou matricial, e é avaliada sem qualquer erro de arredondamento. Por causa disto, o resultado de uma expressão exata tem um erro de no máximo 1 *ulp*, isto é, de no

máximo uma unidade na última casa da mantissa (***unit in the last mantissa place***). As tabelas 2.5 e 2.6 dão uma visão geral das possíveis expressões exatas com a exatidão das expressões (mais detalhes podem ser encontrados em [HAM92]).

A sintaxe das expressões exatas é da forma: #<símbolo> (expressão exata). Para obter resultados não arredondados ou corretamente arredondados de uma expressão de produto escalar, o usuário precisa escrever a expressão entre parêntesis precedida pelo símbolo #, podendo, opcionalmente, ser seguido por um símbolo de modo de arredondamento. A tabela 2.4 mostra os possíveis modos de arredondamento com respeito à forma da expressão do produto escalar.

Tabela 2.4 Tabela dos modos de arredondamento para expressões exatas

Símbolo	forma da expressão	Modo de arredondamento	Simb.Mat
#*	scalar, vector, matrix	simétrico	□
#<	scalar, vector, matrix	para baixo	∇
#>	scalar, vector, matrix	para cima	Δ
##	scalar, vector, matrix	menor intervalo contido	◊
#	somente scalar	exato sem arredondamento	

Na prática, expressões de produto escalar podem conter um grande número de termos resultando numa notação implícita muito incômoda. Para aliviar esta dificuldade matemática, o símbolo somatório $\sum$ é usado. Se por exemplo A e B são matrizes n-dimensionais, então a avaliação de d:= $\sum$ Aik . Bkj, para k=1 até n, representa uma expressão de produto escalar. O Pascal XSC providencia uma notação equivalente para este propósito, o comando ***sum***, sendo a expressão correspondente dada a seguir, onde d é uma variável do tipo ***dotprecision***.

d:= **#(for** k:=1 **to** n **sum** (A[i,k]*B[k,j])),

Expressões produto escalar ou expressões exatas são usadas principalmente no cálculo do resíduo (ou erro). No caso de sistemas de equações lineares Ax = b, onde $A \in \mathbf{R}^{nxn}$, $x,b \in \mathbf{R}^n$, $Ay \approx b$ é tomado como exemplo. Então a inclusão do erro é dada por ◊(b-Ay), o qual pode ser calculado em Pascal XSC pela fórmula ##(b-Ay), com somente uma operação intervalar de arredondamento para cada componente do vetor intervalar resultante. Para se ter inclusões verificadas em sistemas de equações lineares é necessário avaliar a expressão do erro ◊(E-RA), onde $R \approx A^{-1}$ e E é a matriz identidade. Em Pascal XSC esta expressão pode ser programada por **##(id(A)-R*A);** onde uma matriz intervalar é calculada com somente um arredondamento por componente. A função **id**(..) é definida no módulo da aritmética real de vetores e matrizes e produz uma matriz identidade do mesmo tipo dos argumentos.

Tabela 2.5 Tabela das expressões exatas reais

#-símbolo	tipo resultante	operandos permitidos na expressão exata
#	dotprecision	• variáveis, constantes e chamadas de funções especiais do tipo integer, real e dotprecision; • produtos do tipo integer e real; • produto escalar do tipo real.
#*	real	• variáveis, constantes e chamadas de funções especiais do tipo integer, real e dotprecision; • produtos do tipo integer e real; • produto escalar do tipo real.
#<	complex	• variáveis, constantes e chamadas de funções especiais do tipo integer, real,complex e dotprecision; • produtos do tipo integer, real e complex; • produto escalar do tipo real e complex.
#>	rvector	• Variáveis e chamadas de funções especiais do tipo rvector; • Produtos do tipo rvector (rmatrix.rvector, real.rvector).
	cvector	• Variáveis e chamadas de funções especiais do tipo rvector ou cvector; • Produtos do tipo rvector ou cvector (cmatrix.rvector, real.cvector).
	rmatrix	• Variáveis e chamadas de funções especiais do tipo rmatrix; • Produtos do tipo rmatrix..
	cmatrix	• Variáveis e chamadas de funções especiais do tipo rmatrix ou cmatrix; • Produtos do tipo rmatrix ou cmatrix.

Tabela 2.6 Tabela das expressões exatas intervalares e complexas

#-símbolo	tipo resultante	operandos permitidos na expressão exata
##	interval	• variáveis, constantes e chamadas de funções especiais do tipo integer, real, interval ou dotprecision; • produtos do tipo integer, real ou interval; • produto escalar do tipo real ou interval.
	cinterval	• variáveis, constantes e chamadas de funções especiais do tipo integer, real,complex, interval, cinterval e dotprecision; • produtos do tipo integer, real, complex, interval e cinterval; • produto escalar do tipo real, complex, interval e cinterval..
	ivector	• Variáveis e chamadas de funções especiais do tipo rvector ou ivector. • Produtos do tipo rvector ou ivector.
	civector	• Variáveis e chamadas de funções especiais do tipo rvector, cvector, ivector ou civector; • Produtos do tipo rvector, cvector, ivector ou civector.
	imatrix	• Variáveis e chamadas de funções especiais do tipo rmatrix ou imatrix; • Produtos do tipo rmatrix ou imatrix.
	cimatrix	• Variáveis e chamadas de funções especiais do tipo rmatrix, cmatrix, imatrix ou cimatrix; • Produtos do tipo rmatrix, cmatrix, imatrix ou cimatrix;.

2.7 O conceito de *string*

As ferramentas para tratamento de seqüências (string) de caracteres no Pascal Padrão não admitem um processamento de texto conveniente. Por esta razão, o conceito de seqüência foi integrado na definição da linguagem Pascal XSC, a qual admite um tratamento conveniente de informações textuais, e usam o conceito de operador, mesmo em computação simbólica. No novo tipo de dado *string*, o usuário pode trabalhar com seqüências de até MAXINT caracteres. Quando declarado variáveis do tipo *string*, o usuário pode especificar o tamanho máximo da seqüência menor que MAXINT. Portanto uma seqüência s que pode ter até 40 caracteres é declarada por: **var** s: **string**[40];.

As operações padrões disponíveis ao tipo *string* são: concatenação; tamanho atual; conversões *string* para *real, string* para *integer, real* para *string* e *integer* para *string*; retirada de *substring*; posição da primeira aparição; operadores relacionais <=, <, >=, >, <>, = e *in*.

2.8 Módulos Padrões

Os módulos disponíveis são: aritmética intervalar (I_ARI), aritmética complexa (C_ARI), aritmética intervalar complexa (CI_ARI), aritmética real matricial/vetorial (MV_ARI), aritmética intervalar matricial/vetorial (MVI_ARI), aritmética complexa matricial/vetorial (MVC_ARI) e aritmética intervalar complexa matricial/vetorial (MVCI_ARI). Todos estes módulos podem ser encorporados através do comando **use** descrito na seção 2.4. Como um exemplo, a tabela 2.7 apresenta os operadores providos pelo módulo da aritmética intervalar matricial e vetorial.

Tabela 2.7 Tabela do operadores do módulo MVI_ARI

\right left \	integer real	interval	rvector	ivector	rmatrix	imatrix
monadic				+, -		+, -
interger real				*		*
interval			*	*	*	*
rvector		*, /	+* +, -, * in, =, <>			
ivector	*, /	*, /	+* +, -, * =, <>	+*, **, +, -, *, in, =, <>, > <=, <, >,>=		
rmatrix		*, /			+*	+* +, -, * in, =, <>
imatrix	*, /	*, /	*		+* +, -, * =, <>	+*, **, +, -, *, in, =, <>, > <=, <, >,>=

Além destes operadores, o módulo MVI_ARI providencia os operadores, funções e rotinas genéricas denominadas *intval, inf, sup, diam, mid, blow, transp, null, id read* e *write*. A função *intval* é usada para gerar vetores e matrizes e intervalos, onde *inf* e *sup* são funções que selecionam o ínfimo e supremo do intervalo em questão. O diâmetro e o ponto médio do vetor ou matriz intervalar pode ser calculado por *diam* e *mid*. O comando *blow* gera uma inflação no intervalo e *transp* gera a transposta da matriz. Vetores e matrizes nulas são geradas pela função *null*, enquanto que *id* produz uma matriz identidade do tipo apropriado. Finalmente, existem rotinas genéricas de entrada e saída, *read* e *write*, que se pode usar com todos os tipos de vetores e matrizes definidos nos módulos mencionados aqui.

2.9 Rotinas para solução de problemas

Rotinas para a resolução de problemas numéricos comuns foram implementados em Pascal XSC. Os métodos aplicados calculam inclusões de alta exatidão da solução verdadeira do problema e, ao mesmo tempo, provam a existência e unicidade da solução no intervalo calculado. As vantagens deste tipo de rotinas são:

- A solução é calculada com alta ou máxima, mas sempre com uma exatidão controlada, mesmo nos casos mal condicionados;

- A corretude do resultado é verificada automaticamente, isto é, um conjunto de inclusões é calculado de forma a garantir a existência e unicidade da solução verdadeira contida neste intervalo;

- Se não existe solução ou se o problema é extremamente mal condicionado, uma mensagem de erro é provida.

As rotinas implementadas disponíveis, abrangem as áreas de sistemas de equações lineares (sistemas densos, inversão de matrizes, problemas dos mínimos quadrados e cálculo de pseudo inversas, matrizes reais bandas e esparsas); avaliação de polinômios de uma variável (real, complexa, intervalar ou complexa intervalar) ou de várias variáveis reais; cálculo de zeros de polinômios; autovalores e autovetores de matrizes reais simétricas ou de matrizes arbitrárias reais, complexas, intervalares ou complexas intervalares; equações diferenciais ordinárias (problema de contorno

lineares; quadratura numérica; equações integrais, diferenciação automática e otimização.

3 VERSÕES DO PASCAL XSC

A linguagem Pascal XSC extende a linguagem Pascal SC ([BOH86, BOH87, NEA84]). Ambas as linguagens foram definidas e desenvolvidas no Instituto de Matemática Aplicada da Universidade de Karlsruhe. O primeiro compilador Pascal SC foi implementado para processadores Z80 em 1980. Devido à pouca memória da máquina Zilog, um interpretador foi usado, o que diminui o tempo de execução. Este compilador foi transportado para máquinas DOS no início dos anos 80. Três anos mais tarde um compilador Pascal SC gerando código de máquina para processadores Motorola-68000 foi desenvolvido ([KUL87]). Este sistema é muito mais rápido, mas perdeu em portabilidade, pois rodava apenas em processadores Motorola-68000. O novo sistema Pascal XSC está agora disponível para computadores pessoais, estações de trabalho, mainframes e supercomputadores pelo uso do compilador portável que traduz para ANSI C.

O principal objetivo do sistema é a portabilidade. Para este propósito, ele necessita providenciar uma fácil portabilidade do compilador e do sistema de execução; permitir a necessária reconfiguração do compilador para cada novo computador; permitir a portabilidade do código gerado através da compilação cruzada; e providenciar a consistência dos resultados para todas as instalações.

