UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Estudo da utilização de uma ferramenta para construção de programas em Português em disciplinas básicas de ensino de programação

por

MARCO AURÉLIO FREITAS SANTOS

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Dr. Antônio Carlos da Rocha Costa
Orientador

Porto Alegre, maio de 2001.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Marco Aurélio Freitas

Estudo da utilização de uma ferramenta para construção de programas em Português em disciplinas básicas de ensino de programação / por Marco Aurélio Freitas Santos.- Porto Alegre: PPGC da UFRGS, 2001.

85p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, BR – RS, 2001. Orientador: Costa, Antonio Carlos da Rocha

1. Linguagem de Programação. 2. Informática Educativa. 3.Compiladores. 4. Algoritmos. I. Costa, Antonio Carlos da Rocha. II. Título.

Agradecimentos

Quero agradecer a todas as pessoas que estiveram me ajudando durante a realização deste trabalho, em especial à minha esposa Rogéria, pela compreensão das horas ausentes e pelo incentivo nas horas difíceis.

Ao meu orientador Prof. Dr. Antonio Carlos da Rocha Costa, pelas valiosas sugestões que me deram rumo e pelo acompanhamento da elaboração deste trabalho.

Ao Adriano Câmara, meu amigo, que me incentivou a iniciar o mestrado.

Por fim, dedico este trabalho aos meus pais, Valdevino e Maria das Graças, pelos ensinamentos que me guiam durante toda a vida.

Sumário

Lista de Abreviaturas

BNF	Backus-Naur Form
PCP	Pseudo-Compilador Portugol
UNIGRAN	Centro Universitário da Grande Dourados-MS
MS-DOS	Microsoft Disk Operating System

Lista de Símbolos

::=	é definido como
!	ou
{ }	Indica que o conteúdo pode ser repetido nenhuma ou várias vezes; pode indicar também comentários que não influenciam no funcionamento do programa.
< >	Indica que o conteúdo é uma construção sintática BNF
<ENTER>	Refere-se ao nome de uma tecla que deve ser pressionada
<identificador>	Indica que no local indicado deve aparecer o nome de um identificador.

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho tem por objetivo apresentar e estudar a aplicação de uma ferramenta chamada PCP – Pseudo-Compilador Portugol, criada para auxiliar estudantes de programação a aprimorar o raciocínio lógico e a criar programas estruturados, sem que precisem se preocupar com comandos e instruções em Inglês ou tenham conhecimento de uma linguagem de programação específica.

Por ser uma ferramenta que usa somente palavras do nosso idioma, os alunos podem direcionar todo o seu raciocínio no entendimento e resolução do problema em forma de algoritmo.

O estudo experimental realizado neste trabalho pretende analisar e comparar o aprendizado entre grupos de alunos de disciplinas de programação utilizando e não utilizando esta ferramenta. Além de acompanhar o desempenho dos alunos, pretende também coletar informações durante as baterias de testes e obter as opiniões dos mesmos em relação ao PCP, no que se refere às facilidades, dificuldades, pontos positivos e falhas apresentadas.

Este estudo é apresentado em duas etapas, com oito baterias de teste em cada uma. Na primeira etapa foram selecionados alunos do Curso de Ciência da Computação da UNIGRAN, em Dourados-MS; na segunda etapa foram selecionados alunos da Escola Anglo Decisivo. Estas duas etapas possibilitam a análise do aprendizado proporcionado pela ferramenta com alunos que já têm alguma noção de programação e com alunos que não tiveram nenhum contato com o desenvolvimento de programas.

Palavras-Chave: Linguagem de Programação, Informática Educativa, Compiladores, Algoritmos.

TITLE: "STUDY ON THE UTILIZATION OF A TOOL FOR WRITING SOFTWARE IN PORTUGUESE LANGUAGE IN BASIC DISCIPLINES OF PROGRAMMING."

Abstract

This paper's aim is to present and study the use of a tool named PCP – Portugol Pseudo-Compiler – that has been created for helping computing students to enhance their logical thoughts and conceive structured application programs, without being concerned with English commands and instructions nor being acquainted with some specific programming language.

Since it is a tool that uses words only from Portuguese Language, the students can drive all their reasoning to the problem's comprehension and solution in form of algorithms.

The empiric study carried out in this work intends analyzing and examining the learning in groups of students who utilized and did not utilize this tool in the programming disciplines. Besides heeding the students' performances, it also intends gathering information, through the batteries of tests, and record the students' opinion about the PCP, regarding its easiness and difficulties, positive points and malfunctions perceived.

This study is shown up in two sections, with eight series of tests each one. In the first part, it has been chosen undergraduate students from the Computing Science Course of UNIGRAN, Dourados-MS; in the next part, the students selected are from Anglo-Decisivo High School. These phases made it possible to analyze the apprenticeship that this tool furthered, considering the students who already had notion of software building an those who had never had contact with software development.

Keywords: Programming Language, Educational Computing Science, Compilers, Algorithms.

1 Introdução

Construir um programa é o mesmo que construir um algoritmo. “Um algoritmo é a descrição de um padrão de comportamento, expressado em termos de um repertório bem definido e finito de ações primitivas, das quais damos por certo que elas podem ser executadas” [GUI 95].

A necessidade de facilitar o trabalho em computador por parte dos profissionais de informática é constante e uma das formas para se conseguir este objetivo é fazer com que o computador, cada vez mais, compreenda a linguagem escrita ou falada, reduzindo ao máximo a quantidade de códigos e símbolos que precisam ser aprendidos e memorizados. Por este motivo, estamos sempre buscando formas para fazer com que o computador aprenda a nossa língua, ao invés de nós aprendermos a “língua” dele.

Os programadores teriam seu trabalho facilitado se os programas fossem escritos em sentenças padronizadas da linguagem humana; infelizmente, isso não acontece. Os programas têm de ser escritos em uma linguagem de programação e há muitas dessas linguagens.

Muitas ferramentas de informática foram criadas para auxiliar o aprendizado em diversas áreas: matemática, gramática, geografia, etc. Isto nos faz perceber que é necessário também a utilização de ferramentas para o aprendizado da própria informática, para que seja possível facilitar a estruturação e codificação de programas cuja execução produza o objetivo esperado, de acordo com a especificação inicial descrita.

Em tese, o estudante de programação deve focalizar seus esforços no entendimento e resolução do problema, bem como no desenvolvimento do raciocínio lógico necessário e da abstração, ou seja, a capacidade de definir e usar estruturas ou operações complicadas, sem visualizar muitos detalhes [SEB 99]. Isto só será possível se o mesmo não tiver que se preocupar com a tradução de cada um dos comandos que ele deve utilizar como também não perder tempo por causa de uma mensagem de erro cujo significado não foi entendido corretamente.

Este trabalho visa suprir a necessidade da criação de um software específico para a elaboração de programas estruturados e aprimoramento da lógica de programação, voltada para alunos de disciplinas de programação, sem a preocupação com o idioma em que se apresentam os programas da linguagem estrutura, bem como a necessidade de conhecimento de uma linguagem de programação específica.

Visa também o acompanhamento e estudo da utilização deste software pelos alunos matriculados no curso de Ciência da Computação da UNIGRAN – Centro Universitário da Grande Dourados, na cidade de Dourados-MS., afim de verificar a viabilidade da inclusão desta ferramenta no conteúdo programático do curso, apontando as dificuldades apresentadas, vantagens e desvantagens de se utilizar uma ferramenta para o aprendizado da programação e analisar o desempenho dos alunos na resolução dos problemas propostos utilizando ou não esta ferramenta, apontando inclusive críticas feitas por eles.

A este software passaremos a chamar de PCP – Pseudo-Compilador Portugol.

A pseudo-linguagem de programação a qual chamamos de PCP é uma simbiose do Português com o PASCAL e ALGOL. Tem como objetivo principal fazer com que a pessoa que pretende desenvolver o programa se aproxime da máquina sem se prender a ela, fixando seus pensamentos no problema a solucionar. Em outras palavras, auxiliará na construção do algoritmo para que se alcance uma melhor solução, agindo como ferramenta motivadora do processo de ensino-aprendizagem.

Será um sistema que eliminará a codificação manual do algoritmo em programa. Isso fará com que o aluno (programador) não necessite de conhecimentos apurados de qualquer tipo de linguagem, bastando apenas conhecer a lógica e seguir as normas definidas para a linguagem algorítmica, criada por este projeto que, assim como qualquer outra linguagem de programação, possuirá sintaxe rígida, para que efetivamente seja verificada por seus analisadores (ver tópico 3.1).

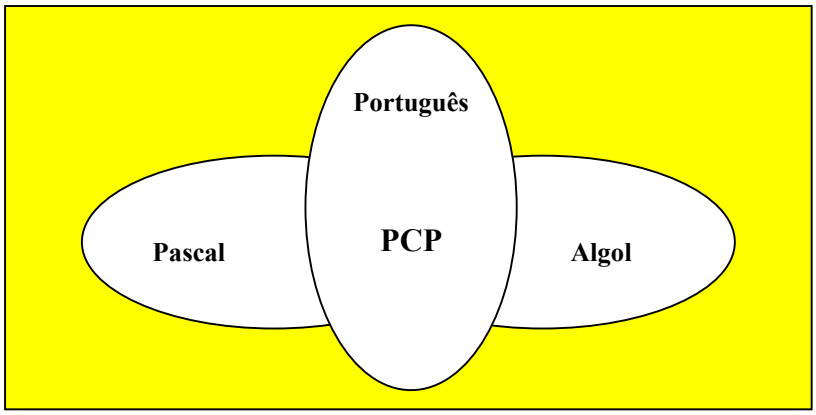

FIGURA 1.1 – Linguagens que formam o PCP

A vantagem deste projeto, portanto, não está no fato de eliminar a adoção de regras comuns na programação, mas no fato do usuário estar escrevendo seu programa em português, o que dará ao programador maior facilidade para compreender e assimilar a lógica do programa, ao mesmo tempo que exigirá o cumprimento de regras obrigatórias para confecção dos mesmos.

1.1 Linguagens utilizadas no projeto

Depois que o algoritmo foi escrito e submetido aos analisadores do PCP sem apresentar erros de programação, será feita a transcodificação deste, ou seja, ele será convertido para um código-final. Adotamos o termo "Transcodificação" porque, na verdade, estaremos transformando o código-fonte, desenvolvido especialmente neste projeto, para um código Pascal, o qual chamaremos de código-final. A linguagem Pascal foi escolhida por ser bem estruturada e muito utilizada nas universidades para o ensino de programação de computadores, principalmente nas disciplinas de linguagens, técnicas de programação e estrutura de dados.

Para a construção do PCP foi escolhida a linguagem "C". Inicialmente usamos o Turbo C, porém, para alguns aprimoramentos posteriores foi utilizado o Borland C ++. A escolha se deu por se tratar de uma linguagem com muitos recursos os quais tornam

mais fáceis a utilização de estruturas de dados complexas e a grande capacidade de manipulação de caracteres e arquivos em baixo nível.

1.2 Estrutura do texto

Este trabalho está organizado e dividido em 6 capítulos, os quais apresentam-se estruturados como mostra a seguir:

No capítulo 2 é feito um estudo apresentando os conceitos e principais problemas sobre a construção de algoritmos.

No capítulo 3 apresentamos a ferramenta objeto do presente trabalho, suas características, conceitos e estrutura.

No capítulo 4 fazemos uma breve apresentação do estudo experimental realizado neste trabalho, que foram divididas em duas etapas.

No capítulo 5 apresentamos a primeira etapa do estudo experimental feito com alunos do curso de Ciência da Computação. No final do capítulo são apresentadas conclusões em relação às experiências da primeira etapa.

No capítulo 6 apresentamos a segunda etapa do estudo experimental feito com alunos do segundo grau, da Escola Anglo Decisivo. No final do capítulo são apresentadas conclusões em relação às experiências da segunda etapa.

No capítulo 7 encontramos as resoluções de alguns dos exercícios aplicados com os alunos das duas etapas

No capítulo 8 são apresentadas as conclusões finais, com algumas comparações entre os resultados obtidos nas duas etapas da pesquisa. Incluímos também considerações sobre o trabalho desenvolvido na opinião de alunos e professores e, baseado nestas considerações, apresentamos sugestões para futuras extensões deste trabalho.

O texto possui também, na seção de anexos, a sintaxe utilizada no projeto segundo o formalismo Bakus-Naur, alguns trechos importantes da ferramenta analisada neste trabalho e a gramática do Pseudo-Compilador Portugol.

2 O Problema e a programação

"A construção metódica de programas confiáveis é, sem dúvida, uma das questões centrais da Ciência da Computação" [VEL 87]. Para cada problema a ser levado ao computador, deve-se planejar as operações correspondentes. O automatismo exige que o planejamento destas operações seja feito previamente, antes de se utilizar o computador.

Então, a utilização de um computador, para se resolver qualquer problema exige, antes de mais nada, que se desenvolva um algoritmo, isto é, que se faça a descrição do conjunto de comandos ordenados que, quando obedecidos, resultarão na realização da tarefa desejada obtendo os resultados esperados. Os algoritmos encontram-se entre a idéia de se resolver um problema através do computador e o sistema desenvolvido pelas linguagens de programação.

A elaboração de algoritmos é, para estudantes de Ciência da Computação e áreas afins, o passo inicial para a solução de qualquer problema computacional. Isto significa que, identificado o problema, será necessária a escrita de uma série de instruções, normalmente escritas em português, ou bem próximas dele, numa ordem logicamente definida, para que se obtenha a solução final do problema. É deste modo que se torna possível a construção de programas em qualquer linguagem de computador.

O processo de aprender uma linguagem de programação para resolver os problemas do algoritmo pode ser uma tarefa extensa e difícil. A programação em uma linguagem algorítmica é na verdade uma simples transcrição de palavras-chave [SAL 98], o que torna o processo muito mais fácil. Uma vez pré-definidas as seqüências lógicas das tarefas, ou instruções, a serem realizadas passo a passo, necessita-se apenas traduzi-las em uma linguagem própria, que o computador reconheça, para então submetê-las à máquina para análise e obter o seu resultado.

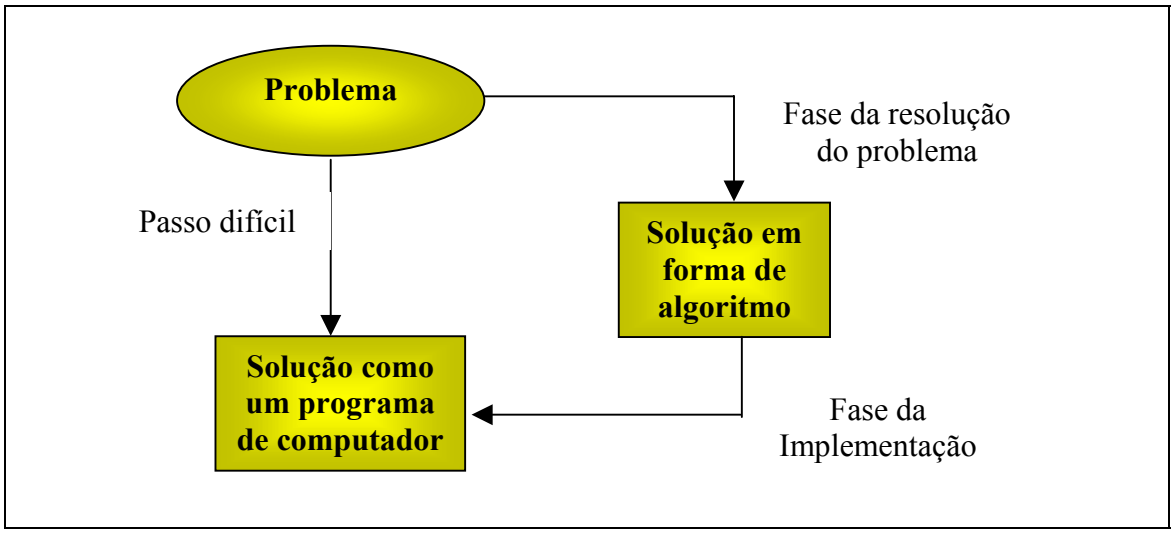

FIGURA 2.1 - Resolução do problema e programação.

Para os iniciantes na área da computação, a verificação dos resultados das possíveis soluções do problema obtidas pelo algoritmo torna-se extremamente trabalhosa. Isto porque, uma vez definidas as seqüências, é necessário simular sua execução, realizando manualmente cada instrução, até o resultado final (o que é chamado de "Teste de Mesa" ou simulação). Sendo a lógica estabelecida pela seqüência de instruções, que é a parte mais difícil para a solução de um programa de computador, o fato de poder testá-la rapidamente por meio da transformação do código-fonte em um programa Pascal faz com que a aprendizagem de uma linguagem qualquer se torne muito facilitada, pois quando o estudante chegar neste ponto, o seu programa já foi desenvolvido e testado e não foram necessárias pausas no raciocínio para esclarecer dúvidas sobre comandos ou mensagens em inglês.

Professores de informática e programação garantem que o maior problema enfrentado pelos futuros profissionais encontra-se na fase de desenvolvimento ou aperfeiçoamento da lógica computacional, sendo este o fundamento, a base para a criação de programas e sistemas computacionais. Desta forma, espera-se poder facilitar o treinamento e o aperfeiçoamento desta lógica, não necessitando de domínio ou sequer o conhecimento de alguma linguagem de programação.

3 O Pseudo-Compilador Portugol

3.1 A estrutura do PCP

Segundo [SEB 99] "a sintaxe de uma linguagem de programação é a forma de suas expressões, de suas instruções e de suas unidades de programa".

O algoritmo, quando submetido à análise do PCP, passará pelas seguintes etapas de verificação que compõem a ferramenta:

- análise léxica;
- análise sintática;
- análise de expressão;
- conversão para o Pascal.

A rotina central é o analisador sintático, que verifica se a construção das instruções dentro do algoritmo está de acordo com as regras pré-impostas por este projeto. Quando a análise sintática necessitar de um novo item léxico, é chamado o analisador léxico, que retorna parâmetros de saída com as informações sobre esse item. Para tal, o analisador léxico lê o programa fonte de cima para baixo, da esquerda para a direita e analisa as palavras do algoritmo para constatar se foram escritas corretamente produzindo a listagem do mesmo, bem como de eventuais erros léxicos.

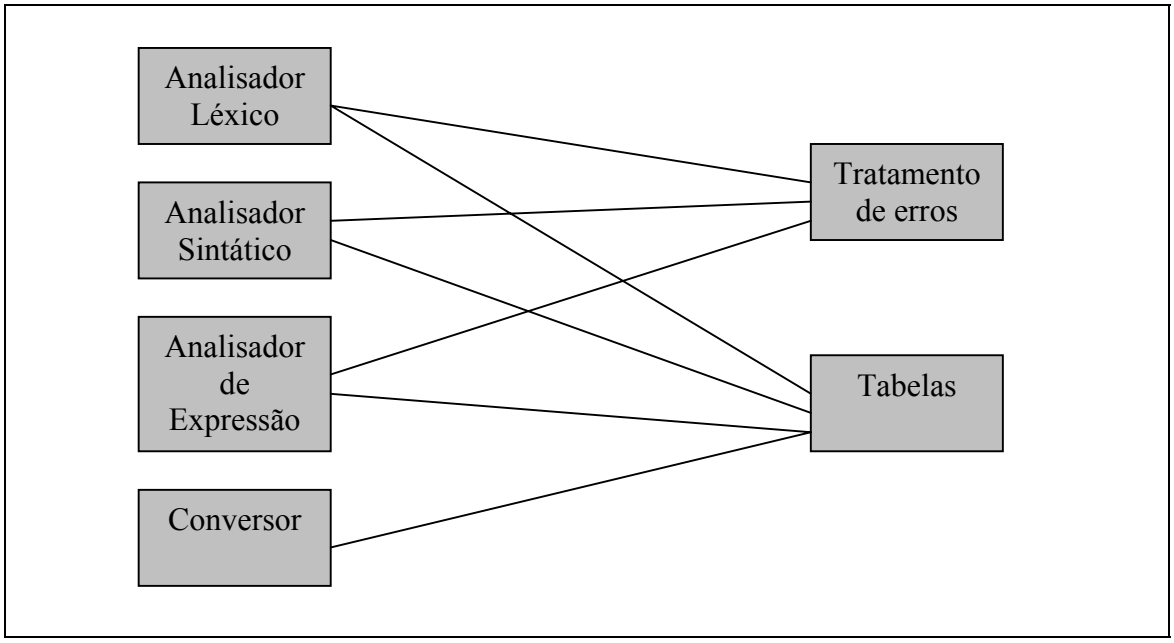

FIGURA 3.1 - Módulos que compõem o PCP.

O analisador sintático utiliza, para seu funcionamento, uma estrutura de dados em forma de pilha. Isto se deve ao fato de que se uma unidade sintática é composta de outras unidades, estas devem ser analisadas primeiramente, para em seguida voltar à análise da unidade sintática inicial. Desta forma torna-se possível se as estruturas de controle do programa foram utilizadas corretamente.

No decorrer das análises, todos os erros encontrados receberão um tratamento específico. Será mostrado para o usuário a descrição destes erros e a linha onde se localizam no código-fonte. A tabela a seguir apresenta todos os erros que poderão ser encontrados pelo PCP:

TABELA 3.1 - Erros de programação tratados pelo PCP.