A linguagem ANSI C foi escolhida como linguagem de implementação e linguagem de configuração. A principal razão para esta escolha foi a grande variedade de computadores onde uma ou mais linguagens C estão disponíveis. Além do que a linguagem C permite a programação de código portável. A linguagem ANSI C padrão estimularia os fabricantes de compiladores C a construir compiladores que correspondam ao padrão e os estimularia a unificar os compiladores existentes. Isto facilita a portabilidade. Opções especiais do compilador existem para providenciar compilação cruzada. A linguagem C é altamente modular. Pequenos gerenciamentos para chamadas de funções resultam em alta eficiência do objetivo do código.

Há grandes diferenças semânticas entre Pascal XSC e a linguagem C. Visto que Pascal XSC permite expressões *dotprecision*, aninhamento de declaração de subrotinas, sobrecarga de subrotinas, *arrays* dinâmicos e *subarrays*, conjuntos e sequências, isto torna necessário simular estes conceitos de código alvo. Parcialmente esta tarefa pode ser resolvida pelo uso de funções apropriadas nas bibliotecas de tempo real, mas alguns problemas, assim como a simulação de rotinas aninhadas, têm de ser resolvidos dentro do compilador.

3.1 Aritmética Real Diferente

Uma característica especial do novo compilador é que as operações básicas da aritmética real são modificadas para suportar diferentes aplicações, as quais podem requerer diferentes propriedades aritméticas, como portabilidade, velocidade e exatidão. Detalhes são dados em [COR91].

As aritméticas suportadas são:

- A que por software emula a aritmética padrão IEEE 754. Uma simulação completa da aritmética de dupla precisão do padrão binário de ponto flutuante via software. Todos os requisitos do padrão são cumpridos, incluindo arredondamentos direcionados, infinitos e excessões. Não é requerida nenhuma prioridade especial de hardware ou do sistema de execução da linguagem C.

- A aritmética de hardware do computador em uso. As operações aritméticas são garantidas pelo sistema de execução da linguagem C. O formato dos dados e a exatidão necessária das operações não satisfaz necessariamente o padrão IEEE 754. Esta aritmética é projetada para ser usada por programas por serem rápidas.

- A aritmética de múltipla precisão é pretendida para programas que implementem algoritmos numéricos de alta precisão. As operações aritméticas são baseadas em tipos especiais de dados de múltipla precisão. Variáveis deste tipo podem garantir valores com vários dígitos na mantissa durante a execução do programa.

- Uma aritmética definida pelo usuário. A aritmética real padrão pode ser substituída por uma aritmética real definida pelo usuário de uma forma bem simples, como é mostrado em [ALL91, ALL92]. O usuário precisa garantir que todas as características definidas pelo padrão de aritmética real também estejam disponíveis nesta nova aritmética.

3.2 O sistema de desenvolvimento do Pascal XSC

O sistema Pascal XSC inclui: o gerenciador, o compilador Pascal XSC para C, gerador de listas, biblioteca de tempo real e o programa de configuração.

O principal propósito do gerenciador é fazer o ciclo de desenvolvimento de programas mais amigável e reduzir o número de erros acidentais. Isto é alcançado pela liberação do usuário do fornecimento de informações sobre convenções de diretórios e opções dos compiladores Pascal XSC e C e do *linkador* usado. O gerenciador oferece facilidades pela *linkagem* automática de todos os módulos de que o programa corrente depende.

Módulos podem ser compilados separadamente. As dependências e conexões dos módulos são verificadas pela interface consistente usada, arquivos associados com os módulos que são mencionados em cláusulas use. O número e tipo dos argumentos atuais são verificados em conformidade com os parâmetros formais. Um módulo pode importar outros módulos. Em geral, as dependências dos módulos em um programa executável podem ser descritas por um gráfico acíclico.

O compilador fornece uma relativa facilidade de verificação de erros, incluindo verificação léxicas, sintáticas e semânticas e, restauração de erros. Se um módulo é mudado, é bem natural que ele necessite ser recompilado antes que os módulos que o referenciem sejam compilados. O compilador verifica esta condição automaticamente, gerando uma mensagem de erro se o tempo de compatibilidade do módulo foi violado. O gerador de listas é chamado depois da compilação de um programa ou módulo que contenha erros. Ele produz uma lista de leitura com as mensagens de erro e aponta a posição exata do erro: a linha e coluna do erro. É possível corrigir erros pela edição da listagem. Então existe um programa que lê a listagem e reconstroi o arquivo fonte a partir da listagem.

Depois da instalação do compilador, o usuário pode mudar alguns sistemas de dependências, como rastro dos nomes e nomes dos tipos de arquivos assim como os valores definidos para as opções do compilador. Estas modificações são feitas pelo programa configurador.

3.3 O estado atual da implementação

A compatibilidade do compilador Pascal XSC com o Pascal Padrão foi testada usando o "Pascal validation suite" da Universidade da Tasmânia. Exastivos testes foram efetuados considerando diferentes extensões do Pascal XSC. O sistema é conhecido e usado por muitos estudantes para propósitos educacionais e para desenvolvimento de softwares.

Até agora o sistema Pascal XSC tem, com sucesso, sido instalado e passado por testes em muitos computadores como mostra a tabela 3.1. Em alguns sistemas a aritmética de hardware é garantida, possibilitando programas gerados mais velozmente que outros onde a aritmética de hardware está sendo desenvolvida, mas ainda não disponível.

A versão do IBM PC para o compilador C Zoetech suporta a geração de código de 32 bits e é construído numa estensão do DOS, a qual permite ultrapassar o limite de 640K de memória. A versão para estação de Trabalho Sun SPARC providencia acesso direto à aritmética de hardware quádrupla.

A versão do Pascal XSC em Transputers sobre o sistema operacional HELIOS combina conceitos do Pascal XSC de computação numérica real com as vantagens do processamento paralelo. Um módulo de comunicação para paralelização explícita, no nível

de tarefa foi desenvolvido, estendendo a linguagem por rotinas de comunicação, as quais habilitam a programação paralela em Pascal XSC de uma forma similar às linguagens de alto nível que rodam sobre HELIOS. As operações com vetores e matrizes de máxima exatidão são freqüentemente utilizadas, por isto elas podem ser incluídas na biblioteca de tempo real do Pascal XSC e, assim, serem chamadas automaticamente possibilitando inclusive a paralelização implícita em Pascal XSC.

Tabela 3.1 Tabela da disponibilidade do sistema PASCAL-XSC

Computador	Sistema operacional	Compilador C	Hardware support
IBM PC	DOS	Microsoft C 7.0	yes
IBM PC	DOS	Borland C++ 3.1	yes
IBM PC	DOS	Zortech C 3.0	yes
IBM PC	OS/2 2.0	IBM C set/2	
Sun SPARC Station	Sun OS 4.1 Solaris	System	yes
Micro VAX Station	ULTRIX	System	
DEC Station 3100	ULTRIX	System	
IBM PS/2	AIX	System	
IBM RS 6000	AIX	System	
HP 9000/300 Series	UNIX	System	
HP 9000/800 Series	UNIX	System	
HP 9000/700 Series	UNIX	System	yes
NeXT	UNIX	System	
CONVEX	UNIX	System	
VAX	VMS	System	
Transputer T800	HELIOS	System	
Data General	ULTRIX	System	

4 AMBIENTE E INSTALAÇÃO DA LINGUAGEM

Neste capítulo é descrito o ambiente mínimo que suporta a execução do Pascal XSC. São dadas algumas dicas para a correta instalação do sistema de software, incluindo as alterações necessárias nos arquivos *autoexec.bat* e *config.sys*.

Cabe lembrar que o Pascal XSC é um pré-processador da linguagem C, existindo versões de Pascal XSC para diferentes compiladores C e para diferentes plataformas. O compilador C utilizado na instalação disponível no Instituto de Informática da UFRGS é do C GNU, também conhecido como C DJGPP.

Observa-se ainda que o Compilador Pascal-XSC é distribuído pela empresa alemã Numerik Software GmbH[1] e foi cedido para o Instituto de Informática da UFRGS para fins de estudo e pesquisa. Já o compilador C GNU é de domínio público. Ele está disponível na Rede, podendo ser importado através do utilitário *ftp*.

O Grupo de Matemática Computacional do Instituto de Informática da UFRGS, desenvolveu um programa de instalação do Pascal XSC e do C GNU, pois a versão recebida da Universidade de Karlsruhe e da Numerik Software só incluiu o compilador Pascal XSC, sem o compilador C GNU e sem as definições necessárias para a correta instalação. Para tanto são criados dois diretórios: o PXSC e o DJGPP, para o Pascal XSC e para o compilador C, respectivamente. Ambos contêm subdiretórios como serão descritos no ítem 4.2.

4.1 O ambiente

O compilador Pascal XSC está disponível para várias plataformas, como foi visto no capítulo anterior. Em todas elas os cálculos produzem os mesmos resultados, portanto a aritmética que se tornou disponível por esta linguagem é independente da plataforma.

A versão disponível no Instituto de Informática da UFRGS é para equipamento do tipo IBM-PC 486 ou IBM-PC 386 com co-processador aritmético 80387. São necessários 640 KB de memória convencional e, pelo menos, 4 *Megabytes* de memória extendida. Para um melhor desempenho em termos de velocidade de processamento recomenda-se que o equipamento tenha 8 *Megabytes* de memória.

São necessários, ainda, um disco rígido de pelo menos 10 *Megabytes* de espaço disponível, drive de 1.2 ou 1.44 *Megabytes*, um monitor monocromático VGA.

---

[1]Numerik Software Gmbh, POBox 2232 D-76492 Baden-Baden Germany FAX 0049721694418

O sistema operacional necessário para a instalação é o MS-DOS (ou compatível) versão 5.0 ou posterior.

O sistema Pascal XSC é composto pelos programas de configuração, pelo gerenciador, pelo compilador, pelos módulos padrões e por uma biblioteca de rotinas que compõem o sistema de execução de tempo real. O compilador não contém um gerador de código para uma máquina específica, ele produz um código C, conforme padrão C ANSI, na verdade ele é um pré-processador que converte para C. O códogo C é que, quando compilado gera o código objeto para a máquina específica.

4.2 A Instalação

Para a correta instalação do software Pascal XSC é necessário a criação de diretórios para os compiladores Pascal XSC e C e atualizar os arquivos autoexec.bat e config.sys.

O programa de instalação organizado pelo Grupo de Matemática Computacional do Instituto de Informática da UFRGS, está dividido em três etapas, ou seja: a instalação do módulo Pascal, a instalação do módulo C e as configurações, que incluem as alterações nos arquivos autoexec.bat e config.sys.

A primeira etapa, a que instala o módulo do compilador Pascal está contido em um disquete de $3^{1/2}$ polegadas. O programa de instalação cria os diretórios e descompacta os arquivos necessários à utilização do compilador.

A segunda etapa, a da instalação do módulo do compilador C, está contida em dois disquetes de $3^{1/2}$; segue as mesmas estapas do compilador Pascal, cria os diretórios necessários para o compilador e descompacta os arquivos correspondentes.

Após essas etapas de instalação é necessário reconfigurar os arquivos config.sys e autoexec.bat. As alterações são necessárias para se definir os caminhos, definir os dispositivos e expandir a memória de trabalho.

4.2.1 Instalação do módulo Pascal XSC

O módulo do Pascal XSC é denominado de PXSC. Ele possui três subdiretórios como mostra a figura 4.1. O conteúdo de cada subdiretório é:

- **\pxsc\bin**: neste diretório encontram-se o compilador e programas utilitários, como também o arquivo de configuração P88.ENV gerado pelo PXSCCFG.EXE para configurar o ambiente de trabalho da linguagem.

- **\pxsc\sys**: este diretório possui as bibliotecas (arquivos **.mod**) que são utilizadas pelo compilador durante o processo de compilação de um programa (cláusula **use**).

- **\pxsc\fontes**: diretório aonde estão os programas do usuário.

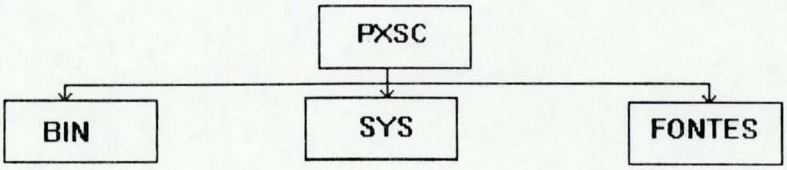

Fig. 4.1 Diretório do módulo Pascal XSC

Este diretório com seus subdiretórios são criados automaticamente pelo programa de instalação. O primeiro disquete contém o arquivo INSTALAP.BAT, três subdiretórios, onde se encontram os arquivos dos referidos subdiretórios compactados (zipados). Neste disquete se encontra também os arquivos: MUDACONF.TEX e MUDAUTO.TEX, que contém as modificações de configuração descritos no ítem 4.2.3. Existe ainda os programas de descompactam os arquivos, ou seja o programa pkunzip.exe.

A figura 4.2 mostra o conteúdo do arquivo INSTALAP.BAT.

O subdiretório **pxsc\sys** deve conter todos os programas executáveis do sistema do Pascal XSC, estes são:

- pxsc — Compilador Pascal XSC;
- mxsc — Gerenciador de *batch;*
- dxsc — Gerenciador Interativo;
- exsc — Gerador de listagem curta;
- psclist — Gerador de listagem longa;
- pxsccfg — Programa configurador;
- dismod — Descompilador para arquivos interface;
- splitmod — Programa para split um módulo Pascal XSC;
- mod2lib — Rotina do Shell para criar uma biblioteca a partir de um módulo;
- mvmod — Rotina do Shell para mover um módulo Pascal XSC.

Além dos programas executáveis, existem os arquivos de ajuda (.hlp), de inclusão, bibliotecas, módulos padrões compilados (.o, .h e .mod).

```
@echo off
cls
echo ********************************************************************************
echo Programa de Instalacao da linguagem PASCAL-XSC
echo UFRGS -    Instituto de Informatica
echo  Grupo de Matematica Computacional
echo
echo  Versão de:  Outubro de 1994
echo
echo  Nos arquivos MUDAUTO.TEX e MUDACONF.TEX presentes no  disquete,
echo  estao as configuracoes necessarias  a serem includidas/modificadas nos arquivos
echo  AUTOEXEC.BAT e CONFIG.SYS
echo  Aguarde...
echo ********************************************************************************
c:
cd\
md temp
md pxsc
cd pxsc
md bin
md sys
md fontes
copy b:\fontes\hello.p \pxsc\fontes
copy b:\sys\sys.zip    \pxsc\sys
copy b:\bin\bin.zip    \pxsc\bin
b:pkunzip c:\pxsc\sys\sys.zip c:\pxsc\sys
b:pkunzip c:\pxsc\bin\bin.zip c:\pxsc\bin
del c:\pxsc\sys\sys.zip
del c:\pxsc\bin\bin.zip
cd\
echo Instalacao do modulo PASCAL concluida
```

Fig 4.2 Arquivo INSTALAP.BAT

4.2.2 Instalação do módulo C GNU

O módulo do C GNU é denominado de DJGPP. Este compilador é de domínio público, sendo encontrado no diretório */pub/msdos/djgpp*, no endereço eletrônico *oak.oakland.edu*, podendo ser transferido pelo programa ftp. A versão descrita aqui, não é completa, só estão sendo instalados os arquivos necessários para a execução do Pascal XSC. Maiores detalhes sobre o compilador djgpp podem ser encontrados no diretório de documentação.

A configuração mínima para a execução do C inclui quatro subdiretórios, como ilustra a figura 4.3. Estes subdiretórios são:

- **\djgpp\bin**: o diretório possui programas utilitários e os necessários para a utilização do compilador C que compõem a linguagem.
- **\djgpp\include**: arquivos de cabeçalho (*.h).
- **\djgpp\include\sys**: arquivos de cabeçalho relativos ao sistema operacional.
- **\djgpp\lib**: bibliotecas necessárias a compilação C (arquivos .o e .a).
- **\djgpp\drivers**: arquivos que servem como drivers de vídeo (arquivos .grd).

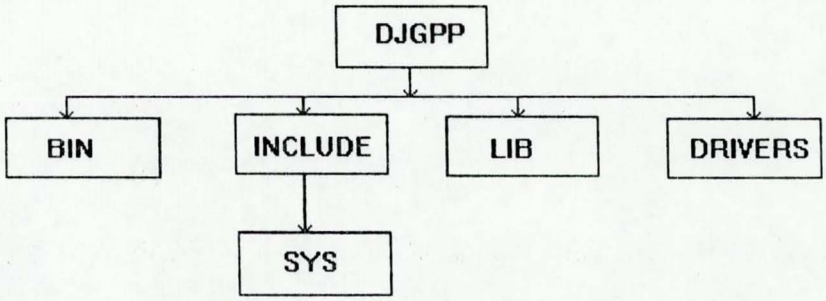

Fig. 4.3 Diretório do módulo DJGPP

Para a correta instalação do C GNU foi desenvolvido um arquivo de instalação, juntamente com os arquivos mínimos compactados. Este segundo módulo de instalação também cria os diretórios automaticamente e descompacta os arquivos, colocando-os nos devidos subdiretórios. Estes arquivos se encontram em dois disquetes. No primeiro disquete encontram-se os arquivos dos subdiretórios DRIVES, INCLUDE e LIB. No Segundo disquete está o arquivo compactado do diretório BIN. A figura 4.4 mostra o conteúdo do arquivo INSTALAC.BAT.

```
@echo off
cls
echo ******************************************************
echo  Instalacao do C GNU (módulo djgpp)
echo
echo
echo  Aguarde...
echo ******************************************************
c:
cd\
md djgpp
cd djgpp
md include
cd include
md sys
cd..
md lib
md drivers
md bin

copy b:pk*.exe
copy b:\include\includec.zip        c:\djgpp\include
copy b:\include\sys\sysc.zip        c:\djgpp\include\sys
copy b:\lib\libc.zip                c:\djgpp\lib
copy b:\drivers\driversc.zip        c:\djgpp\drivers

rem ***********************************************************
rem SOLICITA A TROCA DE DISQUETE PARA COMPLETAR COPIA- Disco 2
rem ***********************************************************
echo POR FAVOR INSIRA O OUTRO DISQUETE  - Disquete  djgpp 2 parte
pause
copy b:binc.zip                     c:\djgpp\bin

rem************************************************
rem  DESCOMPACTA ARQUIVOS
rem ************************************************
pkunzip c:\djgpp\include\includec.zip       c:\djgpp\include
pkunzip c:\djgpp\include\sys\sysc.zip       c:\djgpp\include\sys
pkunzip c:\djgpp\lib\libc.zip               c:\djgpp\lib
pkunzip c:\djgpp\drivers\driversc.zip       c:\djgpp\drivers
pkunzip c:\djgpp\bin\binc.zip               c:\djgpp\bin

del  c:\djgpp\include\includec.zip
del  c:\djgpp\include\sys\sysc.zip
del  c:\djgpp\lib\libc.zip
del  c:\djgpp\drivers\driversc.zip
del  c:\djgpp\bin\binc.zip
del pk*.exe
echo Instalacao concluida
```

Fig. 4.4 Arquivo INSTALAC.BAT

4.2.3 Configuração

A fase final da instalação do Pascal XSC é a da configuração, que compreende a inclusão de algumas definições nos arquivos do sistema. Esta definições atingem dois arquivos o *config.sys* e o *autoexec.bat*. Estes arquivos são ativados no momento da carga do sistema. Após a correção ou edição, deve-se recarregar o sistema por um *reset*.

No arquivo *config.sys* é necessário expandir a memória de trabalho para 4096. É aconselhável aumentar o número de files e de buffers, como mostra a figura 4.5. A linha de comando device define um dispositivo padrão que se encontra no diretório do sistema operacional DOS. Assumiu-se que exista um diretório em c:, denominado de DOS, onde estão incluídos os arquivos do sistema operacional DOS e uma cópia do arquivo *command.com*. Nesta caso, foi suposto que o rastro do diretório seja: c:\dos\.

```
REM******************************************************
REM Configuracoes a serem incluidas no arquivo CONFIG.SYS
REM******************************************************
device=c:\dos\ansi.sys
BUFFERS=30,0
FILES=100
shell=c:\dos\command.com c:\dos\ /e:4096 /p
```

Fig.4.5 Alterações no arquivo config.sys

O outro arquivo a ser alterado é o autoexec.bat. Devem ser incluídos a este arquivo, as informações para o sistema operacional localizar os diretórios e subdiretórios dos módulos PXSC e DJGPP. Estas informações são feitas na linha do comando PATH.

A seguir devem ser feitas algumas definições através do comando SET. São definidos onde os arquivos temporários devem ser armazenados, no caso é definido como diretório temporário o c;\temp. É definido, ainda, o editor a ser utilizado pelo sistema pxsc.

```
REM******************************************************
REM Configurações a serem incluidas no arquivo AUTOEXEC.BAT
REM******************************************************
PATH c:\djgpp\bin;c:\djgpp\sys;c:\pxsc\bin;c:\pxsc\sys;
set tmpdir=c:\temp
set go32tmp=c:\temp
set temp=c:\temp
set compiler_path=c:\djgpp\bin
set c_include_path=c:\djgpp\include
set library_path=c:\djgpp\lib
set go32=c:\dos\ansi
set pxsc_sys=c:\pxsc\sys\
set pxsc_edit=edit.com
```

Fig 4.6 Alterações no arquivo autoexec.bat

4.3 Considerações sobre a instalação

Nas alterações de configuração descritas no ítem 4.2.3, foi suposto que o equipamento era um do tipo IBM-PC 486DX, no qual já existe o co-processador aritmético incluído no próprio processador. Caso o equipamento seja um 386 com co-processador aritmético 80387, deve ser trocada a linha **set go32=c:\dos\ansi** pela nova linha, que indica e ativa o co-processador, **set go32=emu c:\djgpp\bin\emu387**

Se ao tentar compilar um programa, aparecer a mensagem **p88.env not found**, deve ser verificado se a variável pxsc_sys está corretamente configurada no arquivo autoexec.bat da seguinte maneira: **set pxsc_sys=c:\pxsc\sys**. Persistindo o erro, é provável que não tenha sido executado o programa PXSCCFG.EXE, que é o responsável pela configuração do ambiente da linguagem.