Código	**Descrição**
1	ERRO de sintaxe.
2	ERRO de sintaxe em expressão , parênteses ')' esperado.
3	ERRO de sintaxe em expressão , parênteses '(' esperado.
4	ERRO de sintaxe em expressão , em '>'.
5	ERRO de sintaxe em expressão , em '<'.
6	ERRO de sintaxe em expressão , em '=' .
7	ERRO, variável não inicializada.
8	ERRO, identificador não conhecido.
9	ERRO, falta parênteses '(' depois da função.
10	ERRO, em comentário '{}'.
11	ERRO, tipos incompatíveis na expressão.
12	ERRO de sintaxe em comando IMPRIMA(); .
13	ERRO de sintaxe em comando LEIA(); .
14	ERRO de sintaxe em declaração de variável.
15	ERRO de sintaxe em declaração, '[' esperado.
16	ERRO de sintaxe em declaração, ']' esperado.
17	ERRO de sintaxe em declaração, número esperado.
18	ERRO, redeclaração de variável.
19	ERRO de sintaxe em declaração, ';' esperado.
20	ERRO de sintaxe, fim de string (') não encontrado.
21	ERRO de sintaxe, identificador esperado.
22	ADVERTENCIA: variável não inicializada.
23	ADVERTENCIA: variável não é usada no programa.
24	ERRO de sintaxe, ';' esperado.
25	ERRO de sintaxe, variável esperada.

Código	Descrição
26	ERRO de sintaxe, SE sem ENTAO.
27	ERRO, PARA sem FIM_PARA.
28	ERRO, FIM_PARA sem PARA.
29	ERRO, REPITA sem ATE_QUE.
30	ERRO , ATE_QUE sem REPITA.
31	ERRO, ENQUANTO sem FIM_ENQUANTO.
32	ERRO, FIM_ENQUANTO sem ENQUANTO.
33	ERRO, INICIO sem FIM.
34	ERRO, FIM sem INICIO.
35	ERRO, rotina principal sem 'FIM.'.
36	ERRO, FIM de rotina principal já definido.
37	ERRO, SE sem FIM_SE.
38	ERRO, FIM_SE sem SE.
39	ERRO, redeclaração de procedimento.
40	ERRO, INICIO_PROC sem FIM_PROC.
41	ERRO, FIM_PROC sem INICIO_PROC.
42	ERRO em declaração de variável, tipo não especificado.
43	ERRO de sintaxe em expressão.
44	ERRO, declaração de variáveis depois do 'INICIO'.
45	ERRO, tentativa de nova declaração com 'VARIAVEL'.
46	ERRO de sintaxe em atribuição ':='.
47	ERRO, Comprimento da String é inválido.
48	ERRO, Tipo de dado indefinido.
49	ERRO de sintaxe em comando. '(' esperado.
50	ERRO de sintaxe. 'ATE' faltando.
51	ERRO de sintaxe. 'FACA' faltando.
52	ERRO de sintaxe em comando PARA.
53	ERRO na definição do cabeçalho do programa.
54	ERRO de sintaxe, 'INICIO_PROC' faltando.
55	ERRO de sintaxe em comando 'SE'.
56	ERRO de sintaxe em comando, ';' faltando.
57	ERRO de sintaxe ou identificador desconhecido.
58	ERRO FATAL. Número de variáveis é maior que o permitido.

Depois de passar com sucesso pelas fases de análise, o PCP irá converter as instruções escritas em Portugol para a linguagem de programação Turbo Pascal.

O algoritmo poderá ser digitado em qualquer editor de textos que utilize o formato ASCII, seguindo as regras de criação de algoritmos do PCP e depois poderá ser submetido às análises.

3.1.1 Analisador léxico

A primeira fase do processo de compilação é a análise lexicográfica, que é o agrupamento de cadeias de caracteres indicando os identificadores, as constantes ou as palavras reservadas, bem como os demais símbolos e instruções componentes do programa, ou seja, nela é feita a análise das palavras do algoritmo para verificar se foram escritas corretamente .

O analisador léxico varre o programa fonte de cima para baixo, da esquerda para a direita, agrupando os símbolos de cada item léxico e determinando sua classe. Dentre as classes existentes, podemos citar algumas: os identificadores (SOMA, CONTADOR, A, etc.); as palavras reservadas (SE, ENTAO, etc.); os números inteiros (5, 100, etc.). Quando algum símbolo não é identificado em nenhuma das possíveis classes, significa que foi encontrado um erro léxico no programa fonte. Este erro será identificado de acordo com a tabela 3.1 e apresentado ao usuário.

Algumas funções do analisador léxico

O analisador léxico executa algumas operações que influem como infra-estrutura para a operação das partes do PCP mais ligadas à tradução propriamente dita do código-fonte:

- Extração e classificação de átomos - Tem a função de mapear o código-fonte em outro texto formado pelos átomos que os símbolos componentes do texto-fonte representam.
- Eliminação de delimitadores de comentários - Elimina os comentários, espaços em branco, símbolos separadores ou delimitadores.
- Conversão numérica - Converte cadeias de caracteres que representam números, através do mapeamento, para a forma inteira de representação.
- Tratamento de identificadores - Compreensão na representação dos identificadores, que é feita, em geral, com o auxílio de uma tabela de símbolos.
- Identificação de palavras reservadas - Verificar os identificadores que fazem parte de um conjunto de identificadores especiais chamados palavras reservadas.

3.1.2 Analisador sintático

A análise sintática cuida exclusivamente da forma das sentenças da linguagem, e procura, com base na sua gramática, levantar a estrutura das mesmas, verificando se a

construção da instrução ou da expressão dentro do algoritmo está de acordo com as regras pré-impostas por este projeto.

O analisador sintático verifica a formação de unidades sintáticas, do mesmo modo que é especificado pela gramática de uma língua, como o Português. Por exemplo, na seqüência A+*B deverá ser detectado a ocorrência de um operador a mais. Em outro exemplo, no comando "A=5;" é feita a seguinte análise:

- 'A' foi declarado como sendo uma variável? Se não foi, há um erro de contexto;
- Onde foi declarado 'A'?;
- Qual o tipo de 'A'?
- O tipo de 'A' é compatível com o tipo de valor que está sendo atribuído a ele? Se não for, há um erro de contexto.

O analisador sintático deve reconhecer uma cadeia de caracteres de acordo com uma gramática. Para isso ele deve:

- Ler a cadeia da esquerda para a direita, sem repetir a leitura em determinadas partes;
- Fazer o reconhecimento canônico, isto é, seguir os passos exatamente contrários aos de uma geração canônica;
- Qualquer forma sentencial da análise é constituída de duas subcadeias: a da direita, contendo partes da cadeia ainda não lidas e a da esquerda, contendo símbolos não terminais.

Algumas funções da análise sintática

- Identificador de sentenças - Aceitador de cadeias;
- Detector de erros de sintaxe - Informa a presença de erros de sintaxe e, se possível, indicando o ponto de detecção do erro;
- Comandos da ativação do analisador léxico.

Gramáticas independentes do contexto

Tradicionalmente, as gramáticas independentes do contexto são usadas como base da compilação dedicada à análise sintática. Quando uma linguagem de programação não pode ser gerada por uma gramática independente do contexto, é sempre possível descobrir uma gramática nessas condições que produza um superconjunto dessa mesma linguagem de programação desejada. Usando-se esta gramática como base da análise sintática tem-se como conseqüência que certas restrições da linguagem concreta (por exemplo, a necessidade de declaração de todos os identificadores do programa) não serão verificadas pelo *parser*[1]. No entanto, não é geralmente difícil

[1] Parser, ou análise sintática, é um processo onde a string é examinada para determinar se esta obedece um grupo de convenções estruturais explícitas na definição sintática da linguagem [AHO 74].

incorporar outras ações no pseudo-compilador (por exemplo recorrendo ao uso de uma tabela de símbolos) de modo a realizar as verificações necessárias no programa fonte.

3.1.3 Analisador de expressão

O analisador de expressão verifica se as expressões foram escritas corretamente. Para isto ele utiliza um grafo, representado pela tabela a seguir:

TABELA 3.2 - Grafo do analisador de expressão.

Nó	Símbolo	Próximo nó permitido
1	Função	4
2	Variável	56789ABDEFC
3	Número	56789ADEF
4	(	123467
5	)	56789ACDEF
6	-	1234
7	+	1234
8	*	1234
9	/	1234
A	^	1234
B	:	C
C	=	123467H
D	;	0
E	>	123467C
F	<	123467CE
N	, (vírgula)	12346789ABCDEFNH
H	‘ (apóstrofo)	

Para cada símbolo da tabela, existe uma ligação com o próximo símbolo em potencial, indicado através de um “endereço” que identifica cada nó. Por exemplo, quando for encontrado o símbolo ‘(‘ que é identificado pelo nó 4, o símbolo sucessor poderá ser qualquer um dos nós do grupo 123467, ou seja, poderá ser: função, variável, número, ‘(‘, ‘-‘ou ‘+’. Desta forma é possível detectar se a expressão foi formada corretamente ou se algum símbolo foi utilizado em local não permitido.

3.1.4 Analisador semântico

Refere-se à tradução propriamente dita do código-fonte para a forma do código-objeto.

Citamos esta etapa para completar o grupo das três grandes tarefas do compilador, porém, neste trabalho, não nos concentraremos nesta, ficando documentado para complementação posterior.

4 Estudo experimental

Visando atingir o objetivo principal deste trabalho, que é o estudo da utilização de uma ferramenta para construção de programas em Português nas disciplinas básicas de programação, escolhemos trabalhar diretamente com os alunos através de exercícios práticos em laboratório, onde eles pudessem utilizar esta ferramenta, o PCP, como apoio na resolução dos exercícios propostos.

O estudo experimental foi dividido em duas etapas. Na primeira aplicamos exercícios para alunos do primeiro e segundo anos do curso de Ciência da Computação da UNIGRAN. Na segunda, aplicamos exercícios para alunos do segundo grau da Escola Anglo Decisivo, afim de analisar o aprendizado obtido com alunos que nunca tiveram contato com programação.

Em ambas etapas, acompanhamos o desenvolvimento dos alunos na resolução dos exercícios e observamos os pontos favoráveis e desfavoráveis em relação à utilização da ferramenta PCP. Foram coletadas também opiniões dos alunos participantes em relação às características do PCP como: facilidade de utilização, qualidade da ferramenta, ambiente de programação, dificuldades e aprendizado obtido.

5 Primeira etapa

Para a primeira etapa do estudo experimental, desenvolvida no período de maio a agosto de 1999, nas dependências da UNIGRAN, foram selecionados 40 (quarenta) alunos do primeiro ano e 30 (trinta) alunos do segundo ano, todos do curso de Ciência da Computação. Estes números correspondem à metade dos alunos de cada série. Com a outra metade dos estudantes que não foram selecionados, foram aplicados os mesmos exercícios para serem resolvidos em Pascal, tornando possível fazer análises comparativas em relação ao grupo que utilizou a ferramenta PCP. Esta análise será apresentada ao final desta primeira etapa.

Durante o mês de maio de 1999, fizemos a preparação para o trabalho de pesquisa. O primeiro passo foi entrar em contato com os professores das disciplinas de programação do primeiro e segundo anos do curso de Ciência da Computação para esclarecer sobre o trabalho que se pretendia desenvolver. A resposta obtida dos professores foi muito boa, pois todos se mostraram muito interessados em colaborar com a pesquisa para conhecer a ferramenta, afim de viabilizar a utilização da ferramenta proposta como parte do conteúdo da disciplina para os próximos anos.

O passo seguinte foi selecionar alguns algoritmos que deveriam ser aplicados em laboratório como testes. Os exercícios escolhidos com a ajuda dos professores foram separados em oito grupos, que passamos a chamar de baterias de testes, contendo dois algoritmos em cada.

As atividades foram preparadas para serem realizadas nos horários normais de aulas das disciplinas de programação, tanto para a turma do primeiro ano como para a turma do segundo, sempre nas terças e quartas-feiras, respectivamente, no período matutino, das 8h às 11h30min. As aulas foram realizadas em dois laboratórios de computação contendo 32 (trinta e dois) computadores Pentium 233Mhz em cada um. Em algumas aulas foram utilizados também o retroprojetor e um projetor multimídia para apresentação de matéria teórica ou exemplos práticos.

5.1 Primeiro encontro

Nos dias 1 e 2 de junho de 1999, entramos em contato com os alunos do primeiro e segundo anos, respectivamente, do curso de Ciência da Computação, para a preparação das aulas que seriam usadas como ambiente de pesquisa. Os alunos receberam uma explicação dos objetivos da pesquisa e souberam que ela seria aplicada através de exercícios, questionários e conversas de grupo, tudo isto visando coletar informações sobre os resultados e as opiniões de cada um. Os alunos foram informados também que este trabalho de pesquisa, apesar de ter o apoio total de todos os professores de programação, não iria influenciar em nada na avaliação dos trabalhos normais do curso de Computação e que a não participação nos exercícios da pesquisa não implicaria na dispensa das aulas ou perda de créditos, isto para evitar que a participação e as opiniões obtidas fossem influenciadas pela necessidade de conseguir nota nas disciplinas ou fazer "média" com os professores.

Já neste primeiro encontro, foi feita a separação dos alunos que participariam da pesquisa utilizando o PCP e os que utilizariam o Pascal. Esta seleção se deu através de conversa com os mesmos que optaram por participar ou não da pesquisa, outros foram selecionados com base na opinião dos professores das disciplinas de programação, que procuraram indicar alguns alunos, de forma que a turma fosse formada por integrantes dos diversos níveis de conhecimento de programação, ou seja, desde pessoas que já tenham domínio sobre a disciplina de programação até os que tenham muita dificuldade. Entretanto, nenhum aluno foi obrigado a participar da pesquisa, sabendo que seria computada normalmente a freqüência na disciplina de programação, já que os exercícios deste trabalho seriam realizados concomitantemente às aulas normais da disciplina.

Como a turma do primeiro ano era composta de 40 alunos, esta foi dividida em dois laboratórios, para que cada aluno pudesse trabalhar individualmente nos equipamentos. Para poder atender a todos os alunos nos dois laboratórios, contamos com a ajuda de um aluno do quarto ano de Computação, o qual ajudou a esclarecer dúvidas técnicas do PCP enquanto o professor estivesse em outro laboratório.

5.2 Segundo encontro

No segundo encontro com os alunos, realizado nos dias 8 e 9 de junho, onde só participaram aqueles que foram selecionados, foi apresentada a ferramenta PCP. Os estudantes tiveram uma explicação sobre os comandos e funções da ferramenta, seu funcionamento e como deveriam proceder para submeter o algoritmo aos analisadores do PCP. Nesta apresentação foram usados: o quadro branco, retroprojetor para mostrar a sintaxe dos comandos e um projetor multimídia, usado para mostrar o funcionamento da ferramenta. Todos os alunos foram orientados a utilizar o Edit[2], do Ms-Dos, para digitar os seus programas na sintaxe do PCP.

5.3 Terceiro encontro

As atividades do nosso terceiro encontro, em que os alunos passariam a desenvolver os exercícios com o auxílio do PCP, aconteceram no dia 15 de junho de 1999 com a turma do primeiro ano e no dia 16 de junho com a turma do segundo ano. Todos os 140 alunos participantes da avaliação foram inicialmente questionados sobre as maiores dificuldades que sentem no processo de desenvolvimento de um algoritmo. Como opções de resposta, figuraram as seguintes alternativas:

- Têm dificuldade em entender o problema proposto no enunciado;
- Não conseguem assimilar a lógica da programação com a provável solução do problema;
- Acham difícil utilizar as estruturas básicas de controle, pois se confundem sobre o propósito de cada uma.

[2] Editor de textos que acompanha o MS-DOS

Os alunos foram orientados a escolher apenas uma das alternativas citadas. O gráfico da figura a seguir apresenta o resultado obtido neste questionário de acordo com as respostas dos alunos.

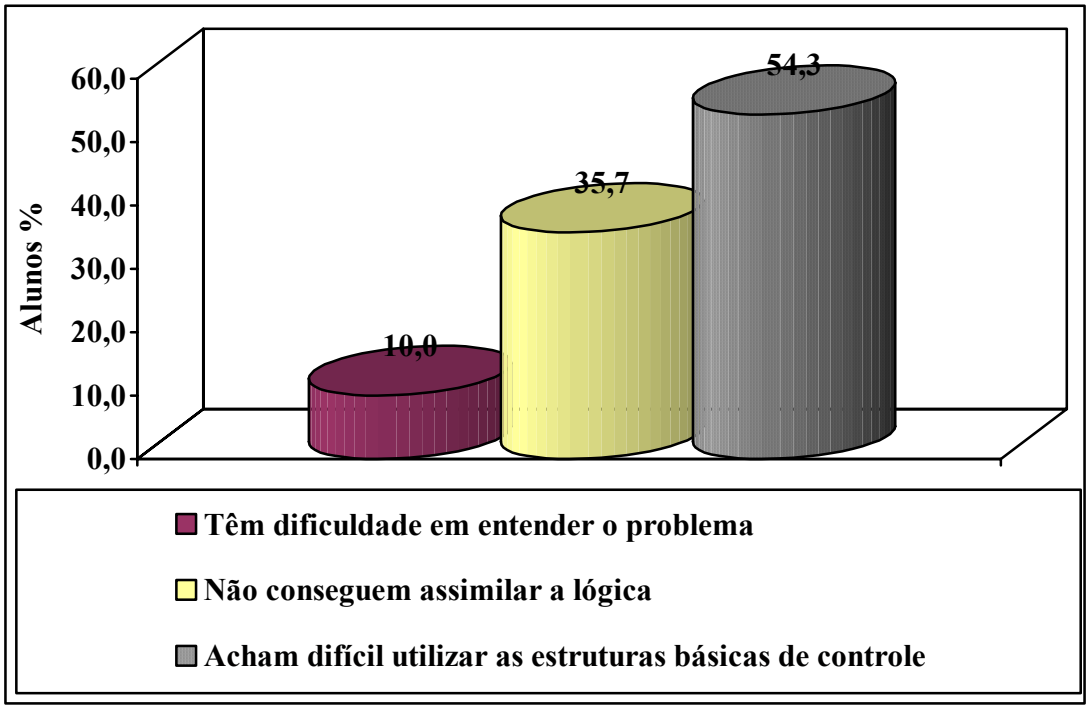

FIGURA 5.1 - Percentual de alunos em relação às dificuldades encontradas

A segunda e a terceira alternativas citadas anteriormente, segundo a opinião dos alunos que escolheram entre elas, se relacionam. Muitos chegaram a afirmar que têm dificuldades com as estruturas básicas de controle por não terem segurança sobre o funcionamento lógico das mesmas e o momento certo em que devem utilizá-las, principalmente com os alunos do segundo ano. Outros disseram que não conseguem assimilar a lógica porque têm pouco domínio sobre as estruturas básicas de controle, ou seja, um problema acaba por provocar o outro. Isto significa afirmar que, se os alunos tiverem um domínio maior sobre as estruturas básicas de controle, conseqüentemente poderão melhorar o raciocínio lógico para a solução do problema, pois estarão focalizando a maior parte do raciocínio em qual das estruturas de controle deverá usar para determinados problemas e não perderão tempo em como é cada uma delas se comporta, o que pode ser conseguido com aplicações de exercícios práticos em laboratório, e é também uma das coisas que pretendemos verificar com a utilização do PCP.

Depois de coletadas as opiniões sobre as maiores dificuldades encontradas na programação passamos a aplicar os exercícios que serviriam de base para a obtenção de resultados referentes às dificuldades ou facilidades que os alunos possam apresentar. Procuramos aplicar os mesmos exercícios para os alunos que utilizaram o PCP e os que não o utilizaram, sendo que os exercícios do pessoal do segundo ano apresentava uma complexidade um pouco maior.

A primeira bateria de exercícios foi composta dos dois algoritmos a seguir:

Exercício 1: Faça um programa para ler os três lados de um triângulo e imprimir uma mensagem informando se o mesmo é eqüilátero.

Exercício 2: Faça um programa para ler um número e um expoente e forneça o valor do número elevado a este expoente.

Para avaliar o impacto que os alunos sentiram com a nova ferramenta, após a primeira bateria de testes eles foram questionados sobre a sua preferência em relação ao modo de trabalho, utilizando ou não o PCP, ou seja, como os alunos acharam mais fácil programar. O resultado é apresentado na figura a seguir:

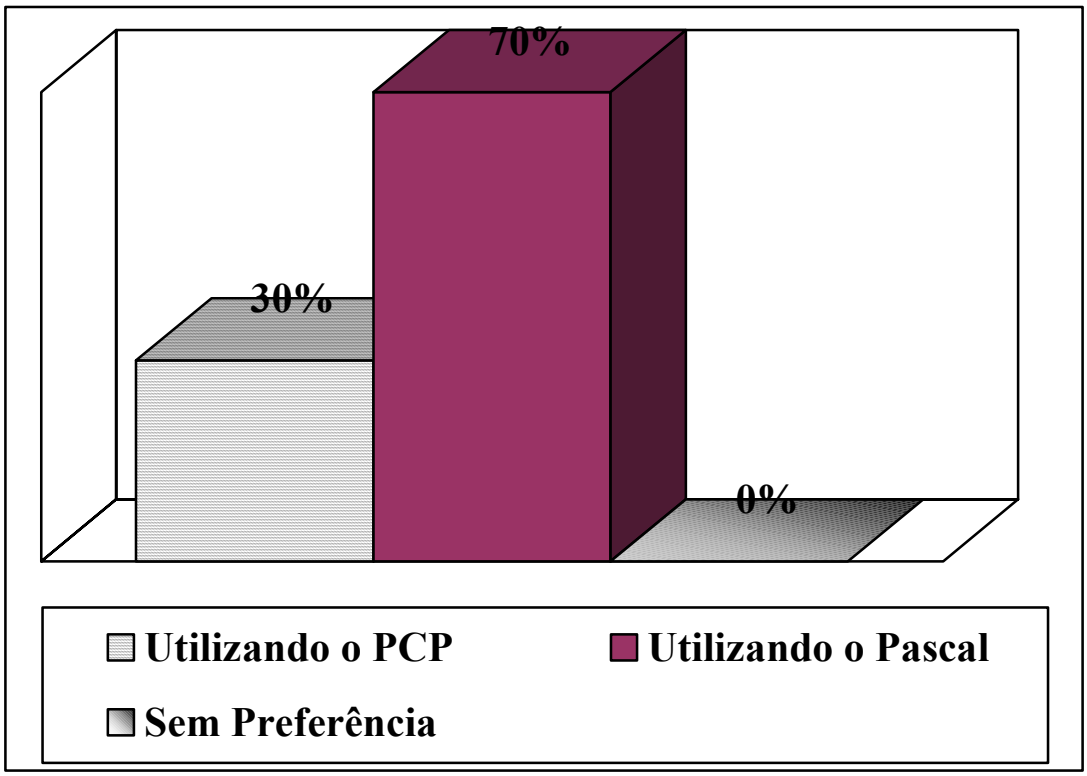

FIGURA 5.2 - Preferência dos alunos em relação ao ambiente de programação

5.4 Quarto encontro

No quarto encontro, ocorrido nos dias 22 e 23 de junho de 1999, foram passados mais dois exercícios. Neles, o objetivo foi o de mostrar o funcionamento da estrutura de repetição “Para...até...faça”.