Na compilação do módulo C que compõe a linguagem, poderão aparecer dois erros: problemas nos drivers GO32.EXE e EMU387. No driver GO32.EXE deve ser verificado se a versão é 1.11. Não aparecendo a versão, o arquivo deve estar danificado.

Outra consideração importante está relacionada ao ambiente ser local ou de rede. Caso os compiladores estejam no servidor, então o rastro dos diretórios descrito na linha PATH de programa do autoexec.bat deve ser alterado, indicando o drive correto e a localização nele.

O mesmo deve ser atentado caso o diretório DOS não esteja na raiz e sim, seja um subdiretório, por exemplo, do diretório adm. Devem ser trocadas, neste caso, todas as referências de c:\dos\ para c:\adm\dos\.

O programa que realiza a configuração do ambiente de compilação, ajustando as opções de compilação, operação e a verificação de limites dos índices das variáveis subscritas (vetores e matrizes, por exemplo) é o *pxsccfg.exe*. A configuração fica disponível em um arquivo chamado *p88.env*. Através do programa de configuração pode-se definir algumas opções de compilação como é mostrado no capítulo 5.

Observa-se, por fim, que existem quatro variáveis de ambiente que devem ser definidas no arquivo autoexec.bat. Estas são: PXSC_SYS, PXSC_EDIT, PXSC_USR e PXSC_LIB. As duas primeiras constam entre as definições já vistas. A terceira define o ambiente do usuário, ela é de importância nos casos onde o sistema é isntalado no servidor da rede, portanto é necessário definir o ambiente específico do usuário. A última refere-se as opções de linkagem, arquivos objetos e localização das bibliotecas.

4.4 Opções do sistema de tempo real

O Pascal XSC tem um conjunto de rotinas que compõem o sistema de tempo real. Este sistema proporciona opções para depuração e documentação de programas. Neste item são descritas as opções que podem ser ativadas na instalação do Pascal XSC.

As opções do sistema de tempo real são dadas como argumentos na linha de comando. Estes argumentos contêm um símbolo especial na primeira posição da opção para diferenciar de nomes de arquivos. O símbolo padrão usado é o menos (-).

As opções de sistema de tempo real são:

constant conversion (-cc) Mostra uma mensagem de advertência no caso de uma conversão inexata de dados reais constantes ou valores reais introduzidos no formato decimal para a representação em número de ponto flutuante. Não são mostradas mensagens para conversões com arredondamentos direcionados.

IEEE trap handling (-ieee) Controlam as definições padrões para habilitar os tratadores de exceções. Cada uma das cinco exceções IEEE são caracterizadas por uma letra: ***d*** para divisão por zero, ***i*** para operação inválida, ***o*** para *overflow* do expoente, ***u*** para *underflow* de expoente e ***x*** para resultado inexato. Se uma ou mais letras forem seguidas ao cabeçalho ***-ieee*** da opção, o estado do tratador de exceção será modificado de habilitado para desabilitado, ou de desabilitado para habilitado. Por exemplo, a opção ***-ieeedu*** modifica os tratadores das exceções de divisão por zero e de *underflow* de expoente. O estado inicial de definiçao das cinco opções é habilitado para as três primeiras e desabilitado para as duas últimas. As modificações só são possíveis antes de rodar qualquer comando, pois fazem parte do módulo de inicialização.

runtime information (-info) Mostram informações sobre o sistema Pascal XSC e do programa corrente Pascal XSC sobre dispositivo padrão de saída *stdout*. A informação é tanto armazenada explicitamente no sistema de tempo real, como pode ser lida pelo arquivo padrão *info.txt*, o qual é residente no diretório do sistema Pascal XSC. O processamento do programa termina depois de mostrar todas as informações disponíveis. Após a habilitação da opção ***-info***, para se ter informações de auxílio pode-se utilizar: ?, -?, -help, -h, help ou h.

normalized Numbers (-nn) O valor determinado pelas rotinas predecessor (pred) e sucessor (succ) para números reais em ponto flutuante são normalizados.O padrão admite números desnormalizados.

program parameters (-pp) Mostra os nomes das variáveis do arquivo Pascal XSC, as quais são listadas na lista de parâmetros do comando ***program*** do programa ativo. O processamento do programa é finalizado após mostrar os nomes das variáveis na lista de parâmetros do programa.

parameter prompting (-pr) Habilita o prompt para nomes de arquivos externos para parâmetros do programa no início do processamento de um programa. Se existem menos nomes de arquivos que argumentos na linha de comando, então existem nomes de arquivos variáveis na lista de parâmetros do programa.

system directory (-sd) Mostra o caminho de um diretório fixo de sistema que é definido durante a instalação do sistema de tempo real. A sintaxe extendida da opção do sitema de tempo real ***-sd:***_path_ pode ser utilizada para alterar o caminho de um diretório fixo para o diretório _path_ em tempo de execução do programa. Os dois pontos que seguem o nome da opção serão ignorados pois não fazem parte do diretório _path_, que poderá ter até 63 caracteres.

signed zero (-sz) Habilita a geração do sinal de menos se o valor IEEE para o zero negativo for detectado em uma operação de saída.

trace brief (-tb) A saída produzida pelo habilitar a opção do sistema de tempo real ***-tr*** é reduzida no caso de funções recursivas se esta opção é usada.

no temporary files (-tf) Habilita o _prompt_ para nomes de arquivos para arquivos locais variáveis no _reset_ e _rewrite_ se não existir nenhum nome explicitamente definido como segundo argumento e nenhuma associação prévia a nomes de arquivos externos tenha sido feita.

trace (-tr) Habilita a criação de uma função _trace_ durante o processamento de um programa Pascal XSC, proporcionada se a opção do compilador ***+n*** estiver ativada. Qualquer informação disponível sobre entrada e saída em subrotinas, funções e operadores será mostrada no dispositivo padrão de erros _stderr_. O próximo nível de chamadas de subrotina, função ou operador será mostrado obedecendo uma identação. O número de linhas de saídas pode ser reduzido no caso de funções recursivas quando a opção do sistema de tempo real ***-tb*** também estiver habilitada.

user directory (-ud) Mostra o caminho do diretório do usuário que foi definido durante a instalação do sistema de tempo real. A sintaxe extendida da opção do sitema de tempo real ***-ud:***_path_ pode ser utilizada para modificar o diretório do usuário definido para o definido em _path_, que também pode ter até 63 caracteres.

version number (-vn) Mostra a identificação da versão do sistema de tempo real do Pascal XSC no dispositivo de saída padrão (stdout) antes do processamento do programa Pascal XSC iniciar.

Mais detalhes sobre as opões do sistema de tempo real podem ser encontrados em [COR91, KLA92].

5 SISTEMA PASCAL-XSC E A COMPILAÇÃO DE PROGRAMAS

Nesta seção é descrita a interface do sistema Pascal XSC, como editar, compilar e executar programas. Uma vez que se está trabalhando no ambiente do sistema DOS, a variável de ambiente PXSC_EDIT que define o editor de texto foi carregada com editor *edit* através da linha de comando no arquivo *autoexe.bat*, set pxsc_edit=c:\dos\edit.com.

5.1 Sistema de desenvolvimento - módulo interativo

Para ativar o compilador Pascal XSC de uma forma interativa, a qual permite vários ciclos de edição, compilação e execução, basta digitar ***dxsc***. Então aparecerá na tela a mensagem solicitando que seja introduzido o nome do arquivo (programa), pois o arquivo corrente está vazio como mostra a figura 5.1.

```
PASCAL-XSC to C  Development System

Currently empty  Filename
Enter new  Filename :
```

Fig 5.1 Tela Inicial do Sistema Pascal XSC Interativo

Como exemplo inicial, será tomado o programa *hello.p* descrito pela figura 5.2, o qual simplesmente imprime uma mensagem de cumprimento.

```
program hello;
begin
  writeln('Hello, voce esta usando o Pascal XSC!');
end.
```

Fig. 5.2 Exemplo 1 - Programa hello.p

Ao digitar o nome do programa, é mostrado na tela o menu principal do sistema de desenvolvimento Pascal XSC, o qual é dado pela figura 5.3. Antes de ser detalhado o processo de compilação, será dada uma breve descrição de cada uma destas opções do menu principal.

PASCAL-XSC to C Development System Commands are:

E = Enter the Editor
C = Compile a PASCAL-XSC program
M = Make a PASCAL-XSC program
L = edit a Listing file
P = Print a file
R = Run a PASCAL-XSC program
B = create a Batch file
F = change the file to work on
Y = sYstem call
D = toggle DISPLAY MAIN MENU
O = Option menu
H = Help menu
Q = Quit (leave) the PASCAL-XSC development system

Current file is hello.p
Command (f/e/p/c/m/b/r/y/q/d/h/o/l) or ? for help :

Fig 5.3 Menu principal do sistema interativo Pascal XSC

A primeira **opção** ***e*** do menu principal é o editor, com o qual se podem editar os arquivos. Para a edição de um programa digita-se *e filename.p*, onde *filename* é o nome do programa. O editor utilizado é definido pela variável de ambiente.

A **opção** ***c*** inicia a compilação do programa *filename.p* carregado como arquivo corrente. Se o programa fonte contiver erros, eles poderão ser detectados pelo compilador. Os erros são mostrados de uma forma resumida: o número do erro, a linha onde ocorreu o erro e a coluna indicando a posição do erro no arquivo fonte Pascal. Esta informação pode ser utilizada quando o gerador de listagem falha por alguma razão.

Após a mensagem de falha na compilação, *Pascal compilation unseccessful*, o gerador de listagem rápido é chamado automaticamente pelo gerenciador interativo, sendo a listagem mostrada na tela. O arquivo pode ser corrigido pelo uso do editor. Este processo de acerto e recompilação pode ser feito até que a mensagem *linkage complete* seja apresentada.

As opções do compilador Pascal XSC e do compilador C podem ser providas, se for introduzida a letra *c* após a opção *c*. Mais detalhes sobre as opções do compilador serão dadas mais adiante. As opções do compilador C devem ser separadas por ponto e vírgula das opções do compilador Pascal. Para as opções do compilador C *djgpp*, deve-se consultar os manuais apropriados.

A **opção *l***, ativa listagem de erros. Isto é aconselhável quando a listagem de erros é muito grande, ultrapassando a capacidade da tela, então a opção *l* serve para corrigir os erros. Esta opção é chamada de gerador de listagem longa. Ela contém toda a listagem do arquivo fonte Pascal XSC intercalada pelas mensagens de erro. O editor é chamado automaticamente. Após a correção dos erros, um novo código fonte é produzido a partir do arquivo de listagem pelo programa executável *l2p*.

A **opção *r***, ativa o comando de execução, ou seja solicita ao gerenciador executar o programa. Na verdade, é necessário utilizar outro *r*, que é uma das opções de tempo real. Se nenhum parâmetro do programa for especificado após o primeiro *r*, então o gerenciador solicitará que se introduza algum parâmetro. Deve-se introduzir o comando *rr* caso se queira executar o programa com os mesmos parâmetros especificados em uma execução prévia da atual seção do gerenciador interativo.

A **opção *q*,** finaliza a execução do programa gerenciador interativo *dxsc*.

A **opção *f***, possibilita a troca do programa corrente. O programa corrente pode ser editado, compilado e executado. O gerenciador interativo permite que sejam armazenados até dez arquivos. O troca entre eles é feita pelo comando *f*. Os arquivos são identificados por letras e dígitos. A letra *p* identifica o último programa principal, a letra *m* identifica o último módulo e a letra *o* identifica um arquivo que pode ser programa ou módulo.

A **opção *h***, ativa o sistema de ajuda do gerenciador interativo. Pode-se utilizar também o sinal de interrogação (?). O sistema de ajuda fornece, em tempo de execução, informações e facilidades de todos os comandos do gerenciador.

A **opção *d***, habilita e desabilita a aparição do menu principal. Um d desativa a aparição do menu principal, outro d reativa o menu na tela. Este comando é chamado de *display toggle*.

A **opção *y*** do gerenciador interativo possibilita o processamento de comandos do sistema operacional sem sair do gerenciador interativo, para tanto, deve-se intrroduzir a letra *y* seguida do comando do sistema.

A **opção *p*** serve para imprimir um arquivo na impressora. Ao se digitar p, o programa corrente é impresso. O comando p é dependente da instalação e do sistema operacional.

A **opção *m*** na atual implementação tem o mesmo efeito do comando *c*.

A **opção *b*** serve para mudar as opções de linkagem. Este comando serve para gerar arquivos *batch* do DOS, os quais podem ser utilizados pelo compilador C e pelo programa que *linka*. O arquivo gerado é chamado de *lxsc.bat* e pode ser utilizado para mostrar chamadas do compilador C e *linkar* com as chamadas feitas, para modificar as chamadas do compilador C, para *linkar* as chamadas, ou, ainda, para recompilar um programa quando a compilação Pascal XSC ou C falha por falta de memória.

Para modificar chamadas, são necessários os passos:

- Executar o compilador Pascal XSC pelo comando *c*;
- Executar o comando *b* do gerenciador interativo;
- Editar o arquivo *batch* gerado *lxsc.bat* (*e lxsc.bat*);
- Executar o arquivo *batch* pelo gerenciador de comando (*y lxsc.bat*).

Para recompilar um programa que falhou por insuficiência de memória, deve-se realizar os seguintes passos:

- Sair do Gerenciador pelo comando *q*;
- Iniciar a compilação separadamente do Pascal XSC;
- Executar novamente o gerenciador interativo *dxsc*;
- Executar o comando de *batch b* do gerenciador;
- Executar o arquivo *batch* gerado pelo gerenciador (*y lxsc.bat*).

Por fim, a **opção *o*** é utilizada para mudar as opções de compilação. Esta modificação resulta em alterações no arquivo que armazena as opções do ambiente local, ou seja no arquivo *p88.env*. Ao se introduzir o comando *o*, é ativado o submenu, no qual a opção *c* resulta em alterações de configuração. As opções *e* e *l* modificam as variáveis de ambiente que definem o editor e o diretório de bibliotecas. A figura 5.4 mostra o submenu do gerenciador interativo e a figura 5.5 mostra a tela quando executada a subopção *c* que ativa a configuração do ambiente e das opções de compilação.

```
PASCAL-XSC to C  Development System
 You entered the Option submenu.
 Enter the letter "c" and hit RETURN to start the configuration program
                                          of PASCAL-XSC or
 Enter the letter "e" and hit RETURN to show and change the Editor name or
 Enter the letter "l" and hit RETURN to show and change the Library
                                          specification of the Linker call or
 Enter the letter "t" and hit RETURN to toggle test output or
 Enter the letter "h" and hit RETURN for help or
 Enter the letter "q" and hit RETURN to leave the option submenu

Option command (c/e/l/t/h/q) or ? for help :
```

Fig 5.4 Submenu de gerenciador interativo

O arquivo de configuração *p88.env*, contém as definições-padrão e as dependências do sistema. Mais especificamente, o arquivo de configuração contém as atribuições das opções do compilador, as opções do compilador que não são alteradas por linhas de comandos (*nom-commands*), os nomes das extensões dos arquivos e o caminho dos arquivos de interface, como mostra a figura 5.5.

```
Local environment file found.  Modifying file p88.env.

O = command line Options:  v -l -w -m  c +x -n +f -d -h -t +r -s
N = Non command line options: -src +rm +proto +lv +y  c  v

T =  filename-extensions, file Types :
 Source  : .p          C-code  : .c          Dump    : .lst
 Interface: .mod       Header  : .h          Object  : .o
 Library  : .a         Execute : .exe

I = pathname of interface-files: c:\pxsc\sys\
R = pathname of runtime interface is system directory c:\pxsc\sys\
L = length of C identifiers = 8

H = Help                        D = Display setting and main menu (This screen)
U = Update                      E = Update + Exit
Q = Quit without update         K = Kill + Quit

Command (o/n/t/i/r/h/d/u/e/q/k/+/-) or ? for help :
Leaving Configuration program
Configuration file p88.env read.
  <<< Press the RETURN key to continue >>>
```

Fig 5.5 Configuração do ambiente - subopção c

Para modificar o arquivo de configuração, pode-se executar o programa *pxsccfg* ou então utilizar a opção ***o*** do modo interativo. Com isto, aparecerá na tela o menu descrito pela figura 5.4. A opção ***c*** ativa alterações do arquivo de configuração. Para tanto, pode-se utilizar os seguintes comandos:

help (h), o comando ***h*** introduz facilidades de ajuda em tempo real;

display (d), o comando ***d*** mostra a configuração alterada e o menu;

update (u), o comando ***u*** grava as modificações de configuração no arquivo *p88.env,* no diretório corrente, sem sair do programa;

exit (e), o comando ***e*** grava as modificações de configuração da mesma forma que o comando ***u***, mas encerra a execução do programa *pxsccfg*;

kill (k) Elimina o arquivo de configuração existente no diretório corrente;

quit (q) O comando ***q*** encerra a execução do programa *pxsccfg* sem atualizar o arquivo de configuração;

option (o), o comando ***o*** define as opções de compilação. Elas são descritas no ítem 5.3. Não devem ser alteradas as opções: -l c +x -d -t +r, para garantir a compilação.

non-command-line (n) são opções que não podem ser controladas via opções de linha de comando, mas somente pelo programa de configuração. Estas opções são:

A opção ***rm*** quando habilitada remove todos os arquivos de saída desnecessários. Quando desabilitada, todos os arquivos de saída do compilador são mantidos;

A opção ***src*** quando habilitada direciona arquivos de saída do compilador para o diretório do programa fonte, caso contrário são gravados no diretório corrente;

A opção ***proto*** indica se o compilador C admite ou não novas características ou protótipos da linguagem. Caso não admita, deve-se atribuir *-proto*.

A opção ***y*** define de onde serão utilizados os arquivos. Se estiver ativa, então indica que os arquivos estão no diretório do sistema , caso contrário, será adotado o caminho definido pela opção *r* do configurador.

type name (t), o comando ***t*** oferece a possibilidade de alterar as definições padrões das extensões dos arquivos. A definição padrão associa arquivos fontes em Pascal XSC a extensão .p, e arquivos listagem a extensão .lst. As demais extensões devem ser mantidas para evitar erros;

interface (i), através do comando ***i***, o usuário especifica o caminho onde os módulos do Pascal XSC serão encontrados;

runtime interface (r). O comando ***r*** define o caminho de onde o arquivo de inclusão *p88rts.i* do sistema de tempo real será encontrado. Este comando redefine automaticamente a opção *non-command-line* como -y.

5.2 Compilação de um Programa

Uma vez descrita a interface e os comandos de gerenciador interativo do Pascal XSC, será tratada agora a compilação de uma programa. Para fins iniciais, será compilado o programa exemplo descrito pela figura 5.2. Como foi visto, o processo de compilação inicia pelo ativar do gerenciador interativo com o programa *hello.p* e, então executar a opção de compilação *c*. Após a compilação, a execução é ativada pelo comando *rr*. O resultado da execução produzirá a mensagem na tela:

Hello, voce esta usando o Pascal XSC!

Será considerado agora outro exemplo: o desenvolvimento de um programa que realiza a soma de dois reais. Este programa será denominado de *soma.p*. A primeira coisa a ser feita é ativar o gerenciador interativo *dxsc*, então será solicitado o nome do programa corrente, que no caso é *soma.p*. Então aparecerá o menu principal.

Para a edição do programa deve ser utilizada a opção *e*. Uma vez digitado o programa, conforme a figura 5.6, deve ser salvo e retorna-se ao gerenciador interativo.

```
program soma;
var num1, num2,s: real;
begin
  writeln('Soma de dois reais');
  write('Numero 1 : ');read(num1);
  write('Numero 2 : ');read(num2);
  s:=num1 + num2;
  writeln('num1= ' ,num1);
  writeln('num2= ' ,num2);
  writeln('s= ' ,s);
  writeln('Somatorio = ',num1 + num2);
end.
```

Fig 5.6 Exemplo 2 - programa soma.p

Novamente no menu principal , o programa deve ser compilado pela opção *c*, o que produz a impressão na tela das mensagens da figura 5.7, caso não existam erros na compilação. Com a *linkagem* é gerado o programa executável soma.exe, que pode ser executado pela opção *rr* ou diretamente do sistema operacional DOS, pelo digitar soma.exe, produzindo a solução mostrada na figura 5.8, para os valores iniciais 5 e 7.

```
Current file is soma.p
Command (f/e/p/c/m/b/r/y/q/d/h/o/l) or ? for help : c
environment file c:\pxsc\sys\p88.env found.
Compiling hello.p with PASCAL-XSC version 2.03 dated 28.10.92
(C) Copyright University of Karlsruhe 1991,1992
importing module  c:\pxsc\sys\stdmod.mod
Analysis complete
Code generation complete
Pascal Compiler successful
C-compiler successful
Linkage complete.
  <<< Press the RETURN key to continue >>>
```

Fig 5.7 Etapas da compilação

```
Numero 1 :  5
Numero 2 :  7
 num1=  5.000000000000000E+000
num2=  7.000000000000000E+000
s=  1.200000000000000E+001
Somatorio =  1.200000000000000E+001
```

Fig 5.8 Execução de exemplo2 para valores iniciais 5 e 7

5.3 Opções de Compilação

O compilador Pascal XSC é denominado de ***pxsc***. Quando se digita pxsc sem nenhum parâmetro (sem programa), serão mostradas na tela todas as opções do compilador e as definições-padrão correntes, as quais definem a configuração do arquivo, como é mostrado na figura 5.10. A sintaxe da linha de comando do compilador pxsc é dada por: ***pxsc*** [+<*opção*>] [-<*opção*>] [*nome-program*], as opções são precedidas dos sinais de +/- que habilitam ou desabilitam. O nome do programa é geralmente um arquivo fonte de Pascal XSC, com extensão .p.