Exercício 1: Faça um programa para ler as médias finais de 30 alunos de uma turma e informar quantos foram aprovados e quantos foram reprovados, sabendo-se que a média para aprovação é 7,0.

Exercício 2: Faça um programa para ler as quatro notas bimestrais de 30 alunos de uma turma e, no final, informar o total e o percentual de aprovados e reprovados, sabendo-se que a média para aprovação é 7,0.

Nestes dois exercícios, os alunos não demonstraram nenhuma dificuldade em relação à utilização da estrutura de repetição Para...até...faça. Na turma que resolveu os mesmos exercícios com o Pascal, os alunos tiveram poucas dificuldades em relação à utilização da estrutura de repetição For...to...do. O que mais foi motivo para perguntas foram as mensagens de erros de compilação fornecidas pelo Pascal que, para os alunos com maior dificuldade em Inglês, era motivo de constante pedido de auxílio do professor.

Comparando-se a primeira bateria de testes com a segunda, os alunos demonstraram muito mais segurança para utilizar o PCP no segundo grupo de exercícios. Eles já passaram a trabalhar com naturalidade na rotina de abrir o editor de textos, digitar o programa, salvar o texto, sair do editor e compilar o programa. O que no primeiro grupo de exercícios foi motivo para muitas reclamações.

5.5 Quinto encontro

Nosso quinto encontro aconteceu nos dias 29 e 30 de junho de 1999. Neste encontro resolvemos explorar a estrutura de repetição "Enquanto...faça". Para isto aproveitamos os exercícios da aula anterior e fizemos algumas modificações, sem avisar aos alunos que comandos eles deveriam usar. Os exercícios passados nesta etapa foram os seguintes:

Primeiro problema: Faça um programa para ler o número de matrícula, sexo e as médias finais de N alunos de uma turma e imprimir quantos foram aprovados, quantos foram reprovados, o percentual de mulheres e o percentual de homens na turma. Sabendo-se que a média para aprovação é 7,0 e que as média devem parar de ser informada quando for lido um número de matrícula igual a zero.

Segundo problema: Faça um programa para ler o número de matrícula, sexo e as quatro notas bimestrais de N alunos de uma turma e imprimir o total e o percentual de aprovados e reprovados, o total e o percentual de mulheres aprovadas e o total e percentual de homens aprovados. Sabendo-se que a média para aprovação é 7,0 e que as média devem parar de ser informada quando for lido um número de matrícula igual a zero.

Estes dois exercícios foram interessantes para perceber que os alunos que trabalharam com o PCP tiveram mais facilidade do que os alunos que utilizaram o Pascal para escolher a estrutura de controle adequada. Na turma do primeiro ano, 27 alunos dos 40 que utilizaram o PCP conseguiram usar a estrutura corretamente, sem necessitar de orientação do professor. Dos alunos que utilizaram o Pascal no primeiro ano, apenas 21 conseguiram utilizar corretamente a estrutura de repetição sem ajuda do professor. Entre os 30 alunos do segundo ano, 29 dos que usaram o PCP conseguiram escrever corretamente o programa. Na outra turma, apenas 24 deles conseguiram.

Resolvemos então questionar os alunos da turma que usou o PCP para saber qual foi a linha de raciocínio que eles usaram para decidir aplicar a estrutura "Enquanto...faça". De acordo com a resposta fornecida pelos estudantes, eles teriam que ler alguns dados "enquanto" não fosse informada uma matrícula de número zero. Esta resposta pode parecer óbvia para pessoas experientes em programação, mas os alunos costumam se confundir bastante na hora de escolher a estrutura adequada. E quando não fazem a melhor escolha, acabam utilizando artifícios criados por eles mesmos para poder ajustar a estrutura escolhida para a resolução do problema. Estes artifícios muitas vezes

funcionam, mas levam o estudante a tomar linhas de raciocínio que dificultam o trabalho de programação.

5.6 Sexto encontro

Nos dias 6 e 7 de julho, quando tivemos o sexto encontro, como estávamos entrando em período de provas bimestrais e logo em seguida entraríamos em férias, fizemos apenas revisão do que já tínhamos visto nas aulas anteriores e aproveitamos para que os alunos pudessem tirar as dúvidas existentes. Alguns apenas praticaram por com os exercícios que já tinham feito, outros ficaram para terminar ou modificar por conta própria os exercícios das aulas anteriores.

5.7 Sétimo encontro

Terminadas as férias, voltamos a nos encontrar nos dias 3 e 4 de agosto de 1999. Novamente, para que os alunos voltassem ao ritmo normal de trabalho, fizemos uma revisão rápida de alguns dos exercícios das aulas que se passaram. E seguida, foram passados dois novos exercícios, a saber:

Exercício 1: Uma loja de calçados anota diariamente o total de pares de calçados vendidos durante os 30 dias do mês. Faça um programa para ler os dados de um mês qualquer e fornecer qual o dia em que as vendas foram maiores, a quantidade de pares vendidos neste dia e o percentual das vendas deste dia em relação ao total vendido no mês.

Exercício 2: Um frigorífico deve abater um número indeterminado de animais. Para cada animal a ser abatido, deverão ser informados: número do brinco, peso em arrobas, sexo e idade em meses. Depois de abatidos N animais, pretende-se saber: O total de animais abatidos, o total de machos, o total de fêmeas, o total em arrobas, o animal mais novo e o animal mais velho. Sabe-se que para finalizar a leitura dos dados dos animais deve-se informar o número do brinco igual a zero.

Em cada um dos exercícios, os alunos deveriam usar estruturas de controle diferentes, o que não acontecia nos exercícios passados anteriormente, onde ambos eram do mesmo estilo. Como estes exercícios foram colocados no quadro ao mesmo tempo, deixamos os alunos livres para escolherem qual deles preferiam fazer primeiro. A maioria deles escolheu iniciar pelo segundo exercício.

Da duas turmas que usaram o PCP, 53 alunos escolheram as estruturas corretas para cada um dos exercícios. Nas turmas que usaram o Pascal, 46 alunos acertaram nas estruturas de controle, conforme apresentado no gráfico a seguir:

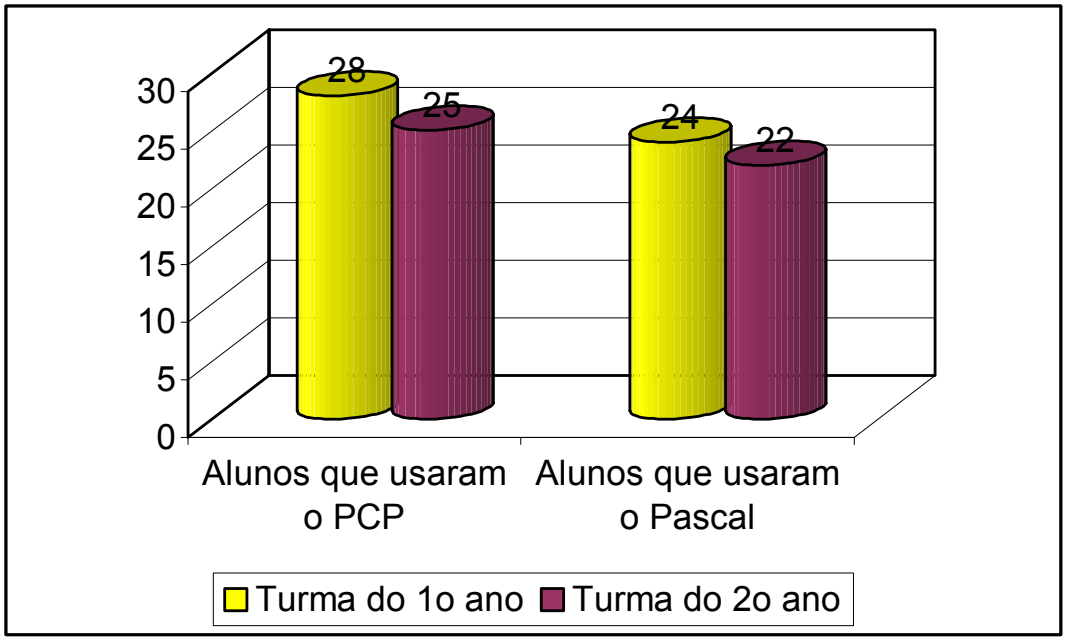

FIGURA 5.3 - Escolha das estruturas corretas pelos alunos

5.8 Oitavo encontro

O encontro seguinte aconteceu nos dias 10 e 11 de agosto de 1999. Nele discutimos sobre algumas dúvidas que apareceram na aula anterior. Numa conversa informal, os alunos relataram suas maiores dificuldades para resolver os exercícios até então. O manifesto dos alunos que utilizaram o Pascal foi maior em relação à decisão sobre qual a estrutura correta a ser utilizada. O segundo ponto de quantidade de reclamações foi em relação à interpretação e correção dos erros apresentados pelo compilador. Já na turma que usou o PCP, todos os alunos que sentiram alguma dificuldade, relataram que elas aconteceram na decisão das estruturas corretas e na criação de variáveis. Nenhum dos alunos afirmou ter dificuldades em relação às mensagens de erro do PCP. Depois deste breve bate-papo, passamos dois exercícios.

Primeiro exercício: Uma companhia de saneamento básico precisa de um programa para calcular o valor que deverá cobrar de seus clientes pela água consumida no mês, de forma que o usuário que gastar mais pagará um valor maior por m^3 de água. Segundo a empresa, o cálculo deverá ser feito da seguinte forma:

- para consumo até 3 m^3 – taxa mínima no valor de R$ 5,00
- para consumo maior que 3 m^3 e menor que 5 m^3 – R$ 1,50 por m^3
- para consumo maior que 5 m^3 e menor que 10 m^3 – R$ 1,70 por m^3
- para consumo acima de 10 m^3 – R$ 2,00 por m^3

Para cada cliente deverão ser informados o número do hidrômetro e o total de m^3 de água consumidos durante o mês. Sabe-se que devem ser lidos dados de clientes até que se informe um número do hidrômetro igual a zero. No final, a empresa pretende saber: o total geral de m^3 de água consumidos no mês, o valor total arrecadado e a média de consumo em m^3.

Segundo exercício: Numa empresa, cada funcionário recebe mensalmente o salário referente às 44 hs semanais trabalhadas. Para cada hora trabalhada, o funcionário recebe R$ 5,00, porém, se trabalhar horas extras, receberá um acréscimo de R$ 1,00 para cada hora extra. Para cada funcionário, sabe-se o número do registro, o total de horas trabalhadas e o sexo. Faça um programa que leia estes dados para N funcionários até que se informe um registro igual a zero. O programa deverá emitir para cada funcionário: O número do registro, o total de horas normais trabalhadas, o salário correspondente às horas normais, o total de horas extras trabalhadas e o salário correspondente às horas extras. No final, o programa deverá mostrar: O total geral que a empresa deverá pagar pelas horas normais trabalhadas, o total geral pelas horas extras, e o total de homens e mulheres que trabalham na empresa.

Nestes exercícios, apenas quatro alunos do primeiro ano da turma que usou o PCP apresentaram dificuldades em relação a quais comandos deveriam empregar. No segundo ano que usou o PCP, nenhum aluno apresentou dificuldades na resolução do problema. Numa conversa ao final dos exercícios, os estudantes disseram que, baseados nos trabalhos anteriores, puderam resolver sem problemas. Com a turma que usou o Pascal, nove alunos tiveram dificuldades no primeiro ano para a utilização dos comandos e com a estruturação do programa, além de várias perguntas sobre as mensagens de erro. No segundo ano, apenas três tiveram dificuldades para resolver os exercícios; segundo eles, o motivo foi a dificuldade em escolher e montar corretamente as estruturas de controle adequadas. Houveram também algumas poucas perguntas sobre as mensagens de erro de compilação.

5.9 Nono encontro

O encontro seguinte ocorreu nos dias 17 e 18 de agosto de 1999. Neste encontro, fizemos os seguintes exercícios:

Exercício 1: Dados o gabarito para dez questões de uma prova (que podem variar de A a D) e as respostas dos N alunos de uma turma de Computação. Se o aluno acertar entre quatro e seis questões, estará de exame, se acertar sete ou mais estará aprovado, se acertar menos de 4 perguntas estará reprovado. Para cada aluno informado, imprimir as mensagens:

- ‘APROVADO’ – para os alunos que acertarem todas as questões;
- ‘EXAME’- para os alunos que acertarem no mínimo 3 questões;
- ‘REPROVADO’- para os alunos que errarem 3 ou mais questões.

Se o aluno ficar de exame, deve ser informado ainda sua nota na prova de exame. Neste último caso, para ser aprovado, ele deverá conseguir uma nota que, somada com sua nota anterior, resulte em dez. Ao final, imprima o total e o percentual para: alunos aprovados, reprovados e de exame.

Exercício 2: Dados 10 valores inteiros, faça um programa para classificá-los utilizando o método *Buble-Sort*, listando os valores ordenados ao final.

Nestes exercícios, os alunos que utilizaram o PCP já não fizeram perguntas querendo saber se as estruturas estavam corretas, mas perguntaram se as estavam utilizando da melhor maneira possível, o que para nós significou uma melhora considerável

comparando-se com a turma do Pascal, que ainda apresentava muitas dúvidas em relação à estruturação do programa.

À medida que os alunos iam se acostumando com o PCP, o trabalho de editar o programa e compilá-lo em programas diferentes não mais era motivo de reclamação, pois estes estavam concentrados em desenvolver a lógica do programa e perdiam menos tempo em entender a ferramenta de trabalho. No caso da turma do Pascal, as dúvidas em relação a mensagens e comandos continuava freqüente. Já tinha diminuído em quantidade de alunos que faziam este tipo de pergunta, mas muitos não conseguiam se lembrar da tradução e ficavam perguntando ao professor. Em conseqüência disto, uma quantidade maior de alunos das turmas do PCP conseguiam terminar os exercícios à frente dos alunos das turmas de Pascal.

5.10 Décimo encontro

No encontro dos dias 24 e 25 de agosto de 1999, aproveitamos para debater com os alunos e tirar possíveis dúvidas que apareciam até então. Aproveitamos o tempo também para que alguns pudessem terminar ou melhorar os exercícios das aulas anteriores. Esta foi apenas uma aula de revisão.

5.11 Décimo primeiro encontro

O encontro seguinte ocorreu nos dias 31 de agosto e 1 de setembro de 1999. Neste encontro, fizemos os seguintes exercícios:

Exercício 1: O Governo Federal pretende distribuir pessoas por todas as regiões do Brasil afim de colherem informações sobre as famílias brasileiras. Para isto, em cada casa visitada do país deverá ser preenchido um questionário com 20 perguntas relativas a, por exemplo, a renda "per capta" familiar, sendo cada pergunta composta por 8 opções de resposta (apenas uma resposta pode ser escolhida em cada pergunta). O governo pretende contratar você para desenvolver um programa que apure o resultado da pesquisa, devendo para isto:

- Ler as respostas dos questionários de cada uma das N famílias pesquisadas;
- Fornecer um relatório final contendo o total e o percentual de respostas fornecidas em cada uma das 8 opções de cada pergunta.

O relatório deverá ser apresentado na tela com o mesmo lay-out a seguir.

Pergunta 1 - Responderam A: 999 pessoas = 999,99%
Responderam B: 999 pessoas = 999,99%
...
Responderam H: 999 pessoas = 999,99%

Pergunta 2 - Responderam A: 999 pessoas = 999,99%
Responderam B: 999 pessoas = 999,99%

...

Responderam H: 999 pessoas = 999,99%

... e assim por diante, até a 20ª pergunta.

Exercício 2: Faça um programa para calcular a expressão a seguir:

$$S = \frac{2^0}{0!} + \frac{2^1}{1!} - \frac{2^2}{2!} + \frac{2^3}{3!} - \frac{2^4}{4!} + \ldots + \frac{2^n}{n!} .$$

- Considerar apenas os 30 primeiros termos da série.

Estes dois exercícios geraram muito debate entre os alunos, pois uns queriam saber como outros resolveriam os problemas. Todas as turmas demonstraram dificuldades em resolver estes exercícios, principalmente no segundo, que exigiu uma maior concentração e criatividade. A turma que programou em Pascal se complicou para resolver o problema da troca de sinais, muitos pensaram que deveriam usar algum comando da linguagem para fazer a troca.

5.12 Décimo segundo encontro

Novamente nos encontrados nos dias 7 e 8 de setembro de 1999. Nestes dias, dois novos exercícios foram passados aos alunos. Estes foram os últimos exercícios a serem desenvolvidos como parte da primeira etapa da pesquisa. Os algoritmos pedidos foram os seguintes:

Exercício 1: Para um grupo de Doutores em Ciência da Computação foi distribuído um questionário contendo dez quesitos. Cada quesito possuía cinco opções de respostas. Deseja-se saber o total e o percentual de Doutores que escolheram cada opção de resposta.

Exercício 2: Dada uma matriz A de dimensão 3x4 e uma matriz B de dimensão 4x3, faça um programa para calcular a matriz C de dimensão 4x4 tal que C é a multiplicação de A por B. Sabe-se que para multiplicar uma matriz por outra, deve-se multiplicar cada linha da primeira por cada coluna da Segunda, conforme o exemplo abaixo:

$$\begin{vmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \\ a41 & a42 & a43 \end{vmatrix} \quad \begin{vmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \end{vmatrix} \quad = \quad \begin{vmatrix} c11 & c12 & c13 & c14 \\ c21 & c22 & c23 & c24 \\ c31 & c32 & c33 & c34 \\ c41 & c42 & c43 & c44 \end{vmatrix}$$

A x B = C

Onde:

c11 = (a11xb11)+(a12xb21)+(a13xb31)

c12 = (a11xb12)+(a12xb22)+(a13xb32)

...

c43 = (a41xb13)+(a42xb23)+(a43xb33)

c44 = (a41xb14)+(a42xb24)+(a43xb34)

Os valores de M e N deverão ser informados pelo usuário, porém só deverão ser aceitos valores no máximo até 10 (dez) e o programa deverá mostrar uma mensagem de erro caso o usuário tente informar valores maiores.

Com esta última bateria de exercícios, completamos o total de oito previstas. Na turma que usou o Pascal, ninguém conseguir terminar os dois exercícios. Na turma do PCP, seis alunos conseguiram terminar, sendo que dois deles usaram a mesma lógica que esperava o professor.

Pela quantidade de alunos que não terminaram os exercícios, resolvemos finalizar nossos encontros na semana seguinte, nos dias 14 e 15 de setembro de 1999. Nestes dias, além de finalizar os exercícios, debatemos sobre as principais dificuldades das turmas. Com esta conversa, pudemos notar que as dificuldades dos alunos que programaram em Inglês eram muito maiores do que os outros.

5.13 Conclusões da primeira etapa

Como os níveis de conhecimento entre os alunos era muito variável, optamos por selecionar algoritmos que não fossem muito simples nem complicados demais. Dos alunos que usaram o PCP, 92,8% conseguiram finalizar todos os exercícios, já dos alunos que usaram o Pascal, 77,1% conseguiram terminar todos os exercícios. Notou-se inclusive que nas aulas com o Pascal, muitos alunos pediam ajuda sobre a resolução dos algoritmos para os colegas próximos. Nas aulas com o PCP, esses pedidos de ajuda eram menos freqüentes entre os alunos.

Notamos também uma diferença entre as turmas no que se refere a escrever e testar os programas até ficarem prontos. Em conversa com os alunos que usaram o PCP, eles disseram que se sentiam mais seguros nas soluções adotadas por estarem programando em Português.

À medida que os exercícios iam sendo resolvidos, os alunos gostavam mais de usar o PCP. Uma aluna que não tinha muito interesse na disciplina de programação relatou: “Até que enfim estou conseguindo fazer um programa funcionar”. E explicou que não se sentia motivada com as aulas por não conseguir assimilar as estruturas de controle necessárias à resolução do problema. Esta afirmação foi confirmada por duas outras colegas que disseram pensar da mesma maneira, e completaram: “...programando em Português talvez a gente consiga tirar nota boa na prova”.