Antes de se introduzir as opções de compilação, cabe mencionar que o compilador pode ser executado no modo *batch*. Para tanto, utiliza-se o programa ***mxsc***, que possui o formato de sintaxe dado pela figura 5.9.

```
Syntax : C:\PXSC\BIN\MXSC.EXE [opt] [_v] [_e] [_x] pascalfile
  opt are options for the PASCAL-XSC COMPILER
  _e    : edit source-file before compilation
  _v    : suppress manager infos
  _t    : display commands as executed
  _c... : add a C-compiler option
  _x    : execute program after linkage
```

Fig 5.9 Sintaxe do comando ***mxsc***

As opções de compilação são formadas pelos símbolos de + ou - seguido pelo nome da opção, sem espaço entre o sinal e a letra. Nesta versão do compilador somente a primeira letra da opção é significativa. O sinal de + habilita e o de - desabilita a opção.

```
Compiler options are:                                          default:
+v   verbose, Compiler reports, what he is doing  No 'Compiling ...', but path & module info
-v        Compiler is quiet
+l   listing, direct messages to file *.lst                    -l
-l        display messages on terminal                         -l
+w   warning, report warnings                                  -w
-w        suppress warnings                                    -w
+m   merge,  merge source text into code                       -m
-m        no source text in code                               -m
+a   abort,  abort compilation in case of error                -a
-a        continue compilation in case of error                -a
+f   fastfor, for-loops for vector processors                  +f
-f        standard conform for-loops                           +f
+c   code,  code generation in any casecode generation only if error free
-c        no code generation
+x   index,  generate index checks                             -x
-x        no index checks                                      -x
+n   number, generate line numbers into code                   -n
-n        no line numbers in code                              -n
+d   dump,  internal dump, for debugging only                  -d
-d        no dump                                              -d
+b   bibc,  generate module bibliothec file                    -b
-b        generate module object file                          -b
+t   terminal,write code to stdout                             -t
-t        write code to file                                   -t
+r   rename, rename Pascal identifiers                         +r
-r        preserve Pascal identifiers                          +r
+s   srcdir, output to directory of source file                -s
-s        output to current directory (working dir)            -s
```

Fig. 5.10 Tela do programa pxsc contendo as opções de compilação

A seguir são descritos os comandos de compilação, eles foram listados pela figura 5.10. Estes comandos são:

verbose (v) Esta opção fornece as seguintes informações: o número da versão do compilador Pascal XSC; o nome do arquivo fonte; o caminho do arquivo de configuração que está em uso; o caminho dos módulos importados; quando a interface de um módulo foi ou não mudada; o nome e o número da linha da rotina que está sendo compilada. Portanto ***+v*** ativa o fornecimento destas informações, enquanto ***-v*** suprime-as.

list file (l) Se a opção estiver ativa ***(+l)*** todas as mensagens de erro e avisos do compilador Pascal XSC serão armazenados no arquivo de listagem (o qual possui a extensão .lst). Esta opção não deve estar ativa quando os módulos iterativos e por *batch* estiverem ativos, pois tanto o gerador curto como o gerador de listagem longa escrevem sobre este arquivo.

warnings (w) Esta opção habilita ou desabilita as advertências do compilador. No caso ***-w***, desabilitado, o compilador dará o número de mensagens suprimidas.

merge (m) Esta opção possibilita que as linhas do programa fonte Pascal XSC se tornem linhas de comentário no gerador de código C. A configuração corrente é ***-m***, a opção desabilitada.

abort (a) Esta opção pode resultar em que a compilação seja abortada quando ocorrer erro. A configuração corrente é a ***-a*** que resulta na compilação até o final, identificando todos os erros.

fastfor (f) Esta opção não está disponível na versão disponível DOS, pois ela implica que ninhos do comando *for* sejam executados por processadores vetoriais. A vetorização não é possível nesta instalação.

code generation (c) Quando ativada, esta opção ***(+c)*** força a geração de código, mesmo que existam erros de compilação. A compilação do código C gerada poderá falhar, no caso de detecção de erros de compilação. Se nenhuma das opções estiver ativa, será gerado o código somente quando não houver erros.

index check (x) Esta opção habilita a realização de verificações no tempo de execução, estas podem ser de índices, intervalos ou ponteiros. Quando desabilitada, possibilita uma execução mais rápida. Deve-se usar a opção desabilitada ***(-x)*** somente quando se tem certeza de que não há erros no programa.

line number (n) Quando habilitada possibilita a informação de linha no programa gerado. No caso de erros em tempo de execução, uma função dinâmica mapeia de volta e fornece o número da linha do programa fonte dado pelo sistema de tempo real. A opção desabilitada não fornece isto, mas proporciona uma execução mais rápida.

dump (d) Quando ativada ***(+d)*** são produzidas várias saídas pelo compilador, geralmente inúteis ao usuário normal, mas úteis para uma depuração do programa.

terminal (t) Quando ativada direciona o código C gerado para a saída padrão, como um arquivo de saída com extensão .c.

rename (r) Quando desabilitada ***(-r)***, faz com que os nomes dos identificadoresPascal XSC sejam preservados. Isto facilita a depuração e leitura do programa C gerado, uma vez que os nomes originais (em Pascal) são mantidos no código fonte C. A compilação e a linkagem pode falhar na configuração ***-r.***

source diretory (s) Esta opção determina o diretório onde as saídas do compilador serão armazenados. Na opção ***(+s)*** o diretório de saída será o mesmo onde o arquivo fonte estiver contido; o caso contrário, na opção ***-s*** o diretório de saída será o corrente.

5.4 Módulos do Pascal XSC

Neste ítem são apresentados os módulos disponíveis do Pascal XSC.Alguns destes módulos são importados automaticamente pelo programa mesmo sem o uso do comando **use** que, geralmente, especifica o uso de um módulo. Outros módulos, como os aritméticos necessitam ser gerados, pois eles são instalados em linguagem fonte, precisando serem compilados.

5.4.1 Módulo stdmod

O módulo stdmod é importado automaticamente sem o uso explícito do comando **use**. Este módulo contém as definições dos identificadores e operadores que são prédeclarados na linguagem Pascal XSC.

5.4.2 Módulos aritméticos

A implementação das partes dependentes dos módulos aritméticos são em relação as operações de saída e ao domínio das funções matemáticas. As operações de saída para vetores e matrizes são especificadas nos módulos mv_ari, mvc_ari, mvi_ari e mvci_ari são mapeados como operações de saída correspondentes ao tipo de componente, que podem ser: reais, complexos, intervalos e complexos intervalares, respectivamente. Não existe nenhuma função matemática definida para vetores e matrizes.

5.4.2.1 Módulo i.ari

No módulo i.ari são implementadas todas as funções matemáticas sobre intervalos. São também implementadas as funções de entrada e saída de dados intervalares, como por exemplo a função: ***procedure write (var f:text, i:interval);*** que produz uma representação decimal dos limites inferior e superior do intervalo com arredondamento para baixo e para cima, respectivamente. Somente os dígitos significativos da representação decimal são

mostrados, isto é, todos os primeiros dígitos coincidentes e um pequeno número de dígitos diferentes são mostrados. O formato de saída padrão consiste de 52 caracteres, sendo estes: um colchete esquerdo, um espaço, vinte e três caracteres do limite inferior do intervalo, uma vírgula, um espaço em branco, vinte e três caracteres do limite superior do intervalo, um espaço em branco e o colchete direito.

A relação das funções matemáticas intervalares, bem como seus domínios de definição podem ser encontrados nos manuais no Pascal XSC. Observa-se que a função arco tangente do quociente de dois argumentos intervalares não está incluído no módulo i_ari, pois foi implementada separadamente, em um outro módulo chamado iatan2.

5.4.2.2 Módulo c.ari

As funções matemáticas para o tipo de dados complexos são viabilizadas por este módulo, a relação destas funções podem ser encontradas na literatura e nos manuais, juntamente com seus domínio de definição. As rotinas de entrada e saída de dados também são implementadas neste módulo. A rotina de impressão de dados complexos, ***procedure write (var f:text, c:complex);*** produz a representação decimal das partes real e imaginária do valor complexo arredondado para a representação decimal mais próxima. O formato de saída padrão, também contém 52 caracteres, sendo um parentese esquerdo, um espaço em branco, vinte e três caracteres da parte real, uma vírgula, espaço em branco, vinte e três caracteres da parte imaginária, um espaço em branco e o parentese direito.

Funções matemáticas para tipos de dados complexo baseados no formato em dupla precisão do Padrão IEEE e uma exatidão de 2 *ulps* não estão ainda implementadas.

5.4.2.3 Módulo ci.ari

Neste módulo são implementadas as funções e operações sobre os dados complexos intervalares. A rotina de impressão de valores, ***procedure write (var f:text, z:interval);*** produz uma representação decimal da parte real e imaginária do intervalo de números complexos. Ambas as partes imaginárias e reais são mostradas pela rotina intervalar de saída descrita anteriormente. Portanto, o formato de saída padrão, consiste de duas linhas separadas por uma linha em branco num total de 111 caracteres (55 caracteres na primeira lina e 56 caracteres na segunda linha). Ou seja, tem-se um parentese esquerdo, seguido de um espaço em branco, 52 caracteres da parte real, uma vírgula seguida por um caracter de nova linha, dois espaços em branco, caracteres da parte imaginária, espaço em branco e um parentese direito.

Funções matemáticas para o tipo de dado intervalar complexo (*cinterval*) baseados no formato em dupla precisão do Padrão IEEE e uma exatidão de 1 *ulp* com respeito à avaliação dos limites não estão ainda implementados.

5.4.3 Módulo iostd

O módulo *iostd* é o módulo padrão de I/O, ele contém declarações de algumas rotinas de entrada e saída adicionais e as definições de constantes. Estas constantes são: entrada padrão (stdin = 0), saída padrão (stdout = 1), erro padrão (stderr = 2), tipo de terminal (stdcon = 3), impressora (stdprn = 4), leitora (stdrdr = 5), *puncher* (stdpun = 6), arquivo temporário a ser removido no final (stdtmp = 8), e de um arquivo associado a linha de comando (stdorg=9). Estes nomes de constantes devem ser utilizados como o parâmetro nr nas rotinas de *reset* e *rewrite*.

As rotinas declaradas neste módulo são: *reset*(var t:text; nr:integer), *rewrite*(var t:text; nr:integer), *close*(var t:text), *flush*(var t:text), *exit* (retcode:integer). Também são declaradas as funções *ilexists*(s:string) do tipo booleana e a *getenv*(s:string).

Uma vez importado o módulo ***iostd,*** os procedimentos *reset* e *rewrite* podem ser chamados com uma das constantes *stdin, stdout, stderr, stdcon, stdprn, stdrdr, stdpun, stdtmp* ou *stdorg* como segundo parâmetro.

5.4.4 Módulo x_intg

O módulo x_intg contém definições adicionais de operadores inteiros, os quais operam sobre inteiros. Estes operadores proporcionam a possibilidade de se acessar e manipular *bits* individuais do formato de dados inteiros. Para eles, os operandos inteiros são interpretados como campos de 32 *bits* e não como um simples valor inteiro.