O resultado da análise de preferência dos alunos em relação ao ambiente de programação é apresentado na figura a seguir:

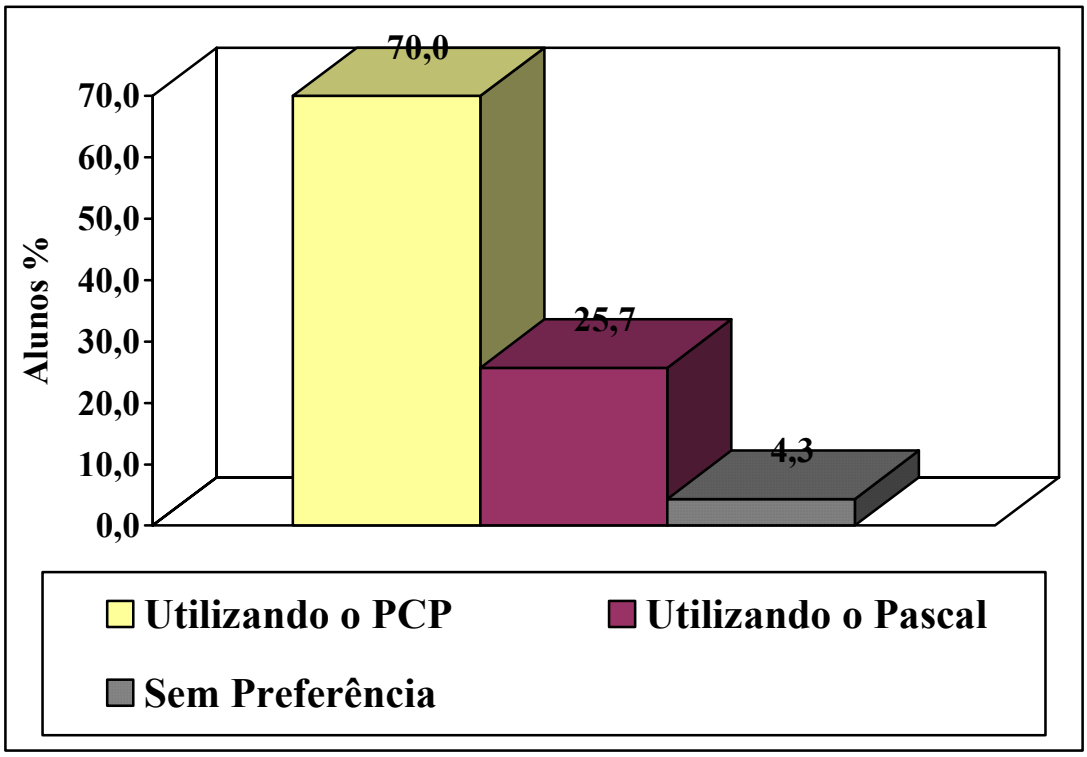

FIGURA 5.4 - Preferência de ambiente de programação

Em relação ao aprendizado adquirido, a maioria dos alunos (71,4%) achou que com a ferramenta ficou mais fácil programar, porque podem focalizar seu raciocínio especificamente no problema a ser resolvido sem se preocupar ou fazer confusão com o significado dos comandos, estruturas e, principalmente, mensagens de erro que, segundo eles, é a maior causa de perda de tempo de programação, pelo fato de não entenderem as mensagens que são todas em inglês na linguagem Pascal. O grupo citou ainda que acham muito importante que a ferramenta passe a fazer parte do conteúdo programático das disciplinas de programação como ferramenta de apoio à aprendizagem, pelos mesmos motivos citados anteriormente.

Um grupo de 25,7% dos alunos acharam que a utilização do PCP nos exercícios propostos apenas completou o aprendizado. Porém, nem todos gostaram de utilizar o PCP; 2,9% dos alunos disseram que a ferramenta não ajudou a aprimorar nada, criticando as limitações desta, o que provocou um debate entre os próprios alunos, onde uns defendiam o uso da ferramenta, outros atacavam para “proteger” o Pascal.

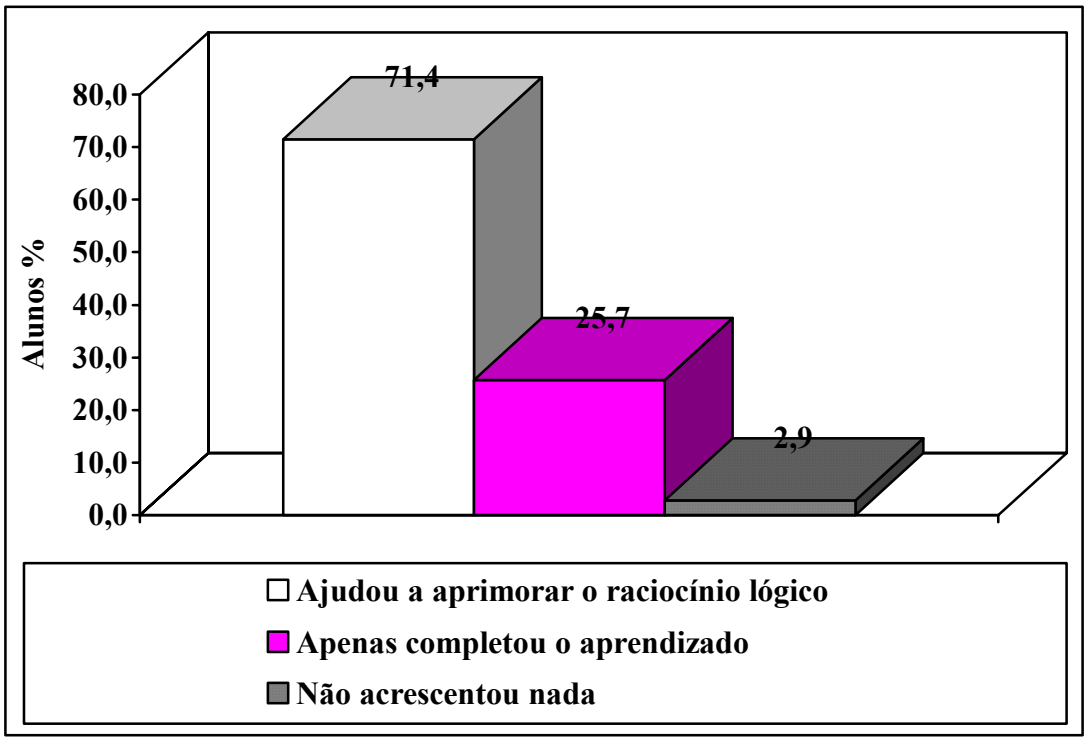

FIGURA 5.5 - Opinião dos alunos em relação ao aprendizado obtido com o PCP

Foram aplicados os mesmos exercícios propostos tanto para a turma que utilizou o PCP como para a turma que utilizou o Pascal, mesmo assim houveram diferenças em relação ao tempo gasto nas duas turmas. Fizemos uma comparação do resultado referente ao tempo gasto para resolver o segundo problema da primeira bateria entre as turmas que usaram e que não usaram o PCP. Os resultados podem ser observados nas duas figuras que se seguem:

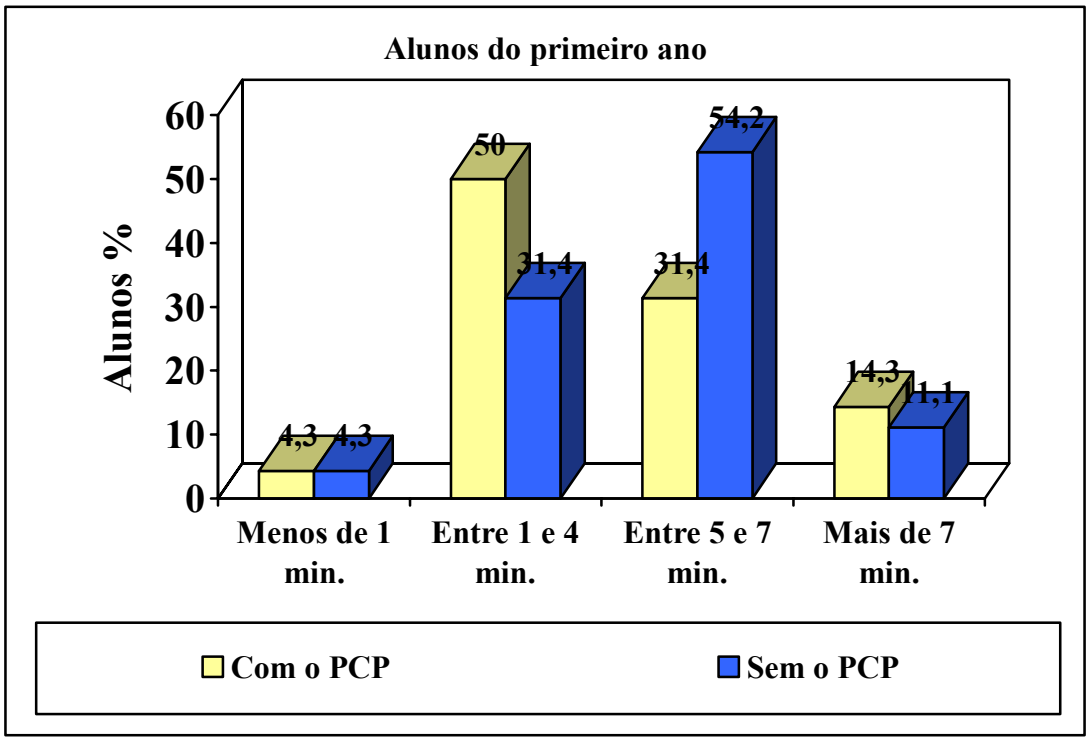

FIGURA 5.6 - Comparação das turmas do primeiro ano na primeira bateria

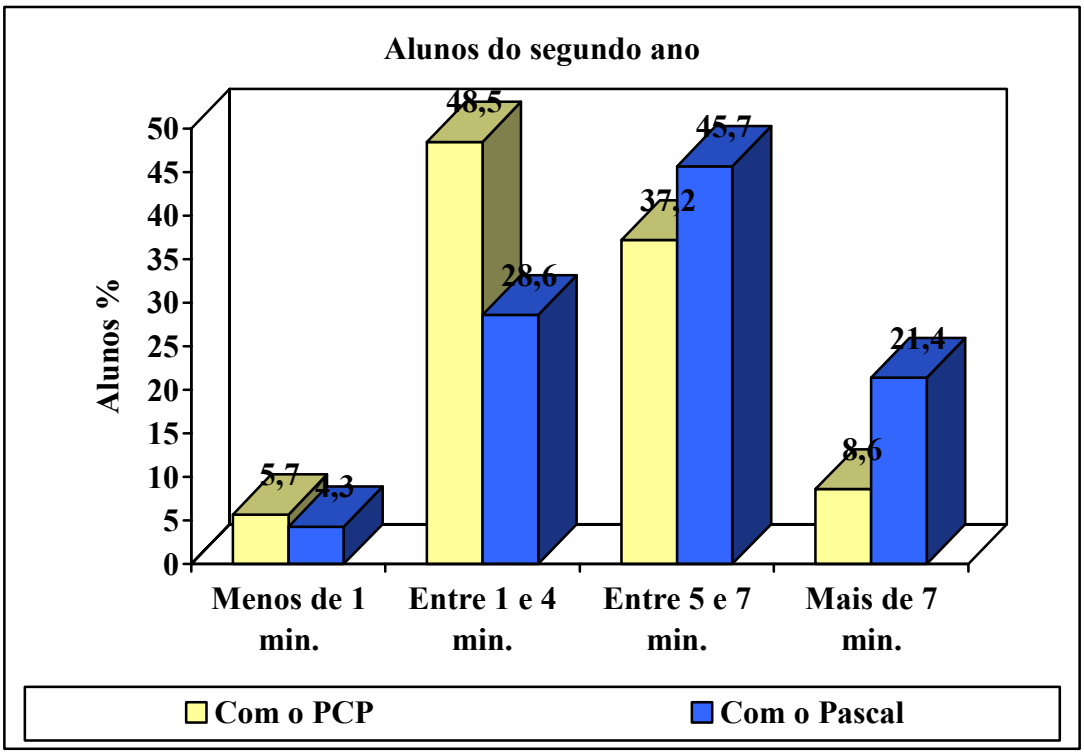

FIGURA 5.7 - Tempo gasto pelas das turmas do segundo ano na primeira bateria

Entre uma bateria de testes e outra, houve diminuição do tempo gasto para resolver problemas com níveis de dificuldade parecidos. Também o número de solicitações de ajuda do professor foram menores. O fato de os alunos entenderem melhor os comandos da linguagem fazia com que eles ficassem mais concentrados no seu exercício e pedissem menos ajuda aos colegas ao lado.

Durante a utilização da ferramenta PCP pelos alunos, alguns problemas técnicos foram encontrados. Foi apontada a questão de que na escrita do programa em PCP, seria possível finalizar a linha de comandos de uma estrutura "Se...entao" com um ponto e vírgula, mesmo quando vinha seguida do comando "Senao", sendo que na linguagem Pascal isto não era permitido. Para que os alunos ficassem o mais próximo possível do ambiente de programação de se depariam depois de finalizados os trabalhos com o PCP, este problema foi corrigido, passando a atuar na estrutura "Se...entao...Senao" da mesma forma que o Pascal, evitando assim que os alunos deixassem de focalizar o aprendizado para perder tempo com comparações técnicas entre o PCP e o Pascal.

A quantidade de solicitações de ajuda ao professor por causa das mensagens de erro diminuiu consideravelmente, comparando-se com os alunos da turma que não utilizou a ferramenta. Por estarem lendo mensagens em Português, os próprios alunos se sentiram mais seguros para procurar os problemas no algoritmo, pois já não tinham mais dúvidas em relação ao motivo do erro, passando a se preocupar somente em achar a solução para o erro apresentado, ou seja, na lógica do problema ou na sintaxe do algoritmo.

Solicitamos que os alunos apontassem os pontos positivos e os negativos em relação à ferramenta utilizada e o aprendizado proporcionado por ela. Na opinião dos alunos de ambas as turmas, os aspectos positivos foram:

- Maior facilidade na formulação do programa usando o Português ao invés do Inglês;
- Acharam mais fácil aprender as estruturas básicas de controle, uma vez que não precisam se preocupar com a tradução dos comandos;
- As mensagens de erro são todas fáceis de entender e não correm o risco de se confundirem quando ao erro ocorrido;
- Não sentiram nenhuma dificuldade no aprendizado da ferramenta.

Já em relação aos pontos negativos, foram citados dois itens:

- A falta de um ambiente integrado torna mais lento o trabalho de escrever e compilar o programa;
- A quantidade de comandos e funções disponíveis na ferramenta ainda são limitados, como por exemplo, comandos de posicionamento em tela.

Sobre estes pontos negativos citados, somente os alunos do primeiro ano apontaram o primeiro item (falta do ambiente integrado) como sendo um fator que dificultou o trabalho, já os alunos do segundo ano acharam que este item não deve ser considerado um ponto negativo pois eles se sentiram muito à vontade para utilizarem o editor de textos de sua preferência para escrever os programas.

O segundo ponto negativo que diz respeito à limitação da ferramenta, foi apontado na grande maioria pelos alunos do segundo ano, principalmente no referente a comandos de tela como: goxoxy, textcolor e textbackground. Estes comandos fazem parte da biblioteca Pascal e servem, respectivamente, para posicionar o cursor na tela, mudar a cor do texto (cor de frente) e mudar a cor de fundo. A este problema apontado por eles foi dada a explicação de que esta versão do PCP é experimental e que muitas melhorias

poderão ser implementadas nas próximas versões, caso fique constatado que o mesmo poderá vir a ajudar os alunos no aprendizado de programação. Esta afirmação foi apoiada inclusive pelo coordenador do curso de Computação, que mostrou um grande interesse em saber o resultado desta pesquisa para estudar a possibilidade da utilização desta ferramenta como parte das aulas de programação das próximas turmas.

O coordenador afirmou também que no próximo ano pretende montar um grupo de estudo formado por professores e alunos para trabalhar nas melhorias desta ferramenta, enfocando principalmente a criação de um ambiente integrado, conforme foi citado pelos próprios alunos como um fator importante para a ferramenta. Este ambiente incluiria, a princípio, um editor de programas e a chamada ao compilador do Pascal para gerar o código executável, o que evitaria que os alunos tivessem que ficar digitando comandos do sistema operacional para editar e compilar o programa.

6 Segunda etapa

O objetivo desta segunda etapa foi aplicar a ferramenta PCP com alunos que não tinham nenhum conhecimento de programação de computadores, para que fosse possível avaliar o aprendizado alcançado em comparação com o aprendizado obtido utilizando-se a linguagem Pascal.

Para a segunda etapa do estudo experimental, foram selecionados 40 alunos da Escola de 2º grau Anglo Decisivo. A seleção se deu através de convite verbal feito aos alunos da segunda e terceira séries no dia 9 de novembro de 1999 durante o intervalo de uma aula e outra. Os alunos que se dispuseram a participar da pesquisa se mostraram bastante interessados, uma vez que teriam aulas de programação gratuita. Quase todos os alunos selecionados já tiveram algum contato com o computador. Alguns em casa, outros no escritório do pai ou na casa de amigos, na maioria para ter acesso à internet ou jogos, mas nenhum deles havia se interessado em aprender a programar.

Todos os alunos teriam aulas de programação, porém, para que a pesquisa pudesse realmente avaliar a capacidade de ajuda no aprendizado proporcionada pelo PCP, a turma de quarenta alunos foi dividida em duas turmas de vinte, sendo que a primeira teria aulas com a utilização da ferramenta e a outra trabalharia diretamente com a linguagem Pascal. Deste modo seria possível fazer uma comparação entre as duas turmas, visto que em ambas os alunos nada sabiam sobre programação.

Esta etapa estava prevista para acontecer nas primeiras semanas do mês de janeiro de 2000, mais precisamente do dia 5 ao dia 19, período em que os alunos poderiam dispor de tempo sem que as aulas fossem prejudicadas. Porém, no dia 17 de novembro, quando fomos avisar aos alunos sobre o calendário das aulas, dezesseis dos quarenta alunos selecionados alegaram que não poderiam estar presentes porque neste período estariam viajando de férias com os pais ou amigos, outros disseram que não tinham certeza se iriam viajar ou não, mas fizeram questão de afirmar que isto não significava desistência do "curso". Por este motivo, não foi possível realizar as aulas no mês de janeiro, as quais tiveram que ser transferidas para as férias de julho de 2000, único período em que os alunos teriam duas semanas livres para poderem participar do trabalho. Nesta segunda hipótese, ficou marcado então que os trabalhos seriam realizados do dia 3 ao dia 14 de julho de 2000, período em que todos os quarenta alunos selecionados se comprometeram a estarem presentes para as aulas.

6.1 Primeiro encontro

No dia 3 de julho, nos encontramos pela primeira vez para iniciarmos nossa atividade. As aulas foram dadas todos os dias, de segunda a sexta-feira, no período da manhã, das 8hs às 11hs, para os alunos que utilizaram a ferramenta e no período da tarde, das 14hs às 17hs, para os alunos que utilizaram o Pascal. As aulas foram realizadas no laboratório de informática da UNIGRAN, composto por trinta e dois computadores Pentium 233Mhz, retroprojetor e projetor multimídia. Como a primeira aula prevista seria usada para uma explanação geral sobre algoritmos, todos os 40 alunos estiveram juntos na mesma sala no período da manhã.

Nesta primeira aula, foram passados conceitos gerais sobre programação, alguma matéria sobre as linguagens de programação existentes e os seus propósitos, para que os alunos começassem a se familiarizar com o ritmo das aulas de informática e o objetivo das mesmas.

6.2 Segundo encontro

No segundo dia de aula, os alunos já vieram em turmas separadas, conforme havia sido combinado com os mesmos. Para ambas as turmas, foram passados os conceitos da estrutura de um programa, declaração de variáveis, os tipos de dados e as estruturas básicas de controle: Se...então...senão, enquanto...faça, repita...até e para...até...faça. Para a turma da manhã, todos os comandos foram passados em Português e para a turma da tarde, em Inglês. Para cada uma das estruturas de controle, eram passados trechos de programas exemplificando a utilização das referidas estruturas. Neste dia, os alunos fizeram dois pequenos exercícios sem o uso do computador, apenas para fixar melhor a matéria que passariam a utilizar.

6.3 Terceiro encontro

A partir do terceiro dia de aula, passamos a fazer exercícios práticos em laboratório. As atividades para as turmas da manhã e da tarde eram as mesmas, com a diferença de que a primeira utilizou o PCP e a segunda, o Turbo Pascal. Este primeiro contato com o computador serviu para apresentar aos alunos o ambiente de programação que passariam a trabalhar a partir desta data. Os alunos da turma da manhã usaram o editor de textos Edit do Ms-Dos para digitar os programas na sintaxe do PCP, pois a ferramenta PCP não possui um ambiente com editor de programas. Para que eles se sentissem mais seguros, nesta aula todos trabalharam em grupos de dois em cada computador. Foram passados então dois enunciados de problemas:

Primeiro problema: Faça um programa que leia a distância percorrida por um carro em Km e o total de litros de combustível gasto. Calcule e imprima quantos Km o carro percorre com 1 litro do combustível.

Segundo problema: Faça um programa que leia os três lados de um triângulo e informe se o mesmo é eqüilátero.

No primeiro exercício, os alunos de ambas as turmas tiveram problemas para se familiarizarem com o ambiente de programação, no entanto, as dificuldades tinham pouco a ver com o programa propriamente dito. Praticamente não houve diferença de desempenho entre as turmas que utilizaram o PCP e o Pascal. Em ambas as turmas notava-se uma euforia dos estudantes quando conseguiam fazer o programa funcionar. No segundo exercício, os problemas em relação ao ambiente diminuíram bastante. Na primeira turma, 12 alunos conseguiram terminar o exercício, na segunda turma apenas 10 conseguiram terminar.

Nesta primeira análise, foi interessante notar que na turma dos alunos que utilizaram a ferramenta PCP (primeira turma), a dificuldade encontrada pelos estudantes que não conseguiram terminar os exercícios aconteceu por causa do uso incorreto da estrutura

"Se..Então". Já na turma dos alunos que utilizaram o Pascal, a maioria que não conseguiu terminar teve problemas em escolher qual das estruturas de controle deveria usar, alegando dificuldades em saber para quê servia cada uma. Como os dois exercícios não necessitavam de muitas estruturas de controle, as dúvidas em relação às mensagens de erro foram poucas. Depois que a maioria dos alunos havia terminado os exercícios, eles foram corrigidos no quadro para esclarecer as dúvidas.

6.4 Quarto encontro

No quarto dia de aula, ocorrido em 6 de julho de 2000, passamos mais dois exercícios, mas neste caso, os alunos passaram a trabalhar individualmente. Os exercícios passados neste dia precisavam de mais de uma estrutura de controle. Os alunos não foram orientados a utilizar uma ou outra estrutura, eles deveriam decidir qual estrutura achavam melhor baseados nas explicações e exemplos passados nas primeiras aulas. Os exercícios desta aula foram os seguintes:

Primeiro problema: Faça um programa para ler as médias finais de 30 alunos de uma turma e informar quantos foram aprovados e quantos foram reprovados, sabendo-se que a média para aprovação é 7,0.

Segundo problema: Faça um programa para ler as quatro notas bimestrais de 30 alunos de uma turma e, no final, informar quantos estão aprovados e quantos estão reprovados, sabendo-se que a média para aprovação é 7,0.

No primeiro problema foram dados vinte minutos para que os alunos pudessem resolvê-lo. A análise do resultado deste exercício pode ser visto no gráfico apresentado na figura a seguir:

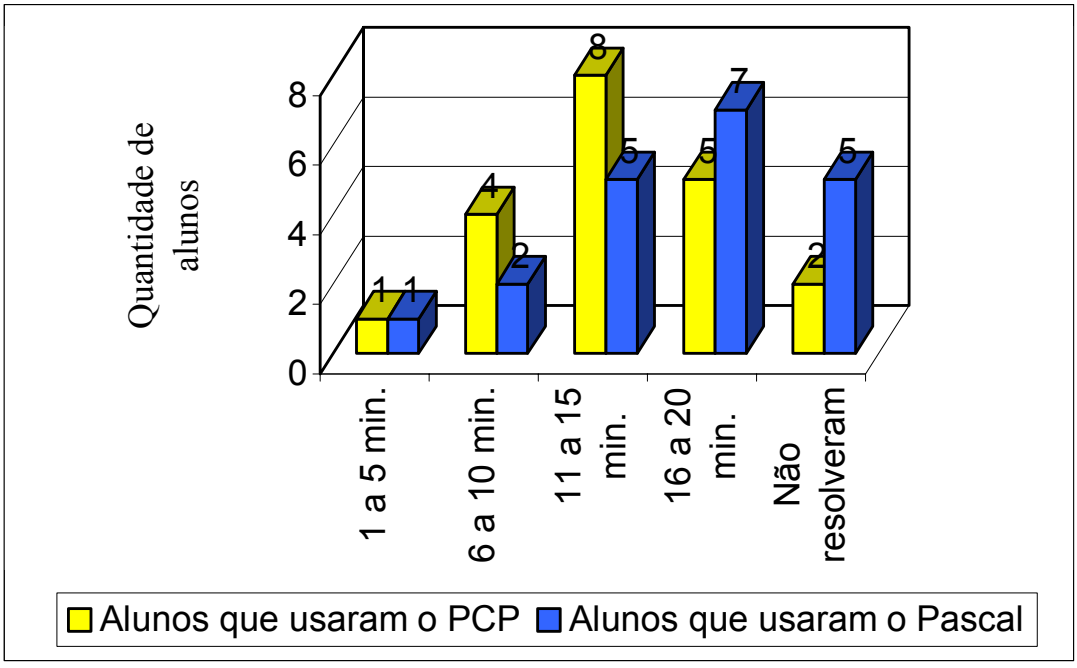

FIGURA 6.1 - Tempo gasto pelos alunos para resolver um algoritmo

Os alunos das duas turmas tiveram as mesmas dificuldades em relação ao entendimento do enunciado do problema. Os da segunda turma fizeram mais perguntas relacionadas às mensagens de erro; geralmente estas perguntas começavam com as dúvidas sobre o que a mensagem significava, para depois procurarem saber o que deveriam fazer para resolver o problema.

6.5 Quinto encontro

No quinto dia de aula, 7 de julho de 2000, foram utilizados exercícios para aplicação da estrutura "Enquanto..faça". Um pequeno exemplo foi passado no quadro para que os alunos pudessem se sentir mais familiarizados com os comandos. Os exercícios passados foram parecidos com os do dia anterior, porém sofreram uma alteração no enunciado afim de que os alunos usassem uma outra estrutura de repetição. Os algoritmos foram os seguintes:

Primeiro problema: Faça um programa para ler o número de matrícula, sexo e as médias finais de N alunos de uma turma e imprimir quantos foram aprovados, quantos foram reprovados, o percentual de mulheres e o percentual de homens na turma. Sabe-se que a média para aprovação é 7,0 e que as média devem parar de ser informadas quando for lido um número de matrícula igual a zero.

Segundo problema: Faça um programa para ler o número de matrícula, o sexo e as quatro notas bimestrais de N alunos de uma turma. Sabendo-se que a média para aprovação é 7,0 e que as média devem parar de ser informadas quando for lido um número de matrícula igual a zero, imprimir o total e o percentual de aprovados e reprovados, o total e o percentual de mulheres aprovadas e o total e percentual de homens aprovados.

No final da aula em cada turma, quando os alunos já tinham resolvido os dois exercícios, fizemos uma roda de "bate papo" para discutirmos as maiores dificuldades encontradas e podermos fazer uma comparação dos pontos mais críticos entre as duas turmas. Para todos os alunos foi feita a seguinte pergunta:

- Qual a maior dificuldade que você encontrou para resolver os problemas propostos até agora:

- Tenho dificuldade em entender o enunciado do problema
- Acho difícil utilizar as estruturas básicas de controle
- Não consigo montar a lógica para o algoritmo
- Não consigo entender as mensagens de erro de compilação
- Não senti nenhuma dificuldade

Pedimos aos alunos que escolhessem apenas uma das alternativas de resposta que mais lhe conviessem. O resultado das respostas obtidas podem ser vistas dois gráficos seguintes:

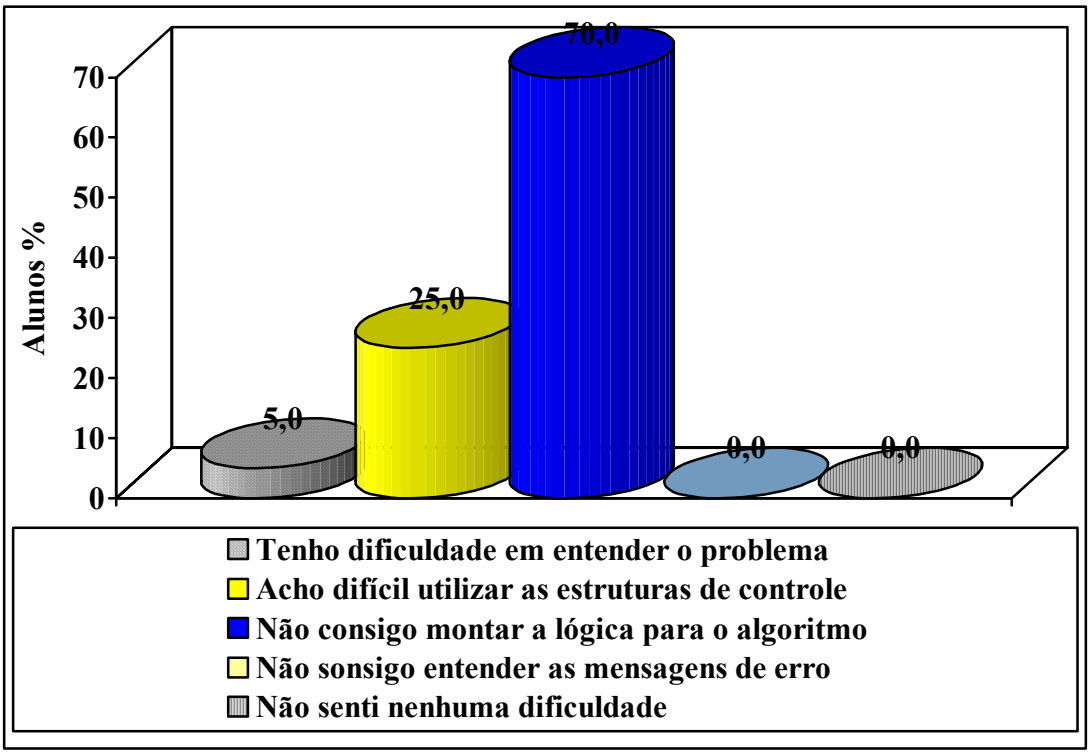

FIGURA 6.2 - Respostas dos alunos da turma do PCP em relação às dificuldades

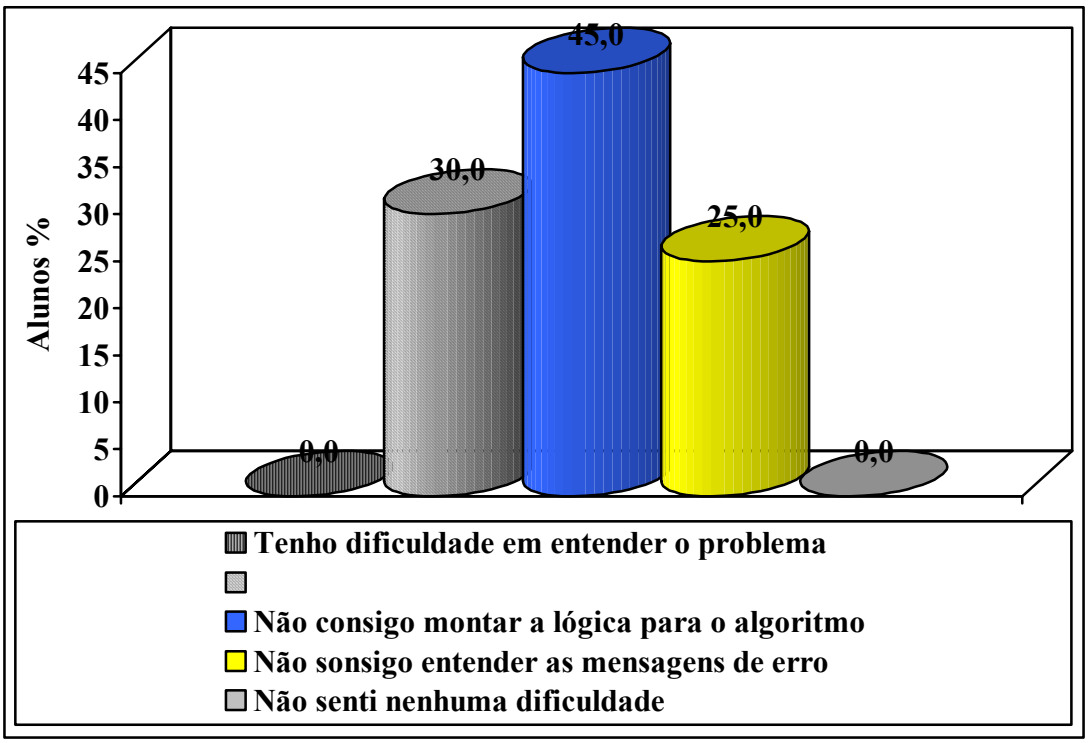

FIGURA 6.3 - Respostas dos alunos do Pascal em relação às dificuldades

Com estes resultados podemos notar que os alunos da primeira turma tiveram seus problemas mais voltados para o raciocínio lógico, enquanto que os alunos da segunda

turma tiveram muitos problemas com as mensagens do compilador e as estruturas de controle, uma vez que, segundo eles, se confundem sobre o objetivo de uma e outra quando vão escrever o programa.

6.6 Sexto encontro

No sexto encontro, ocorrido em 10 de julho de 2000, foram passados três exercícios com o objetivo de praticar as estruturas de controle já vistas até então. Como estes exercícios tinham uma lógica muito parecida com os da aula anterior, as dificuldades de entendimento do problema não existiram nas duas turmas. Na segunda turma foi necessário fazer uma revisão rápida das estruturas básicas de controle, pois alguns alunos estavam confundindo a estrutura "Enquanto...faça" com a estrutura "Para...até...faça".

6.7 Encontros finais

Do sétimo ao nono dia foram passados outros exercícios como forma de melhorar a lógica de programação. No décimo e último dia de aula nos reunimos para tirar as últimas dúvidas e discutirmos sobre o aprendizado adquirido nas duas semanas de aula que tivemos. Este debate aconteceu com cada uma das turmas em separado, para que os alunos não dessem respostas influenciadas pelas dos outros colegas da outra turma, numa intenção de se mostrarem melhores ou de ficarem fazendo comparações por conta própria.

6.8 Conclusões da segunda etapa

A turma dos alunos que utilizou a ferramenta PCP teve maior agilidade durante a correção dos erros nos algoritmos, tanto erros nas estruturas de controle como de sintaxe de expressões.

Em conseqüência da programação em Português, a turma do PCP demonstrou menos dificuldades em relação a encontrar as soluções dos algoritmos. Dois alunos chegaram a resolver alguns exercícios duas vezes, usando estruturas diferentes.

A turma dos alunos que utilizou o Pascal tinha maior facilidade no momento de compilar o programa do que os alunos da primeira turma, isto se deve ao fato de que o Pascal possui um ambiente integrado que facilita o trabalho de edição e compilação, o que não aconteceu com a primeira turma, que tinha que editar o programa no Edit do Ms-Dos, sair do editor para submeter o programa aos analisadores do PCP para depois compilar o programa chamando o compilador do Pascal. Com este processo eles perdiam muito tempo.

Fazendo-se uma análise do aprendizado obtido nas últimas aulas entre as duas turmas, os alunos que usaram o PCP tiveram muito mais segurança para começar a escrever os programas propostos nos exercícios e conseguiam terminar com mais rapidez, uma vez que a única preocupação deles era a de resolver a lógica do problema proposto, sem a

necessidade de ficar traduzindo cada mensagem de erro que aparecia ou até mesmo os comandos e estruturas de controle.

Para se ter uma idéia das dúvidas que os alunos da segunda turma tiveram em relação à tradução dos comandos, muitas vezes perguntavam ao professor se o comando para escrever na tela era o "Read" ou o "Write", o que não aconteceu com a primeira turma, pois os comandos semelhantes em PCP são o "Leia" e o "Imprima", que obviamente dispensam esta pergunta.

Os alunos da primeira turma chegaram a colocar a seguinte observação: "programar o computador nada mais é do que dizer para ele o que fazer, usando alguns comandinhos simples". Já os alunos da segunda turma terminaram as aulas com um pouco mais de receio no que se refere à facilidade de programar e apontaram que o mais difícil era ficar lembrando para que servia cada comando.

Para finalizar a conversa com os alunos, em ambas as turmas foi feita a mesma pergunta, a saber:

- Em relação ao nível de dificuldade encontrado para fazer um programa, você pode afirmar que programar é:

- Fácil
- Dificuldade média
- Difícil

As respostas das duas turmas são apresentadas nos gráficos a seguir:

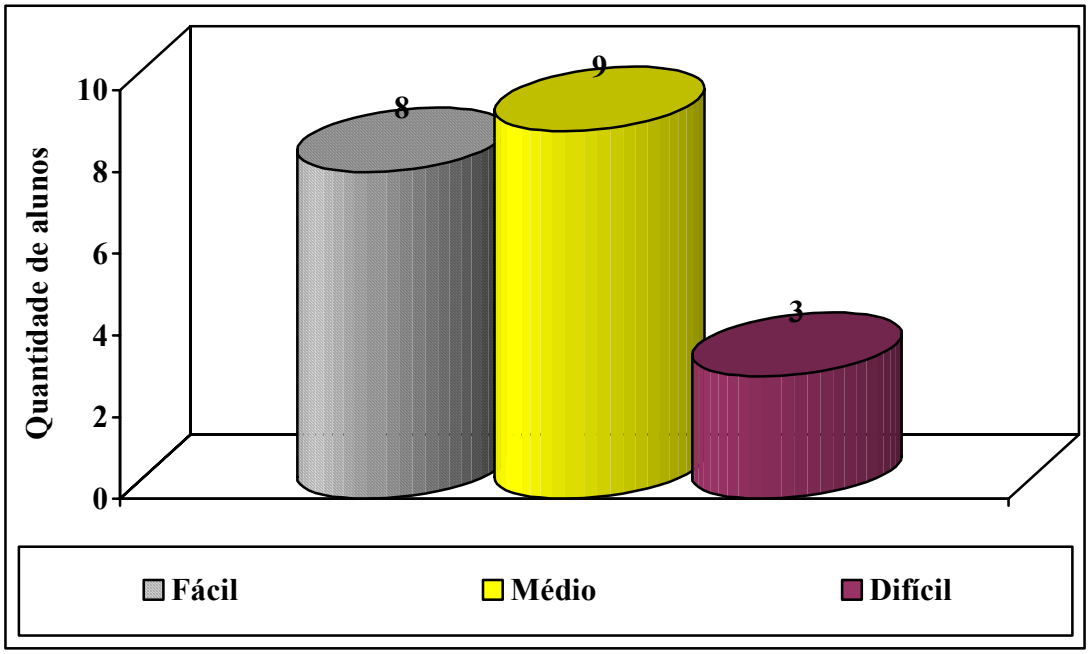

FIGURA 6.4 - Opinião da primeira turma em relação à dificuldade de programar

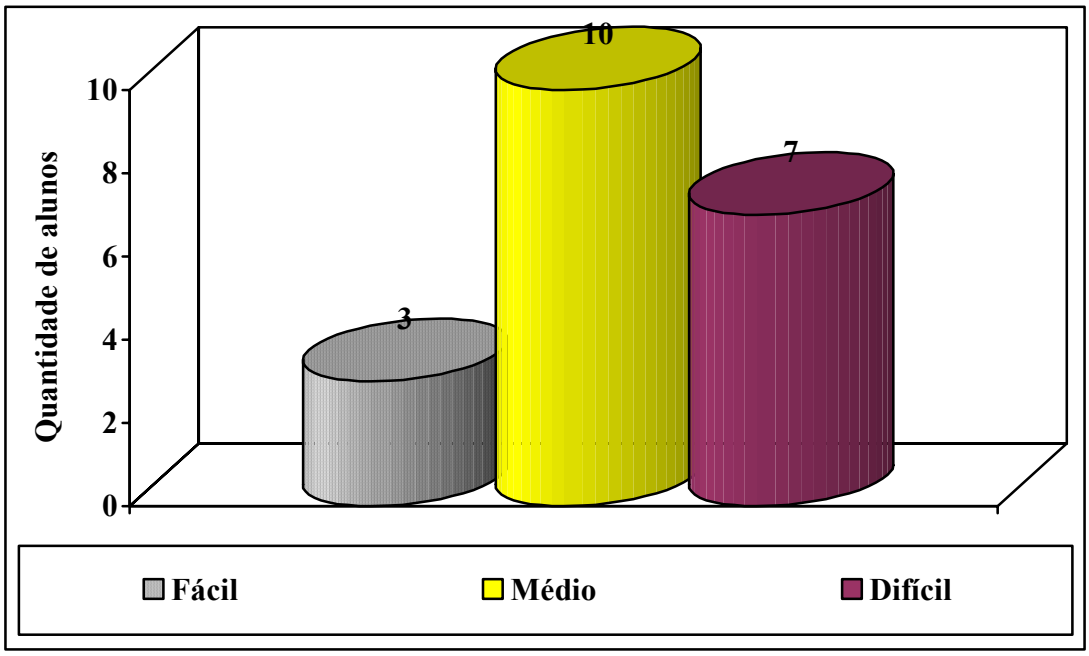

FIGURA 6.5 - Opinião da segunda turma em relação à dificuldade de programar

Apesar de alguns alunos terem apresentado dificuldade na manipulação dos comandos do editor de textos e do sistema operacional, a utilização do PCP fez que uma das turmas tivesse um melhor aproveitamento nas aulas, pois a quantidade de dúvidas apresentadas por eles durante as aulas eram bem menores e raramente incluíam perguntas sobre mensagens do compilador.

Em relação ao aproveitamento durante as aulas, ficou claro que o uso do PCP serviu de estímulo para os alunos tentarem resolver o mesmo exercício mais de uma vez, usando estruturas diferentes.

7 Resolução de exercícios em PCP

Apresentaremos a seguir a resolução de dois dos algoritmos utilizados nas experiências, na sintaxe do PCP.