5.4.5 Módulo x_real

O módulo x_real contém constantes, tipos, funções e procedimentos adicionais para um processamento extendido ou alternativo de valores reais.

Este módulo permite a classificação de valores reais, a composição e decomposição de valores reais, possui a implementação das funções matemáticas com argumentos reais, tais como exponenciação, raiz quadrada, seno, cosseno, tangente, arco seno, arco cosseno, arco tangente, etc.

Permite a manipulação das exceções do padrão IEEE. Um número de constantes são definidas que podem ser utlilizadas junto com procedimentos *IEEE_environment e IEEE_trap_enable.*

As constantes *IEEE_INV_OP, IEEE_DIV_BY_ZERO, IEEE_OVERFLOW, IEEE_UNDERFLOW* e *IEEE_INEXAT* caracterizam as cinco exceções especificadas pelo padrão IEEE.

5.4.6 Módulo s_strg

O módulo s_strg contém as declarações de rotinas que manipulam *strings*, ou variáveis do tipo *string*, como por exemplo a rotina ***x_release (var s: string)***, o qual é somente significativo a argumentos *strings* sem especificação de tamanho estático na declaração. O propósito é liberar a porção não utilizada de uma variável *string*, isto é, o tamanho da variável *string* alocada é reduzida ao tamanho da variável *string* correntemente armazenda. A variável *string* corrente pode ser mais curta devido ao uso do procedimento padrão *setlength* ou a atribuição de uma *string* mais curta.

5.4.7 Módulos lss, ilss, clss, cilss

O PASCAL-XSC possui módulos para a solução de sistemas de equações lineares e inversão de matriz.

Os módulos chamados lss, ilss, clss, cilss contém dois procedimentos chamados LSS e INV para argumentos da matriz e do vetor com componetes do tipo *real, interval, complex* e *cinterval* respectivamente.

Para os módulos xlss (x= vazio, i, c, ci) os procedimentos LSS e INV são declarados da seguinte forma com y=R, I, C, CI e z=R, I, C, CI de acordo com a escolha de x.

procedure *LSS*(var A: yMATRIX, var B:yVECTOR, var y:zVECTOR,var ERRCODE: INTEGER)
procedure *INV*(var A:yMATRIX, var Y:zMATRIZ, var ERRCODE: INTEGER)

LSS resolve sistemas de equações lineares da forma A.x = b, com A uma matriz m.n, b um vetor m.1, e x um vetor 1.n. A inclusão verficada da solução exata é retornada pelo argumento Y. Os conteúdos dos argumentos A e B não são trocados.

INV determina a inversa da matriz A sendo esta m.n, se m=n. De outra maneira, a pseudo inversa (Inversa de Moore-Penrose) é determinada.

5.4.8 Geração dos módulos e bibliotecas

O PASCAL-XSC utiliza bibliotecas, como os módulos aritméticos que implementam intervalos e complexos, durante o processo de compilação, estas bibliotecas possuem a extensão **.mod**. Estas bibliotecas são conhecidas como os módulos avançados, nos quais são incorporados os novos tipos de dados juntamente com as operações que os manipulam, formando as aritméticas descritas na figura 5.11, as quais podem ser incorporadas aos programas através do comando **use**.

I_ARI	aritmética intervalar
C_ARI	aritmética complexa
CI_ARI	aritmética intervalar complexa
MV_ARI	aritmética real matricial/vetorial
MVI_ARI	aritmética intervalar matricial/vetorial
MVC_ARI	aritmética complexa matricial/vetorial
MVCI_ARI	aritmética intervalar complexa matricial/vetorial

Fig. 5.11 Módulos avançados

Estes módulos necessitam ser gerados a partir dos módulos de sufixo **.a**, gerando os **.mod**. As bibliotecas são geradas a partir de programas em pascal (**.p**), gerando arquivos **.c** dos quais são gerados os **.mod**. As bibliotecas são criadas pelo programa utilitário **gcc.exe**, aumentando, com isso, a flexibilidade da linguagem. Para se criar uma biblioteca, devem-se seguir os seguintes passos:

- Cria-se um programa em pascal (com extensão **.p**) através da edição (um editor);
- Compila-se este programa com o compilador **pxsc.exe**, gerando um arquivo com extensão **.c**;
- Utiliza-se o utilitário **gcc.exe** para gerar os módulos (ex: gcc -c -w rotina.c)

Após estes passos ter-se-á um arquivo chamado rotina.mod, que pode ser utilizado como uma biblioteca da mesma maneira que os módulos já existentes no PASCAL-XSC.

6 EXEMPLOS DE PROGRAMAS EM PASCAL XSC

A seguir, serão apresentados alguns programas em Pascal XSC, demonstrando o uso dos módulos aritméticos e vários conceitos do Pascal XSC. Algoritmos bem conhecidos foram escolhidos intencionalmente, para que apenas uma breve introdução matemática fosse necessária. Além disto, os programas contêm comentários e estão auto-documentados. Os programas abrangem o Método de Newton Intervalar, o Método de Runge Kutta, cálculo de um produto de matrizes e a verificação da solução de um sistema linear.

6.1 Método de Newton Intervalar

No Método de Newton Intervalar é calculado um zero da função f(x) de valores reais. É assumido que f'(x) é uma função contínua no intervalo [a,b], onde $0 \notin \{f'(x): x \in [a, b]\}$ e $f(a).f(b) < 0$. Se uma inclusão Xn para o zero da função f(x) já é conhecida, a melhor inclusão Xn+1 pode ser, geralmente, calculada pela fórmula de iteração:

$$X_{n+1} := (m(X_n) - \frac{f(m(X_n))}{f'(X_n)} \cap X_n$$

onde m(X) é algum ponto do intervalo X (por exemplo o ponto médio). Para este exemplo, a função f(x) será tomada como:

$$f(x) = \sqrt{x} + (x+1).\cos(x)$$

Em Pascal XSC, expressões intervalares são escritas na notação intervalar. Nomes das funções genéricas são usados para funções intervalares, como raiz quadrada intervalar, seno e cosseno intervalar. Maiores detalhes da teoria pode ser achado em [ALE83]. A figura 6.1 contém o programa do método de Newton em Pascal XSC.

```
program inewt (input, output);
use
 i_ari; {i_ari : interval arithmetic}
var  x,y : interval;
fuction f(r : real) : interval;
var  x : interval;
begin
 x:=r; {converts r to type interval to obtain a verified inclusion}
 f:= sqrt(x)+(x+1)*cos(x)
end;
function deriv(x: interval) : interval;
begin
 deriv:= 1 / (2 * sqrt(x)) + cos(x) - (x+1) * sin(x)
end;
function criter (x: interval) : boolean;
begin
 criter:= (sup(f(inf(x)) * f(sup(x))) < 0) and not (o in deriv(x));
end;
begin
write('Entre com o intervalo inicial:');
read(y);
while inf(y)<>sup(y) do
 begin
    if  criter(y) then
        repeat
          x:= y;
          writeln (x);
          y:= ( mid(x) - f(mid(x))/deriv(x) ** x;
        until x=y
    else
       writeln ('Critério não satisfeito !');
    writeln;
    write ('Entre o intervalo inicial');
    read(y);
  end;
end.
```

Fig 6.1 Método de Newton Intrevalar em Pascal XSC

Com o intervalo inicial [2, 3], as inclusões calculadas foram:

```
[                2.0E+000                3.0E+000   ]
[                2.0E+000                2.3E+000   ]
[                2.05E+000               2.07E+000  ]
[             2.05903E+000            2.05906E+000  ]
[        2.059045253413E+000     2.059045253417E+000  ]
[      2.059045253415143E+000  2.059045253415145E+000  ]
```

6.2 Método de Runge Kutta

O método de Runge Kutta resolve o problema de valores iniciais para sistemas de equações diferenciais. O Método para resolver uma equação diferencial pode ser escrito no Pascal Padrão na notação matemática usual. Em Pascal XSC é possível se utilizar a mesma notação para sistemas de equações diferenciais. O conceito de arrays dinâmicos é usado para desenvolver um programa, independente do tamanho do sistema. Somente necessita mais armazenagens no tempo de execução. Seja o seguinte sistema de equações diferenciais de primeira ordem $Y' = F(x,Y)$, com a condição inicial $Y(x0) = Y0$. Se a solução Y é conhecida para o ponto x, então a aproximação $Y(x+h)$ é calculada por:

$K1 = h.F(x,Y)$

$K2 = h.F(x+h/2, Y+K1/2)$

$K3 = h.F(x+h/2, Y+K2/2)$

$K4 = h.F(x+h, Y+K3)$

$Y(x+h) = Y + (K1 +2*K2 +2*K3 +K4)/6$

Iniciando com x0, uma solução aproximada pode ser calculada nos pontos xi = x+ i.h.

```
module f;
use
  mv_ari; {mv_ari : aritmética vetorial e matricial}
global const
  dim = 3;
global function f(x : real; y : rvector) : rvector[1..dim];
begin
  f[1] := y[1] - y[2];
  f[2] := exp(x) * y[3];
  f[3] := (y[1] -y[2])/exp(x);
end;
```

```
global procedure init (var x,h : real; var y : rvector);
 begin
  x :=0;
  h:= 0.1;
  y[1]:=0;
  y[2] := 1;
  y[3] := 1;
 end;
end. {fim do módulo}

program runge (input, output);
use
mv_ari, f;
var
i                       : integer;
x,h                     : real;
y, k1, k2, k3, k4       : rvector[1..dim];
begin
 init(x, h, y);
 for i:= 1 to 10 do
 begin
        k1 := h*f(x,y);
        k2 := h*f(x+h/2, y+k1/2);
        k3 := h*f(x+h/2, y+k2/2);
        k4 := h*f(x+h, y+k3);
        y(x+h) := y + (k1 +2*k2  +2*k3 +k4)/6;
        x := x+h;
        writeln('x =',x);
        writeln('y =',y);
 end;
end.
```

Fig 6.2 Programa do Método de Runge Kuta

6.3 Cálculo de um produto de matrizes

O próximo programa em Pascal XSC demonstra o uso de expressões exatas. O cálculo de um produto de matrizes A.B é calculado sem a avaliação do produto das matrizes propriamente dito. O resultado será em máxima exatidão, isto é, será o melhor ponto flutuante possível que aproxime a solução exata. O cálculo de um produto de matrizes é dado pelos somatórios

$$\sum_{i=1}^{n}\sum_{j=1}^{n} aij.bij\,.$$

```
program trace (input, output);
use
 mv_ari; {mv_ari : aritmética vetorial e matricial}
var
 n : integer;
procedure main (n : integer);
var
 i, j : integer;
 s, d : real;
 A, B : rmatrix [1..n, 1..n];
begin
 read(A, B);
 s:= 0;
 for i:= 1 to n do
   s:= s +A[i]*rvector(B[*,i]);
 writeln (`Cálculo de A*B pelo produto escalar:', s);
 d:= #*( for i:= 1 to n sum (A[i] *rvector(B[*,i])));
 writeln (`Cálculo de A*B pela expressão exata #: ', d);
end;
begin
 read(n); main(n);
end.
```

Fig 6.3 Programa Multiplicação de Matrizes

Se as seguintes matrizes são introduzidas

$$A=\begin{bmatrix} 1e9 & 8 & 126 & -237 \\ 100 & 2 & -12 & 1 \\ 1e5 & 10 & -1e7 & 81 \\ 13 & -3 & 30 & 1e-7 \end{bmatrix} \text{ e } B=\begin{bmatrix} 1e8 & 85 & 8 & 6 \\ 12 & 3 & 1e3 & 156 \\ 3 & 14 & 1e10 & 13 \\ 2 & -8332 & -1e4 & -1e-8 \end{bmatrix}$$

os resultados calculados pelo programa são:

```
Cálculo de A*B pelo produto escalar:      -9.999999999999999E-016
Cálculo de A*B pela expressão exata #:     5.999999999999999E+000
```

6.4 Verificação da solução de um sistema linear

O seguinte exemplo demonstra um programa para a verificação da solução de um sistema de equações lineares. O programa produz tanto a verificação da solução quanto uma mensagem correspondente a falha.