Algoritmo 1: Faça um programa para ler as médias finais de 30 alunos de uma turma e informar quantos foram aprovados e quantos foram reprovados, sabendo-se que a média para aprovação é 7,0.

```
Programa MEDIAS;
Variavel
  TOTAL_ALUNOS, TOTAL_APROVADOS, TOTAL_REPROVADOS: inteiro;
  MEDIA_FINAL: real;
Inicio
  TOTAL_APROVADOS := 0;
  TOTAL_REPROVADOS := 0;
  Imprima_ln('Cálculo da média dos alunos');
  Imprima_ln('Informe a média para 30 alunos da turma');
  Para TOTAL_ALUNOS de 1 ate 30 faca
        Imprima('Informe a média do aluno ',TOTAL_ALUNOS,':');
        Leia_ln(MEDIA_FINAL);
        Se MEDIA_FINAL >= 7 entao
            TOTAL_APROVADOS := TOTAL_APROVADOS + 1
        Senao
             TOTAL_REPROVADOS := TOTAL_REPROVADOS +1;
        Fim-se;
  Fim-para;
  Imprima_ln('Total de aprovados: ',TOTAL_APROVADOS);
  Imprima_ln('Total de reprovados: ',TOTAL_REPROVADOS);
Fim.
```

Algoritmo 2: Faça um programa para ler o número de matrícula, sexo e as médias finais de N alunos de uma turma e imprimir quantos foram aprovados, quantos foram reprovados, o percentual de mulheres e o percentual de homens na turma. Sabe-se que a média para aprovação é 7,0 e que as médias devem parar de ser informadas quando for lido um número de matrícula igual a zero.

```
Programa EXEMPLO_ENQUANTO_FACA;
Variavel
  MATRICULA, TOTAL_ALUNOS, TOTAL_HOMENS, TOTAL_MULHERES,
  TOTAL_APROVADOS, TOTAL_REPROVADOS: inteiro;
  MEDIA_FINAL, PERC_HOMENS, PERC_MULHERES: real;
  SEXO: caracter;
Inicio
  TOTAL_APROVADOS := 0;
  TOTAL_REPROVADOS := 0;
  TOTAL_HOMENS := 0;
  TOTAL_MULHERES := 0;
  Imprima_ln('Cálculo da média dos alunos');
  Imprima_ln('Informe o número de matrícula');
  Leia_ln(MATRICULA);
  Enquanto MATRICULA > 0 faca
        Imprima('Informe o sexo do aluno (M/F):');
        Leia_ln(SEXO);
        Imprima('Informe a média do aluno ',TOTAL_ALUNOS,':');
        Leia_ln(MEDIA_FINAL);
        Se MEDIA_FINAL >= 7 entao
           TOTAL_APROVADOS := TOTAL_APROVADOS + 1
        Senao
            TOTAL_REPROVADOS := TOTAL_REPROVADOS +1;
        Fim-se;
        Se SEXO='F' entao
           TOTAL_HOMENS := TOTAL_HOMENS + 1;
        senao
           TOTAL_MULHERES := TOTAL_MULHERES + 1;
         Fim-se;
         Imprima_ln('Informe o número de matrícula do próximo aluno');
         Leia_ln(MATRICULA);
  Fim-enquanto;
  TOTAL_ALUNOS := TOTAL_APROVADOS + TOTAL_REPROVADOS;
  Se TOTAL_ALUNOS > 0
    Entao inicio
      Imprima_ln('Total de aprovados: ',TOTAL_APROVADOS);
```

```
      Imprima_ln('Total de reprovados: ',TOTAL_REPROVADOS);
      PERC_HOMENS := TOTAL_HOMENS * 100 / TOTAL_ALUNOS;
      PERC_MULHERES := TOTAL_MULHERES * 100 / TOTAL_ALUNOS;
      Imprima_ln('Percentual de homens na turma: ',PERC_HOMENS);
      Imprima_ln('Percentual de mulheres na turma: ',PERC_MULHERES);
    Fim;
  Senao
    Imprima_ln('Nenhum aluno foi informado');
  Fim-se;
Fim.
```

8 Conclusões e trabalhos futuros

Este trabalho apresentou o PCP - Pseudo-Compilador Portugol, uma ferramenta para auxílio no desenvolvimento de algoritmos, e sua aplicação como apoiador no aprendizado de programação. Com esta pesquisa pudemos comparar os resultados obtidos entre turmas que usaram ou não esta ferramenta.

"Um dos critérios mais importantes para julgar uma linguagem de programação é a facilidade com que os programas podem ser lidos e entendidos" [SEB 99]. Com os resultados obtidos através dos vários exercícios aplicados, ficou claro que a utilização de uma ferramenta para auxiliar no ensino de programação deixa de ser uma expectativa de futuro para ser uma necessidade do presente, haja vista que atualmente diversos programas de computador são usados com o intuito de facilitar a aprendizagem nas mais diversas áreas existentes no mercado de trabalho.

Em ambas as etapas de pesquisa realizadas, foi possível notar que as dúvidas e problemas apresentados pelos alunos das turmas que usaram o PCP eram diferentes das dúvidas das turmas do Pascal. Na primeira, as perguntas eram mais voltadas para a lógica do problema. Na segunda, eram mais voltadas para erros de escrita de programa, o que conseqüentemente atrasava o desenvolvimento da lógica.

Na primeira etapa da pesquisa, realizada com alunos do curso de Ciência da Computação, a turma que trabalhou com o PCP melhorou a concentração durante a resolução dos exercícios propostos em laboratório.

Na segunda etapa da pesquisa, realizada com alunos do segundo grau que nunca tinham programado, percebemos que o aprendizado foi melhor entre os alunos que usaram o PCP.

Em ambas as etapas, foi fácil perceber que os alunos que programavam em Português terminavam os trabalhos primeiro. Esta diferença de tempo entre as turmas foi aumentando à medida que os exercícios ficavam mais difíceis. Havia uma motivação maior por parte dos alunos que programaram em Português, a cada novo exercício, a reação destes alunos era a de ter recebido um desafio. Podia-se perceber inclusive uma certa disputa entre alguns grupos de alunos para saber quem conseguia resolver primeiro. Os alunos que programaram em Pascal recebiam os novos exercícios como uma tarefa a mais a ser feita, alguns chegavam a se mostrar desinteressados.

No estudo feito com a ferramenta PCP, pudemos perceber que os alunos não apresentaram nenhuma dificuldade em manipulá-la, mas algumas reclamações recebidas devem ser anotadas para que melhorias sejam feitas nela em versões futuras, afim de atender cada vez mais a necessidade dos alunos e proporcionar maiores avanços na qualidade do aprendizado de programação. Entre as reclamações apontadas, podemos destacar:

- A utilização de ambientes separados para editar e testar o programa gerou um certo desconforto aos usuários. Depois de alguns exercícios resolvidos, eles não mais reclamavam desta dificuldade, mas era possível perceber que isto quebrava o ritmo de trabalho;

- Os alunos de computação com maior facilidade de programação citaram a limitação do PCP em relação aos comandos disponíveis na linguagem Pascal, principalmente de manipulação de tela;

Segundo os professores de computação, que estiveram acompanhando atentamente a pesquisa, a aula produzirá mais utilizando a ferramenta PCP, uma vez que as explicações dadas no quadro podem ser feitas com algoritmos escritos em Português e os alunos imediatamente fazem os exercícios nos computadores, sem precisar fazer analogia dos comandos apresentados pelo professor com os da linguagem Pascal.

Os alunos de computação que usaram o PCP, principalmente os do primeiro ano, passaram a não demonstrar mais o "medo" que é comum entre estudantes de programação, quando não se sentem seguros sobre a solução do problema ora proposto. Eles mesmos relataram que o fato de ter que associar cada comando do Pascal com o seu propósito já é um fator que atrapalha o raciocínio lógico, pois se ainda não tiverem decorado a tradução de todos eles, a preocupação inicial fica toda voltada para o problema de se usar o comando errado, pensando estar usando o certo.

A linguagem definida para a ferramenta PCP pareceu tão simples para os alunos que eles programavam de uma forma muito natural e tranqüila, que é o que se espera de uma linguagem de programação. Apesar de alguns alunos terem citado que a falta de determinados recursos da linguagem do PCP fez com que a programação ficasse um pouco limitada, outros disseram que aprenderam facilmente justamente porque é uma linguagem que possui os componentes básicos necessários, sem recursos complicados.

O fato de a linguagem do PCP suportar uma maneira natural de escrever o algoritmo, torna o programa confiável e legível, sendo que com estes dois atributos fica fácil para os alunos ou programadores se certificarem se foram escritos corretamente.

A implantação da ferramenta PCP, como parte integrante das aulas de programação, não implica em nenhum custo para os professores, em termos de complexidade na manipulação, treinamento e adaptação dos exercícios.

A manutenção dos programas pelos alunos também mostrou-se muito mais simples com o PCP do que com o Pascal; é claro que os programas podem possuir linhas de comentários, o que aliás é muito importante, mas não é comum os alunos colocarem comentários em todas as linhas do programa. Então, por mais comentado que seja o programa, ele nunca será mais fácil de entender do que um que tenha sido escrito em uma linguagem natural, como é o caso do PCP.

Finalmente, a linguagem de programação da ferramenta PCP possibilita aos estudantes, e por que não dizer também aos professores, maior domínio e clareza no desenvolvimento e manutenção dos programas, assim como melhora a confiabilidade nas estruturas empregadas e nos resultados obtidos.

Como trabalhos futuros, podemos citar as seguintes propostas:

- A construção de um ambiente integrado para a ferramenta PCP, incluindo um editor de programas, analisando inclusive opiniões dos alunos para que este ambiente se molde às necessidades e anseios deles;
- Oferecer esta ferramenta aos professores de disciplinas de programação, não só nos cursos de Ciência da Computação, mas em todos os cursos que tiverem esta disciplina em sua grade;

- Incrementar o PCP com outros comandos disponíveis na linguagem Pascal, para que todas as necessidades de desenvolvimento de algoritmos sejam atendidas;
- Montar grupos de estudos com alunos de Computação para que eles possam trabalhar em cima de melhorias na ferramenta PCP, com possibilidade até de atuarem como auxiliares nas aulas de programação que utilizarem o PCP;
- A divulgação deste trabalho para outras universidades.

Anexo 1 Formalismo Backus-Naur (BNF)

Segundo [SEB 99] “a BNF é uma notação muito natural para descrever a sintaxe de uma linguagem”. A seguir é apresentada a sintaxe do PCP.

<area-comando> ::= <comando-composto>

<area-declaracao> ::= {<secao-declaracao>}

<area-declaracao-variaveis> ::= variável <declaracao-variavel>;

<bloco> ::= <area-declaracao> <area-comando>

<cabecalho-programa> ::= <vazio> ! programa <identificador- programa>;

<cabecalho-procedimento> ::= procedimento <identificador> ;

<cadeia> ::= {<elemento-string>}

<caractere> ::= <letra-ou-digito> ! <caractere-especial>

<comando> ::= <comando-simples> ! <comando-estruturado>

<comando-atribuicao> ::= <variavel> := <expressão>

<comando-composto> ::= inicio <comando> {;<comando>} fim

<comando-condicional> ::= <comando-se>

<comando-enquanto> ::= enquanto <expressão> faca <comando> fim_enquanto

<comando-estruturado> ::= <comando-composto> ! <comando-condicional> ! comando-repeticao>

<comando-para> ::= para <variavel-controle> de <lista-para> faca <comando> fim_para

<comando-procedimento> ::= <identificador-procedimento> ! <identificador-procedimento> (<parametro-atual> {,<parametro-atual>})

<comando-repita> ::= repita <comando> {;<comando>} ate_que <expressão>

<comando-repeticao> ::= <comando-enquanto> ! <comando-repita> ! <comando-para>

<comando-se> ::= se <expressão> então <comando> {senao <comando>} fim_se

<comando-simples> ::= <comando-atribuicao> ! <comando-procedimento> ! <comando-vazio>

<comando-vazio> ::= <vazio>;

<constante> ::= <numero-sem-sinal> ! <sinal> <numero-sem-sinal> ! <identificador-constante> ! <sinal> <identificados-constante> ! <cadeia>

<constante-sem-sinal> ::= <numero-sem-sinal> ! <cadeia> ! <identificador-constante> !

<declaracao-variavel> ::= <lista-identificadores> : <tipo>;

<declaracao-constante> ::= <definicao-constante-sem-tipo>

<declaracao-procedimento> ::= <cabecalho-procedimento> <bloco>

<definicao-constante-sem-tipo> ::= <identificador> = <constante>

<digito> ::= 0! 1! 2! 3! 4! 5! 6! 7! 8! 9

<elemento-string> ::= <texto-string>

<expressao-simples> ::= <termo> {<operador-adicao> <termo>}

<expressão> ::= <expressao-simples> {<operador-comparacao> <expressao-simples>}

<grupo-parametros> ::= <lista-identificadores> : <identificador-tipo>;

<identificador-constante> ::= <identificador>

<identificador-programa> ::= <identificador>

<identificador-procedimento> ::= <identificador>

<identificador-tipo> ::= <identificador>

<identificador-variavel> ::= <identificador>

<identificador> ::= <letra> {<letra-ou-digito>}

<inteiro-sem-sinal> ::= <sequencia-digitos>

<letra-ou-digito> ::= <letra> ! <digito>

<letra> ::= A! B! C! D! E! F! G! H! I! I! J! K! L! M! N! O! P! Q! R! S! T! U! V! W! X! Y! Z! a! b! c! d! e! f! g! h! i! j! k! l! m! n! o! p! q! r! s! t! u! v! w! x! y! z! -

<lista-para> ::= <valor-inicial> ate <valor-final>

<lista-identificadores> ::= <identificador> {<identificador>}

<operador-aritmeticos> ::= + ! -

<operador-comparacao> ::= < > ! = ! <= ! >= ! < ! >

<parametro-atual> ::= <expressão> ! <ariavel>

<procedimento> ::= <cabecalho-procedimento> <bloco>

<programa> ::= <cabecalho-programa> <bloco>

<sequencia-digitos> ::= <digito> {<digito>}

<secao-declaracao> ::= <area-declaracao-variaveis>

<sinal> ::= + ! -

<texto-string> ::= '{<caractere>}'

<valor-final> ::= <expressão>

<valor-inicial> ::= <expressão>

<variavel> ::= <identificador-variavel> ! <variavel-componente>

<vazio> ::=

Anexo 2 Blocos importantes do programa

Apresentaremos alguns trechos importantes que compõem o Pseudo-Compilador Portugol.

O trecho a seguir separa os símbolos (função **ps()**)encontrados no código-fonte e armazena-os na variável global str[] juntamente com o seu tipo:

```
ps()
{
   register char *temp;
   str[0]='\0';
   temp = str;
   tipo=0;

  if(*prog=='}') { prog++; erro(10); }/* fecha sem abrir comentario */
  if(*prog=='{')                      /* testa se e' comentario     */
  {
     prog++;
     while(*prog!='}')
     {
      prog++;
      if(*prog==0x0a)  /* verifica se e' fim de linha */
      {
         erro(10);  linhas++;
         break;
      }
     }
     prog++;
  }

  if(*prog=='\x1a') {
    linhas++;
    str[0]='\0';
    return(tipo=FIM_ARQ);            /* testa fim de programa */
  }
  if(*prog==0x0a) linhas++;          /* testa fim de linha    */
```

```
  while(iswhite(*prog)) prog++;      /* salta branco e tab    */

  if(isdelim(*prog))                 /* VERIFICA SE E' DELIMITADOR */
  {
    if(isdelim_esp(*prog))   return(tipo=DELIM_ESPEC);
                                     /* SE FOR FIM DELIM. ESPECIAL */
    *temp=*prog;
    prog++;
    temp++;
    *temp='\0';
    return(tipo=DELIMITADOR);
  }
  if(isnum(*prog)) { /* numero */
    while(!isdelim(*prog)) *temp++=*prog++;
    *temp = '\0';
    return(tipo = NUMERO);
  }

  if(isalfx(*prog)) { /* variavel ou comando */
   str[0]=0;
   while(!isdelim(*prog)) *temp++ = *prog++;
   *temp = '\0';
   return(tipo=STRING);
  }
}
```

O próximo trecho é uma parte componente do analisador de expressão, que testa se as expressões foram escritas corretamente.

```
if(ntipos==999) { ntipos=1;
      /* QUANTIDADE DE TIPOS DE UMA INSTANCIACAO */
      ltipos[1]='\0';
                }
    else    {  ntipos=0;
            /* QUANTIDADE DE TIPOS DE UMA COMPARACAO   */
            ltipos[0]='\0';
            }
    no=verifica_no(str);
    if(strchr("89ABCDEF",no)) erro(1);
            /* ERRO GERAL DE EXPRESSAO  */
```

```
for(;;)
{
   switch(no)
   {
   case '1' : if(!(prox_esta_em("4")))          erro(9);  break;
   case '2' : if(!(ver_marca_incval(str))) erro(22);
              marca_uso(str);
              ltipos[ntipos]=ver_tipo(str); ntipos++;
              if(!(prox_esta_em("56789ABDEFC"))) erro(43); break;
   case '3' : if(!(prox_esta_em("56789ADEF")))   erro(43); break;
   case '4' : if(!(prox_esta_em("123467")))erro(43); par++;break;
   case '5' : if(!(prox_esta_em("56789ACDEF")))  erro(43);
              par--;                                        break;
   case '6' : if(!(prox_esta_em("1234")))        erro(43); break;
   case '7' : if(!(prox_esta_em("1234")))        erro(43); break;
   case '8' : if(!(prox_esta_em("1234")))        erro(43); break;
   case '9' : if(!(prox_esta_em("1234")))        erro(43); break;
   case 'A' : if(!(prox_esta_em("1234")))        erro(43); break;
   case 'B' : if(!(prox_esta_em("C")))           erro(43); break;
   case 'C' : if(!(prox_esta_em("123467H"))      erro(43);
              igu++;                                        break;

   case 'D' : pss();
              maiusc(str);
              if(strcmp("SENAO",str)==0) { erro(55); }
              /* {c_senao();} */
              volta_ps();
              caifora();

   case 'E' : if(!(prox_esta_em("123467C")))     erro(43); ma++;
              break;
   case 'F' : if(!(prox_esta_em("123467CE")))    erro(43); me++;
              break;
   case 'N' : if(!(prox_esta_em("123456789ABCDEFNH"))) erro(1);
              break;
   case 'H' : do
              { ps();
              if(*prog==0x0a) { erro(1); goto caifora; }
              } while(str[0]!='\x27');
```

```
                pss();
                no=verifica_no(str);
                break;
       default :  caifora();
       }
    }
```

Rotina principal do analisador sintático, onde um símbolo é obtido e faz-se a verificação se é uma variável, um comando, função ou procedimento.

```
switch(verifica_simbolo())
  {
      case VARIAVEIS   : expressao();       break;
      case CHAM_PROCED : chama_proced();    break;
           /* checa argumentos    */
      case DECL_PROCED : declara_proced();  break;
           /* guarda identf e arg */

      /* palavras reservadas */
      case SE          : c_se(); tcomandos=0;           break;
      case ENTAO       : tcomandos=0;                   break;
      case SENAO       : if(tcomandos>1) verifica_fim();
                         tcomandos=0;
                         /*ce_senao();*/                break;
      case ENQUANTO    : c_enquanto();                  break;
      case FIM_ENQUANTO: c_fim_enquanto();              break;
      case PARA        : c_laco_para();                 break;
      case FIM_PARA    : c_fim_para();                  break;
      case REPITA      : c_repita();                    break;
      case ATE_QUE     : c_ate_que();                   break;
      case FIM         : c_fim(); tcomandos=0;          break;
      case FIMP        : c_fimp();                      break;
      case FIM_SE      : c_fim_se();
                         if(tcomandos>1) verifica_fim_se();
                         tcomandos=0;                   break;
      case INICIO      : c_inicio();                    break;
      case PROGRAMA    : c_programa();                  break;
      case LEIA        : tcomandos++; c_leia();         break;
      case LEIA_LN     : tcomandos++; c_leia();         break;
      case IMPRIMA     : tcomandos++; c_imprima();      break;
```

```
      case IMPRIMA_LN  : tcomandos++; c_imprima();       break;
      case VARIAVEL    : if(inicio1) erro(44);
                         if(var1)    erro(45);
                         else  var1=1;  c_var();         break;
      case FIM_PROC    : c_fim_proc();                   break;
      default          : erro(8);
   }
```

Parte do módulo conversor, que transforma o código em PCP num código em Pascal.

```
void conv_prog()
{
   int fimse=0;
   prog=inicio_prog;
   for(;;)
   {
     pega_s();
     if(tipo==DELIMITADOR)
     {
         if(fimse==0) {  grava_ch(*prog); prog++; }
         else {  if(*prog!=';') { grava_ch(*prog); prog++; }
                else { prog++; fimse=0; }
            }
     }
     else
     {
       if(verifica_simbolo()==FIM_SE)
       {
          fimse=1;
       }
       else
       {
         troca();  grava_str(tpas);
           if(strcmp("Do",tpas)==0)
           {
            grava_ch('\x0d');
            grava_ch('\x0a');
            grava_str("   Begin");
           }
```

```
        }
       /*   if(isdelim(*prog)) { grava_ch(*prog); } */
      }
      if(*prog==0x1a) break;
   }
}
```

Anexo 3 Gramática do PCP

Segundo [BEC 97], "A gramática de uma linguagem de programação é uma descrição formal da sintaxe, ou forma, dos programas e instruções individuais escritas nessa linguagem".

A3.1 Elementos básicos

O PCP é constituído de letras maiúsculas e minúsculas ("A" - "Z", "a" - "z"), caracter sublinhado ("_"), os dígitos de 0 a 9 e os símbolos especiais +-*/=<>(){}.,;:'. Seus operadores e delimitadores são os seguintes:

Adição	+
Negação, subtração	-
Multiplicação	*
Divisão	/
Igualdade, declaração	=
Menor que	<
Maior que	>
Maior ou igual a	>=
Menor ou igual a	<=
Desigualdade (diferente de)	<>
Precedência	()
Delimitadores de comentários	{}
Fim do programa, decimal	.
Separador de lista	,
Declaração de tipo	:
Separador de comando	;
Delimitador literal de seqüência de caracteres	'
Atribuição	:=

A3.2 Declaração de variáveis.

As variáveis devem ser declaradas antes do delimitador de início de bloco “inicio”. A seção de declaração de variáveis deve ser iniciada pela palavra reservada “variavel”.

Exemplo:

```
programa DECLARACAO_DE_VARIAVEIS;
variavel {seção de declaração de variáveis}
    VARIAVEL_1, VARIAVEL_2 : inteiro;
    VARIAVEL_3, VARIAVEL_4 : real;
inicio {Delimitador de início de bloco}
    leia_ln(VARIAVEL_1, VARIAVEL_2);
    leia_ln(VARIAVEL_3, VARIAVEL_4);
fim.
```

A3.3 Palavras reservadas

As palavras reservadas são nomes utilizados pelo PCP que tem um sentido predeterminado no mesmo, portanto não podem ser redefinidas pelo usuário como identificadores ou utilizados de outra forma senão para a que foram criadas. São elas:

se	variável	programa
então	inicio	para
senao	fim	imprima
enquanto	fim_se	procedimento
faca	fim_enquanto	ate_que
repita	fim_para	de
ate	leia	fim_proc
inicio_proc	leia_ln	imprima_ln

A3.4 Identificadores

Os Identificadores são nomes simbólicos para os objetos referenciados nos programas escritos em PCP. Esses nomes são escolhidos pelo usuário para representar endereços de memória onde vão ser alocadas as informações. O Identificador é um elemento básico do PCP. Sua sintaxe é definida na figura 2.3.

O nome do identificador deve:

- ter como primeiro caractere uma letra ou "_" (sublinhado);
- ter como demais caracteres letras, dígitos, ou "_";
- não ser palavra reservada.

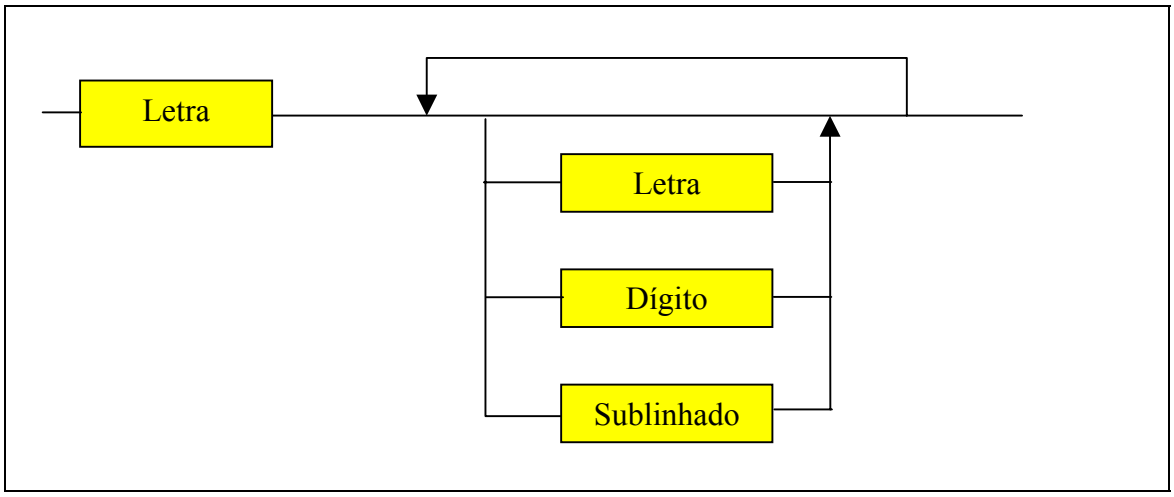

FIGURA A3.1 - Diagrama da definição de um identificador.

O tamanho máximo de um identificador e de 32 (trinta e dois) caracteres. Não há distinção entre letras maiúsculas e minúsculas, por isso, vale dizer que:

NOME_DO_FUNCIONARIO

é igual a

nome_do_funcionario

Exemplos de identificadores válidos:

A, A123, B52f, LETRA, Nome_do_Cliente.

Exemplos de identificadores inválidos:

PROGRAMA, 123_4, -ABC.

A3.5 Tipos básicos de variáveis

Uma variável é um identificador e deve ser declarada de acordo com o diagrama apresentado na figura abaixo:

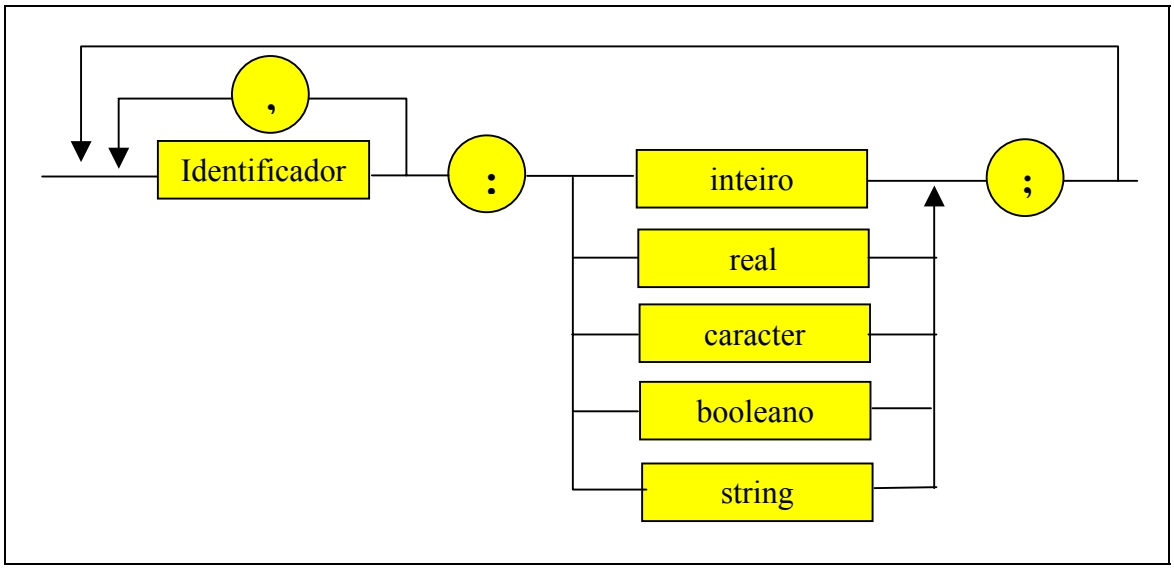

FIGURA A3.2 - Diagrama da definição de uma variável.

Os tipos que os identificadores podem assumir são:

INTEIRO : Todo número inteiro, negativo, nulo ou positivo. As variáveis inteiras podem conter valores contidos no intervalo de 32767 a -32768 e ocupam dois bytes de memória.

Exemplos: 10, -23, 0.

REAL: Todo numero real, negativo, nulo ou positivo, incluindo frações decimais. As variáveis do tipo real devem ter seus valores contidos no intervalo de 1.0E-38 a 1.0E+38, com uma mantissa de 11 dígitos significativos. Este tipo de variável ocupa seis bytes de memória.

Exemplos: 43.5, -10, 0, 100.

CARACTER : Todo conjunto de caracteres alfanuméricos contidos no conjunto de caracteres ASCII, variando de 0 a 255. As variáveis deste tipo de dado devem ser declaradas entre apóstrofos (‘ ‘) e ocupam um byte de memória.

Exemplos: "SOMA", "X", "XYZ".

BOOLEANO : Conjunto de valores FALSO ou VERDADEIRO em proposições lógicas.

STRING: É um conjunto de caracteres que pode conter letras, números ou caracteres especiais. Sua representação deve aparecer entre apóstrofos (‘ ‘). Na sua definição, deve-se informar entre colchetes ([ ]) um valor representando o tamanho da string que se pretende criar.

O número máximo de variáveis que o programador poderá definir em seu programa será de duzentos. A definição do tipo das variáveis deve ser feita conforme mostra o exemplo que segue:

```
programa DEFINICAO_TIPOS;
variavel
    CONTADOR: inteiro;
    X,Y: real;
```

```
        LETRA: caracter;
        RESPOSTA: booleano;
        NOME: string[30];
inicio
    { Comandos do programa principal }
fim.
```

A3.6 Funções numéricas pré-definidas

Estas funções predefinidas estão à disposição dos usuários para o cálculo das funções matemáticas mais comuns. São elas:

SIN (X)	Obtém um número real, que representa o seno de X, onde X é um arco em radianos.
COS (X)	Produz valor real, que indica o co-seno de X, onde X é um arco em radianos.
ARCTAN (X)	Gera um número real, que representa o total de radianos do arco cuja tangente é X.
FRAC (X)	Determina o número real formado pelos dígitos de X situados após o ponto decimal. Ex.: FRAC(2.250) = 0.250
ABS (X)	Estabelece um valor inteiro ou real que contém o valor absoluto ou modulo de X.
SQR (X)	Compõe o valor de X elevado ao quadrado, logrando obter um resultado inteiro ou real, de acordo com as características de X (inteiro ou real, respectivamente).
SQRT (X)	Gera um número real que representa a raiz quadrada de X.
EXP (X)	Obtém um valor real que expressa o resultado de e^x, onde e = 2.71828.
LN (X)	Encontra um número real que corresponde ao logaritmo neperiano de X.
INT (X)	Retorna a parte inteira de X (isto é, o maior número inteiro menor ou igual a X, se X >= 0, ou menor número inteiro maior ou igual a X, se X < 0). Esta função aceita argumentos reais ou inteiros e retorna um resultado real.

Exemplo de programa com funções numéricas:

```
programa FUNCOES_NUMERICAS;
variavel
  SENO, CO_SENO, QUADRADO, RAIZ: real;
  X: real;
inicio
  imprima(' Informe o valor de X -> ');
  leia_ln(X);
  SENO := sin(X);
  CO_SENO := cos(X);
  QUADRADO := sqr(X);
  RAIZ := sqrt(X);
  imprima_ln('O Seno de ',X,' eh igual a -> ',SENO);
  imprima_ln('O Co-seno de ',X,' eh igual a -> ',CO_SENO);
  imprima_ln('O Quadrado de ',X,' eh igual a ->',QUADRADO);
  imprima_ln('A Raiz de ',X,' eh igual a -> ',RAIZ);
fim. {Fim do programa}
```

A3.7 Estrutura do programa

Um programa escrito em PCP é uma seqüência de instruções que implementam o algoritmo ou método de resolução previamente escrito em Portugol.

O PCP utiliza uma forma estruturada de programar, onde cada seção ou parte do programa escrito nesta linguagem deve estar sistematicamente correta.

A3.7.1 Cabeçalho do programa

Os programas escritos em PCP devem obrigatoriamente iniciar-se com um cabeçalho, definido como no exemplo a seguir:

```
programa <nome-do-programa>;
```

A3.7.2 Bloco de programa

Um bloco é uma seqüência de comandos e definições finitas, delimitado pelas palavras reservadas **início** e **fim**. A estruturação da linguagem em blocos permite o agrupamento de declarações e comandos. Um bloco pode conter declarações, comandos do programa, blocos aninhados e outros; são úteis para a organização do programa (estruturação).

A3.7.3 Corpo do programa

O corpo do programa, ou bloco de programa, é constituído de declarações e comandos. As declarações definem os objetos a serem utilizados por todo o programa (objetos globais). Os comandos especificam as ações a serem executadas pelo computador.

O fim do programa deverá conter um ponto seguindo o delimitador de fim de bloco **FIM**.

A seguir, temos um exemplo simplificado de um programa escrito em PCP.

programa Ola;
inicio
    imprima(‘Olá, Porto Alegre!’);
fim.

Este programa irá mostrar na tela a mensagem: *Olá, Porto Alegre!*

A3.8 Comandos básicos

Os comandos especificam as ações a serem realizadas pelo computador, como comparações e atribuições. Eles se constituem por expressões, palavras-chave e operadores.

a) Atribuição

Para atribuirmos um valor a uma variável usaremos o símbolo de atribuição :=. Sua sintaxe é apresentada na figura 2.5.

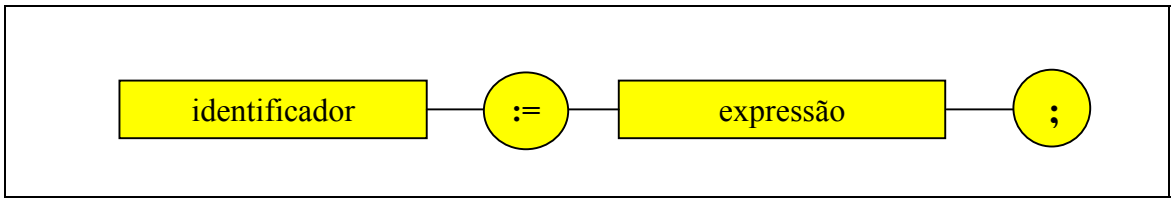

FIGURA A3.3 - Diagrama da atribuição de valores a identificadores.

Exemplo: VALOR := (TOTAL1+TOTAL2) * (A/B)

Neste exemplo, a variável VALOR receberá o resultado do cálculo da expressão da direita.

b) Operadores

Os operadores são fornecidos para possibilitar a formação de vários tipos de expressões.

Os operadores no PCP possuem uma seqüência na qual as expressões serão avaliadas. Se dois operadores numa mesma expressão possuírem o mesmo nível de precedência, a expressão será avaliada da esquerda para a direita. As expressões contidas entre parênteses serão resolvidas em primeiro lugar, a começar pelos parênteses mais internos.

A ordem de precedência é a seguinte:

1º - Expressões dentro de parênteses

2º - Operador unário menos ou negação

3º - Operadores multiplicativos: *, /

4º - Operadores aditivos: +, -

5º - Operadores relacionais: =, <>, <, >, <=, >=

Operadores Aritméticos:

São os símbolos das quatro operações básicas: +, -, *, /. Eles representam as operações de adição, subtração e negação, multiplicação e divisão respectivamente.

Operadores Relacionais:

Nas expressões que utilizam os operadores relacionais, o resultado da relação será sempre Verdadeiro ou Falso. São eles:

=	igual
<>	diferente
>	maior que
<	menor que
>=	maior ou igual a
<=	menor ou igual a

c) Expressões

As expressões são constituídas por constantes, variáveis e operadores, que definem o método para calcular um valor. O resultado das expressões pode ser armazenado em uma variável de um tipo compatível com o valor resultado.

São exemplos de expressões:

```
a > b
Resultado := 3 + (x * y)
```

D + num / alfa

DIA := 30

A3.9 Comandos condicionais

A3.9.1 Se .. entao

Esta é a estrutura básica de controle em quase todas as linguagem de programação. É empregado quando a ação a ser executada depende de uma inspeção ou teste. Este comando nos fornece a habilidade de fazer uma decisão simples, se uma dada condição for verdadeira.

Sua forma é: Se <condição> entao <comando>

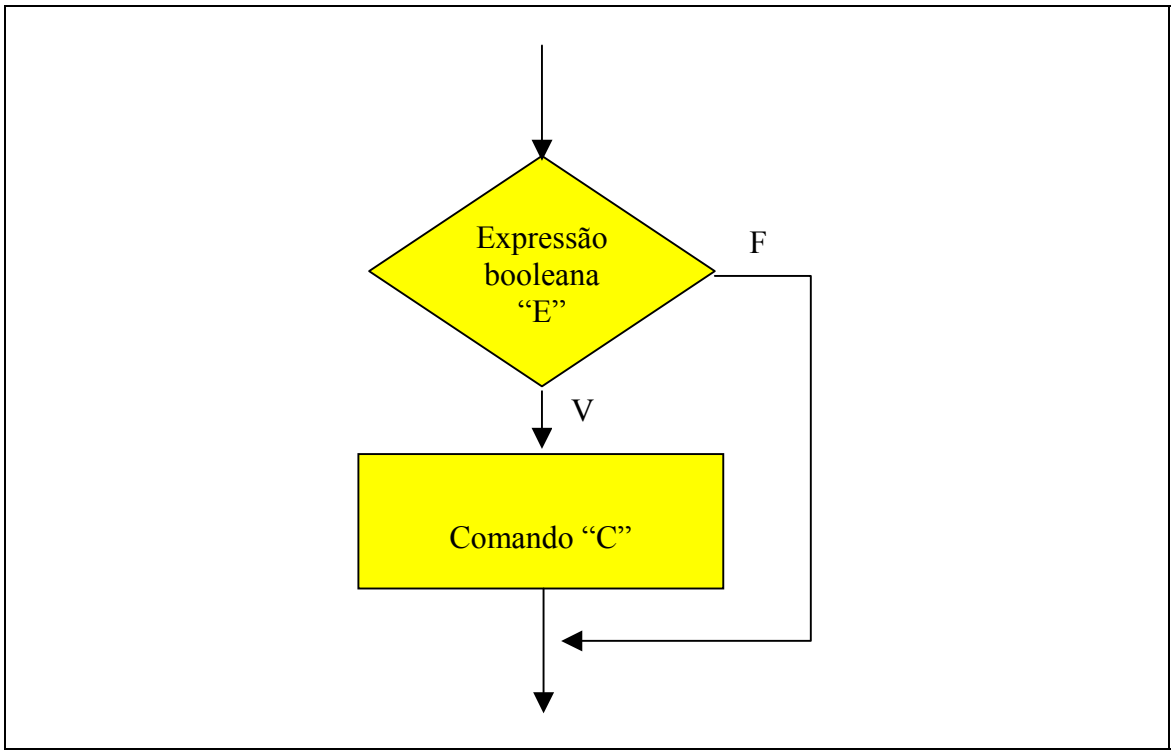

FIGURA A3.4 - Diagrama de fluxo da estrutura SE..ENTAO.

Exemplo:

```
programa Exemplo_Se_Entao;
variavel
        X,Y : inteiro;
inicio
        X := 1;
        Y := 2;
```

```
        se X > Y
           entao  Y := 10;
        fim_se;
fim.
```

A3.9.2 Se .. entao .. senao

Este comando é empregado quando a ação a ser executada depende de uma inspeção ou teste. Ele nos fornece a habilidade de executar um comando composto, se determinada condição for verdadeira ou falsa.

Sua forma é: Se <condição> entao <comando 1>
senao <comando 2>

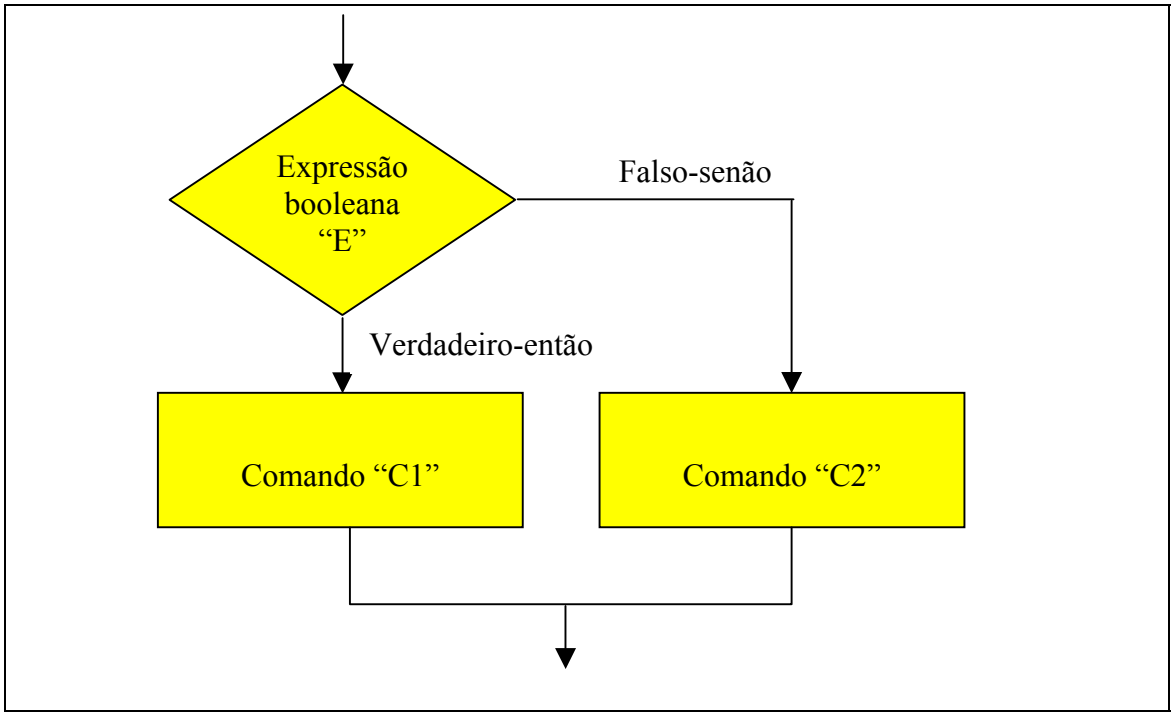

FIGURA A3.5 - Diagrama de fluxo da estrutura SE...ENTAO...SENAO.

Exemplo:

```
programa ILUSTRA_SE_ENTAO_SENAO;
variaveis
        X,Y : inteiro;
inicio
        X := 1;
        Y := 2;
```

```
        se X > Y
        entao X := 10
        senao X := 0;
        fim_se;
fim.
```

Obs.: Se mais de um comando for necessário dentro das estruturas "entao" e "senao", estes comandos devem ser seguidos de delimitadores de bloco de programa (ver item 2.3.2).

A3.10 Comandos de repetição

Em alguns casos é necessário repetir uma parte do algoritmo um determinado número de vezes. Para isto estão disponíveis as estruturas de repetição.

A3.10.1 Enquanto .. faca

Executa um comando simples ou composto repetidas vezes, enquanto uma determinada condição permanece válida (verdadeira). Esta estrutura faz o teste da condição antes de iniciar a repetição; se o primeiro teste falhar, o bloco de instruções dentro do *looping* não será executado nenhuma vez.