O módulo LIN_SOLVproduz a solução de sistemas de equações lineares através de um vetor intervalar calculado por sucessivas iterações intervalares. A rotina *main*, a qual é chamada no corpo do programa *lin_sys*, é somente usado para a leitura da dimensão do sistema e para alocação das variáveis dinâmicas. O método numérico em si, é iniciado pela chamada da rotina linear_system_solver definido no módulo LIN_SOLV. Esta rotina pode ser chamada com arrays dimensões arbitrárias. Para maiores informações sobre métodos de iteração com verificação automática do resultado, ver [KAU87, KUL88, KUL89, RUM83].

```
module lin_solv;

use
 i_ari, mv_ari, mvi_ari;

priority
 inflated = *; {nível 2 de prioridade}

operator inflated (a: ivector; eps: real) infl: ivector[1..ub(a)];

var
 i: integer;
 x: real;

begin
 for i:= 1 to ub(a) do

 begin
  x:= a[i];
  if (diam(x) <> 0) then
     a[i] := (1+eps)*x - eps * x
  else
      a[i] := intval(pred (inf(x)), succ(sup(x)) );
 end; {fim do for}
 infl := a;
end; {fim da definição do operador inflated}
```

```
function approximate_inverse (A:matrix): rmatrix[1..ub(A), 1..ub(A)];
var
 i,j,k, n : integer;
 factor  : real;
 R, Inv, E : rmatrix[1..ub(A), 1..ub(A)];
begin
 n := ub(A); {dimensão de A}
 E := id(E); {Matriz identidade}
 R := A;
 for i := 1 to n do
  for j := (i+1) to n do
  begin
    factor := R[j,i]/R[i,i];
    for k :=1 to n do R[j,k]:= #* (R[j,k]-factor*R[i,k]);
    E[j] := E[j] - factor*E[i];
   end; {fim do for j...}
   for i :=n downto 1 do
     Inv[i] := #*(E[i] - for k:=(i+1) to n sum (R[i,k]*Inv[k]))/R[i,i];
   approximate_inverse := Inv;
end; {fim da função approximate_inverse}

global procedure linear_system_solver (A:rmatrix; b:rvector; var x:ivector; var
ok:boolean);
const
 epsilon = 0.25; {constante para inflação epsilon}
 max_steps = 10; {número máximo de iterações}
var
 i: integer;
 y,z : ivector[1..ub(A)];
 R: rmatrix[1..ub(A),1..ub(A)];
 C: imatrix[1..ub(A),1..ub(A)];
begin
 R:= approximate_inverse(A);
 z := R*intval(b);
 C := ##(id(A)-R*A);
 x :=z;   i :=0;
 repeat
  i:= i+1;
  y := x inflated epsilon;
  x := z+ C*y;
  ok := x in y;
 until ok or (i=max_steps);
end; {fim da rotina linear_system_solver}
end. {fim do módulo lin_solv}
```

```
program lin_sys (input, output);
use
 lin_solv, mv_ari, mvi_ari;
var
 n :integer;
procedure main (n:integer);
var
 ok : boolean;
 b :  rvector[1..n];
 x : ivector[1..n];
  A : rmatrix[1..n,1..n];
begin
 writeln('Entre com a matriz A');  read(A);
 writeln('Entre com o lado direito de b');  read(b);
 linear_system_solver(A,b,x,ok);
 If ok then
  begin
      writeln('A matriz A não é singular e a solução esta contida em');  write (x);
  end
  else
      writeln('Não foi encontrado a solução!');
end;
begin
 write('Entre com a dimensão n do sistema linear');  read(n);
 main(n);
end.
```

Fig 6.4 Módulo de solução de sistemas de equações

BIBLIOGRAFIA

[ADA93] ADAMS,E; KULISCH,U. (Eds) - **Scientific Computing with automatic result verification.** San Diego, Academic Press, 1993.

[ALB93] ALBRECHT,R; ALEFELD,G; STETTER,H.J (Eds) - **Validation numerics- Theory and applications.** Computing Supplementum 9 Wien, Springer Verlag, 1993.

[ALE83] ALEFELD, G.; HERZBERGER,J. **Introduction to Interval Analysis.** Academic Press, 1983.

[ALL91] ALLENDOERFER,U; SHIRIAEV,D. - Pascal-XSC to C a portable Pascal XSC compiler. In:[**Kau91**, pp.91-104], 1991.

[ALL92] ALLENDOERFER,U; SHIRIAEV,D. - Pascal-XSC A portable development system. In: Proceedings of the 13th IMACS World Congress, Dublin, 1992.

[IEEE754] American National Standards Institute / Institute of Electrical and Electronics Engineers: IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE Std. 754-1985, New York, 1985.

[ANSI C] American National Standard for Information Systems: Programming Language C. ANSI X3.159-1989.

[BOH86] BOHLENDER,G; RALL,L; ULLRICH, Ch; WOLFF von GUDENBERG,J. PASCAL-SC: Wirkungsvoll programmieren, kontrolliert rechnen. Bibliographisches Institut, Mannheim, 1986.

[BOH87] BOHLENDER,G RALL,L; ULLRICH, Ch; WOLFF von GUDENBERG,J. A computer language for scientific computation. Academic Press, New York, 1987

[BOH91a] BOHLENDER,G - Vector extension of the IEEE standard for floating point arithmetic. In:[**Kau91**, pp.3-12],1991.

[BOH91c] BOHLENDER,G; KROFEL,A. - A Survey of pipeline hardware support for accurate scalar products. In: [**Kau91**, pp.22-43], 1991.

[BOH92] BOHLENDER,G.; DAVIDENKOFF, A Support for parallel programs in Pascal-XSC on a Transputer System. In: [SCAN91 Proceedings, 1992]

[BOH93] BOHLENDER,G.; CORDES,D; KNÖFEL,A; KULISCH, U; LOHNER,R; WALTER, W. Proposal for accurate floating-point vector arithmetic. In: [ADA93, p.87-102], 1993.

[COR91] CORDES,D,W - Runtime systems for a Pascal XSC compiler. In:[**Kau91**, pp.151-160],1991.

[HAM90] HAMMER,R - How reliable is the arithmetic of vector computers? In:[**Ull90b**, pp.467-482], 1990.

[HAM92] HAMMER,R Maximal genave Berechnung von Skalarproduktausdrücken und hochgenaue Auswertung von Programmteilen. Dissertation, Universität Karlsruhe, 1992.

[HAM93] HAMMER,R; HOCKS,M; KULISCH,U; RATZ,D. - **Numerical Toolbox for Verified Computing I: basic numerical problems.** Berlin, Springer Verlag, 1993.

[HELIOS] PERIHELION SOFTWARE LTDA. The Helios Operating System. Prentice Hall, 1989.

[IMACS91] IMACS,GAMM. - Resolution on computer arithmetic. In: [**Kau91**, pp.477-479],1991

[KAU87] KAUCHER, E; KULISCH, U; ULLRICH, Ch (Eds) - **Computerarithmetic: scientific computing and programming languages.** Stuttgart, BG Teubner Verlag, 1987.

[KAU91] KAUCHER,E; MARKOV,S.M; MAYER,G (Eds) - Computer Arithmetic, Scientific Computation and Mathematical Modelling. **Proceedings of SCAN'90** - Albena,Sept 24-28, 1990. IMACS Annals on Computing and Applied Mathematics, v.12. Basel: J.C.Baltzer, 1991.

[KIR88a] KIRCHNER,R; KULISCH,U - Arithmetic for vector processors. In:[**Moo88**, pp.3-41], 1988.

[KIR88b] KIRCHNER,R; KULISCH,U. - Accurate arithmetic for vector processing. **Journal of parallel and Distributed computing**, v.5, n.3, pp.250-270, June, Academic Press, 1988.

[KLA92] KLATTE, R; KULISC, U; NEAGA, M; RATZ, D; ULLRICH, Ch - **PASCAL-XSC language reference with examples.** Berlin, Springer Verlag, 1992.

[KUL81] KULISCH, U; MIRANKER,W.L. - **Computer arithmetic in theory and Practice.** New York, Academic Press, 1981.

[KUL83] KULISCH, U; MIRANKER, W. L. (Eds) - **A new approach to scientific computation.** Proceedings of symposium held at IBM Research Center, Yorktown Heights, 1982. New York, Academic Press, 1983.

[KUL88] KULISCH, U; STETTER,H.J (Eds) - **Scientific computation with automatic result verification.** Seminar held in Karlsruhe Sept.30-Oct.2, 1987. Computing suplementum 6, Wien, Springer Verlag, 1988.

[KUL89] KULISCH, U. (Ed)- **Wissenschaftliches Rechner mit Ergebnisverifikation eine einfuhrung.** v.58, Akademie Verlag, Berlin, und Vieweg Verlagsgerellschaft, Wiesbaden, 1989.

[MOO88] MOORE,R E (Ed) - **Reliability in Computing: The role of interval methods in scientific computing.** PERSPECTIVES IN COMPUTING V 19. Proceedings of the conference at Colombus, Ohio, Sept 8-11,1987. San Diego, Academic Press, 1988.

[NEA84] NEAGA, M. Erweiterungen von Programmiersprachen für wissenschaftliches Rechnen und Erörterung einer Implementierung. Dissertation, Universität Karlsruhe, 1984.

[RAT90] RATZ,D - The effects of the arithmetic of vector computers on basic numerical algorithms. In: [**Ull90b,** pp.499-514],1990.

[ULL90a] ULLRICH,Ch (Ed) - Computer Arithmetic and self-validating numerical methods. **Proceedings of SCAN'89, invited papers.**Basel, Oct 2-6, 1989. San Diego, Academic Press, 1990.

[ULL90b] ULLRICH,Ch (Ed) - Contributions to computer arithmetic and self-validating Numerical Methods. **Proceedings of SCAN'89, submitted papers.** Basel, Oct 2-6, 1989. IMACS Annals on Computing and Applied Mathematics, v.7. Basel: J.C Baltzer, 1990.