Sua forma é: enquanto <condição> faça <comando>

Exemplo:

```
programa ILUSTRA_ENQUANTO_FACA;
variavel
        X,
        Y,
        A,
        S  : inteiro;
inicio
        A := 0;
        leia_ln(S);
        leia_ln(X);
        leia_ln(Y);
        enquanto A < 10 faca
                S := X + Y;
```

```
            X := Y;
            Y := S;
            A := A + 1;
        fim_enquanto;
        imprima_ln(S,' ',X,' ',Y,' ',A);
fim.
```

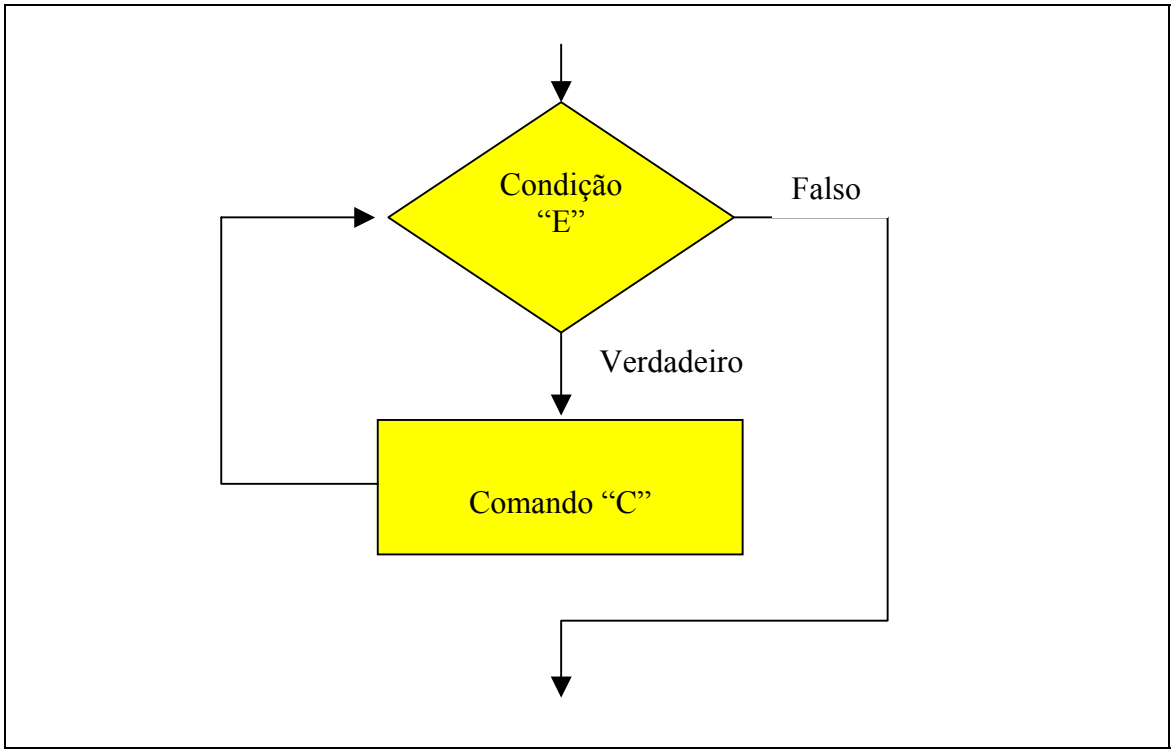

FIGURA A3.6 - Diagrama de fluxo da estrutura ENQUANTO...FACA.

A3.10.2 Repita .. ate_que

Os comandos dentro do bloco desta estrutura serão executados pelo menos uma vez. Quando a condição é encontrada, ela é testada. Se for verdadeira passa o controle para o comando imediatamente abaixo. Se for falsa, os comandos do bloco são novamente executados, até que se tenha uma condição verdadeira.

Sua forma é: Repita <comando(s)> ate_que <condição>

Exemplo:

```
programa ILUSTRA_REPITA_ATE_QUE;
variavel
        A, X, Y, Z: inteiro;
inicio
```

```
A := 1;
Z := 3;
Y := 9;
repita
        X := Y + Z;
        A := A + 1;
ate_que A > 10;
fim.
```

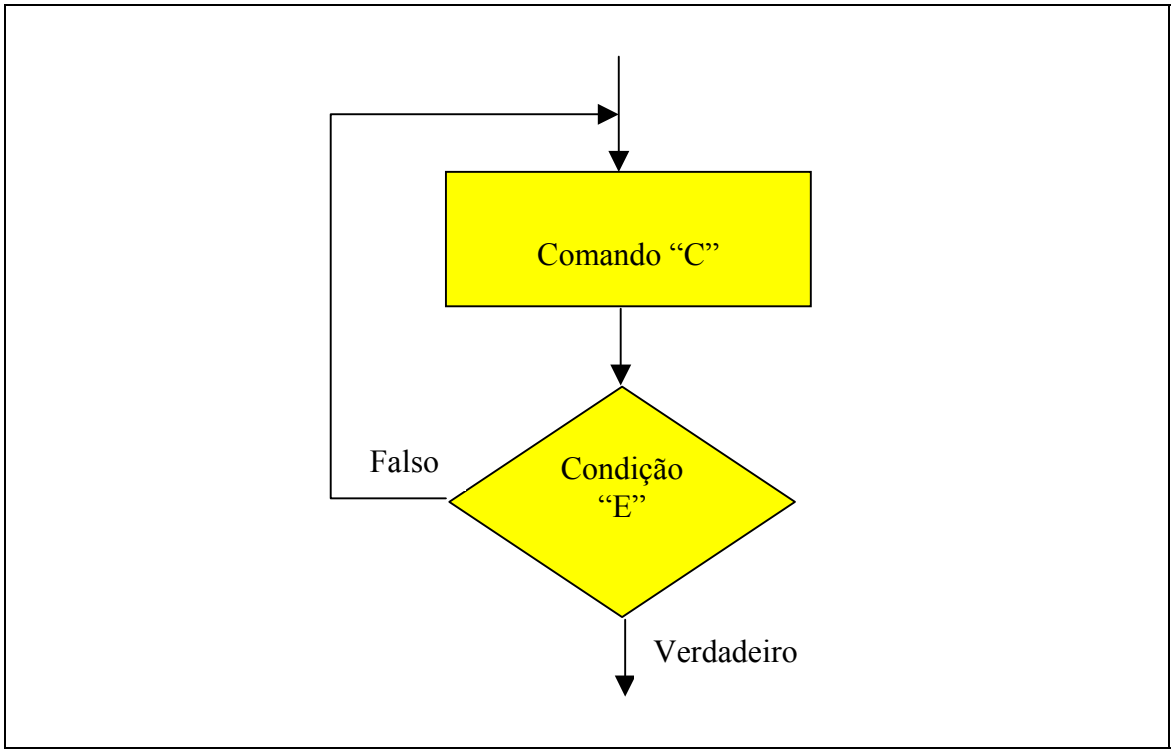

FIGURA A3.7 - Diagrama de fluxo da estrutura REPITA..ATE_QUE.

A3.10.3 Para .. ate .. faca

Esta estrutura nos permite executar um comando ou vários comandos um número específico de vezes.

Sua forma é: Para <variável de controle> de <valor inicial> ate <valor final> faca <comando(s)>

Utiliza-se uma variável a ser incrementada de um valor inicial para um valor final. Os comandos serão executados tantas vezes quantas forem o incremento da variável. Este incremento é de um em um e a variável utilizada para controlá-lo não deverá sofrer alterações dentro da estrutura 'Para'. É recomendável o uso desta instrução sempre que se souber o valor inicial e o valor final da variável de controle.

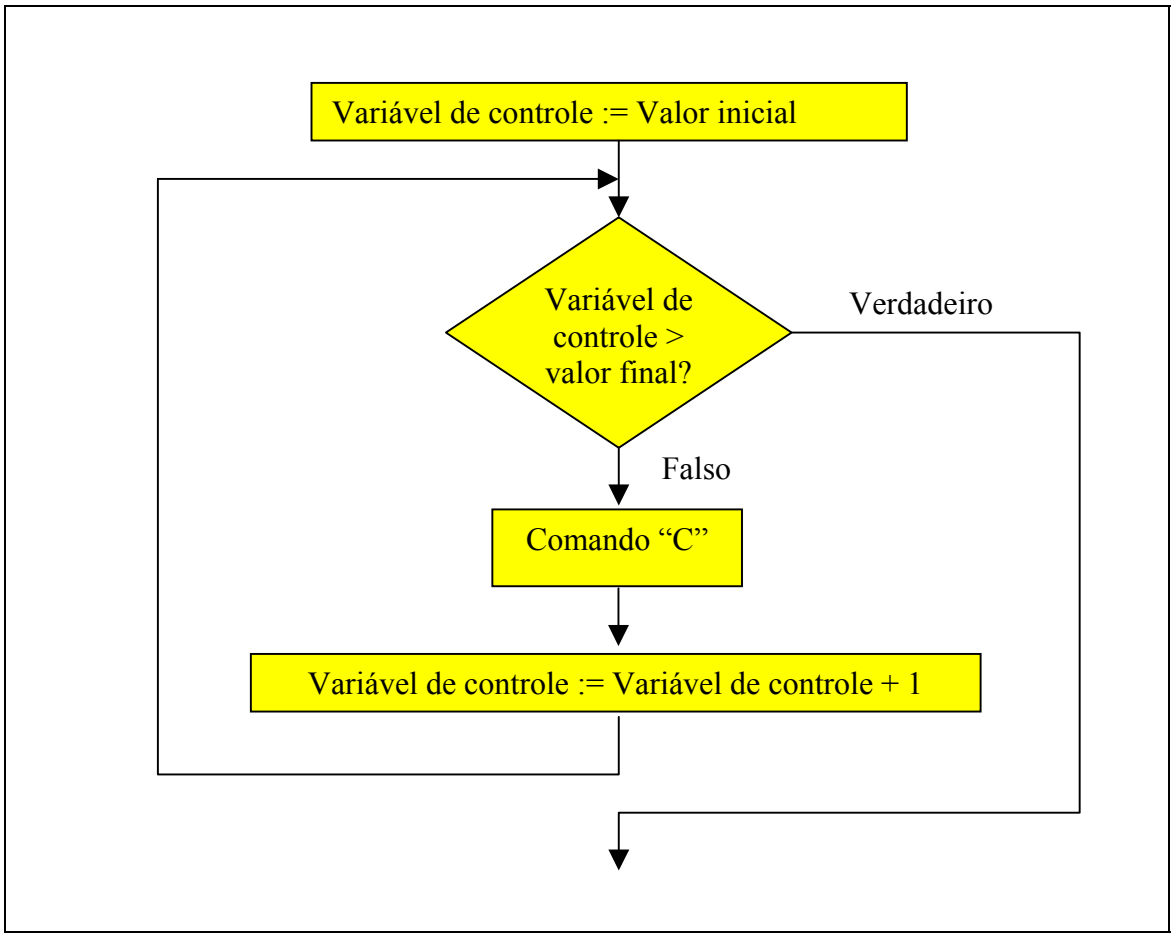

FIGURA A3.8 - Diagrama de fluxo da estrutura PARA..ATE..FACA.

Exemplo:

```
programa ILUSTRA_PARA_ATE_FACA;
variavel
        I, A, B : inteiro;
inicio
        A := 39;
        B := 23;
        para I de 1 ate 10 faca
                A := A + B;
                B := B + 1;
        fim_para;
        imprima_ln('A -> ',A,'   B -> ',B);
fim.
```

A3.11 Comandos de entrada e saída de dados

Todo programa requer alguma forma de entrada e saída de dados e informações. Um programa não teria razão se não houvesse possibilidade de entrar com valores para processamento ou mostrar os resultados desse processamento.

Em alguns programas pode ser necessário incluir alguns dados de entrada. Esses dados serão processados e os resultados serão mostrados ao exterior. Uma das maneiras de exteriorizar os resultados de determinado processamento é através da impressão destes. Para que isto seja possível, temos dois comandos de entrada: **leia** e **leia_ln**; e dois comandos de saída: **imprima** e **imprima_ln**.

A3.11.1 Leia e Leia_ln

Estes comandos permitem que o usuário informe dados de entrada para o algoritmo. As sintaxes destes comandos são as seguintes:

a) Leia:

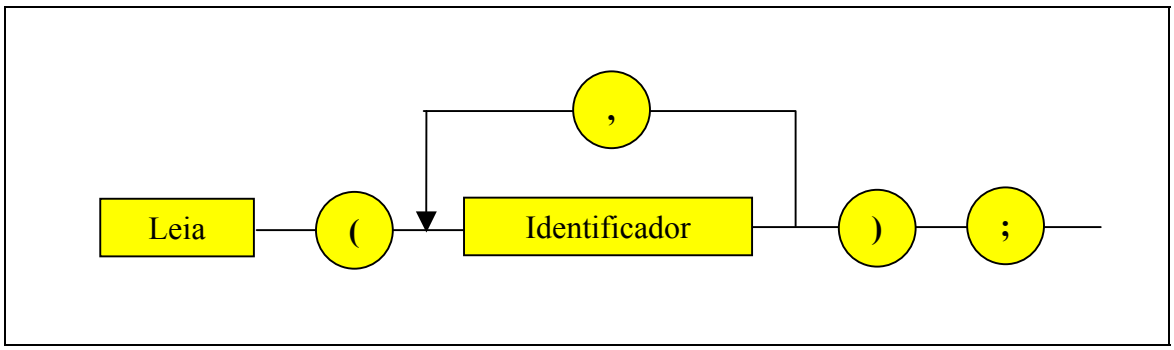

FIGURA A3.9 - Diagrama de fluxo do comando LEIA.

b) Leia_ln:

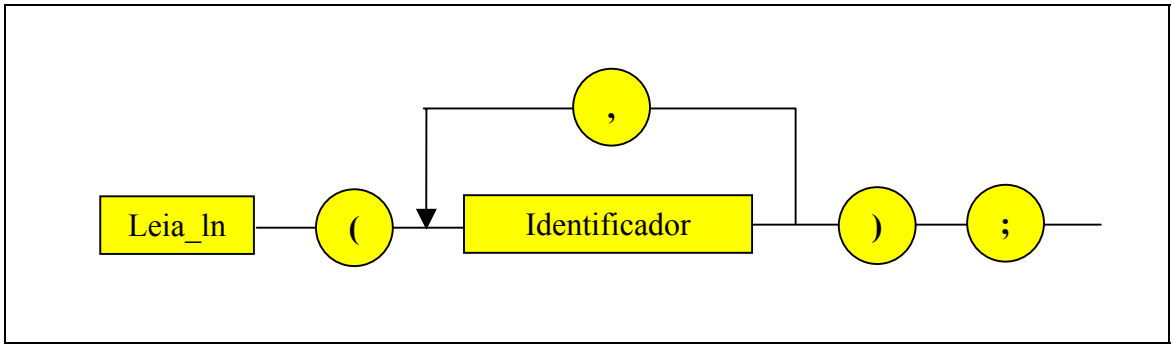

FIGURA A3.10 - Diagrama de fluxo do comando LEIA_LN.

Exemplos:

```
leia(VALOR);
leia(A,B,CONTADOR);
leia_ln(VALOR);
leia_ln(CODIGO, NOME);
```

O comando **Leia** lê as variáveis da lista e mantém o cursor preparado de tal forma que o próximo comando **Leia**, se houver, vai procurar o valor a ser lido na mesma linha.

Exemplo:

```
programa ILUSTRA_LEIA;
variavel
        NOME : string[20];
        NOTA1, NOTA2 : real;
inicio
        leia(NOME);
        leia(NOTA1);
        leia(NOTA2);
        imprima_ln(NOME,' ',NOTA1,' ',NOTA2);
fim.
```

O comando **Leia_ln** lê os dados da lista, de acordo com as variáveis listadas e posiciona o cursor na próxima linha.

Exemplo:

```
programa ILUSTRA_LEIA_LN;
variavel
        NOME : string[20];
        NOTA1, NOTA2 : real;
inicio
        leia_ln(NOME);
        leia_ln(NOTA1);
        leia_ln(NOTA2);
fim.
```

Ao executar o programa EXEMPLO_LEIA_LN observamos que, se pressionarmos <ENTER> depois de digitar o nome, o cursor muda de linha. Fato idêntico ocorre com cada nota. Isto confirma o fato de NOME, NOTA1 e NOTA2 serem digitados cada um numa linha.

A3.11.2 Imprima e Imprima_ln

Estes dois comandos permitem apresentar ao usuário mensagens e dados contidos em identificadores. As sintaxes dos comandos de saída são apresentadas nas duas figuras a seguir:

a) Imprima:

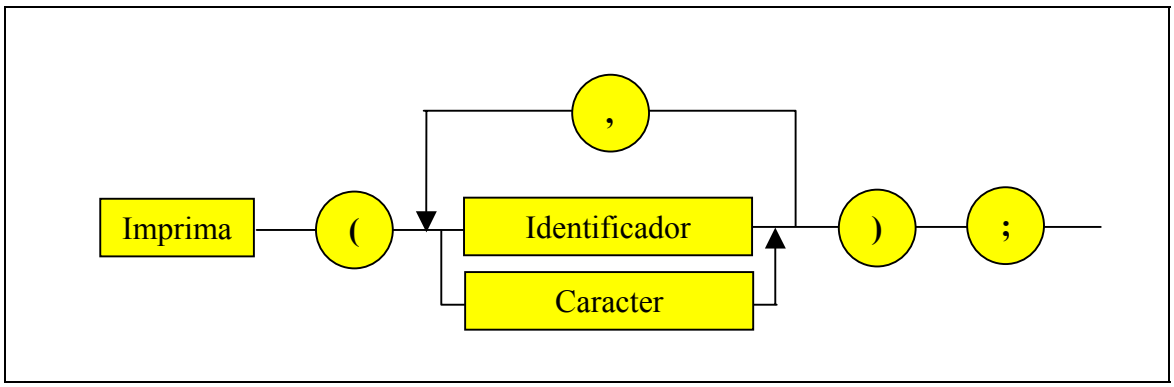

FIGURA A3.11 - Diagrama de fluxo do comando IMPRIMA.

b) Imprima_ln:

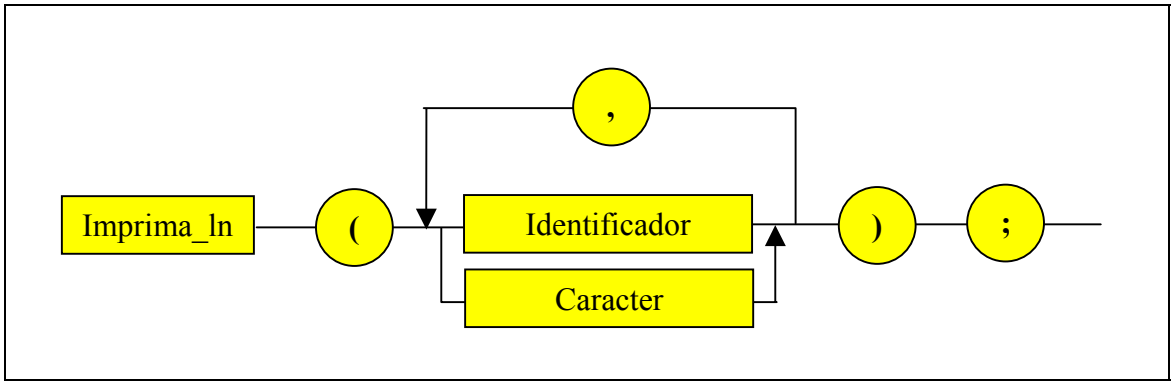

FIGURA A3.12 - Diagrama de fluxo do comando IMPRIMA_LN.

Exemplos:

imprima(SOMA);

imprima(“Media = “,MED);

imprima_ln(VALOR);

imprima_ln(“Valor total = “,VALOR);

O comando **Imprima** escreve o que é pedido e mantém o cursor na mesma linha, indicando que o próximo comando **Imprima**, se houver, vai escrever outras informações na mesma linha, ou seja, na frente do que acabou de escrever. Se utilizarmos as instruções:

```
imprima('O MAIOR PAI DO MUNDO:');
imprima(' O MEU.');
```

aparecera no vídeo:

```
O MELHOR PAI DO MUNDO: O MEU.
```

O comando **Imprima_ln** escreve o que é pedido e em seguida avança para a próxima linha. Se utilizarmos as instruções:

```
imprima_ln('O MELHOR LUGAR PARA PASSEAR:');
imprima_ln('O CAMPO.');
```

será exibido no vídeo:

```
O MELHOR LUGAR PARA PASSEAR:
O CAMPO.
```

Um programa simples ilustrando os comandos leia_ln e imprima_ln é mostrado a seguir:

```
programa EXEMPLO_1;
variavel
    a : inteiro;
    b : inteiro;
    soma : inteiro;
inicio
    leia_ln(a);
    leia_ln(b);
    soma := a + b;
    imprima_ln('A soma de A e B = ',soma);
fim.
```

A3.12 Procedimentos

Os procedimentos são subrotinas de programas que ajudam na construção de códigos estruturados. As subrotinas ou subprogramas são, na realidade, programas com vida

própria, mas que para serem processados têm que ser solicitados pelo programa principal que o contém, ou por outro subprograma.

Em PCP, os procedimentos devem iniciar pelo identificador **procedimento**, seguido por outro identificador que representa o cabeçalho do mesmo, o identificador **inicio_proc**, que indica o início do bloco de comandos, e depois finalizado com o identificador **fim_proc**.

Exemplo:

```
procedimento SOMA;
variavel
        A,B,SOM : inteiro;
inicio_proc
        A:= 2;
        B:= 4;
        SOM := A + B;
fim_proc;
```

As variáveis declaradas dentro dos procedimentos, conhecidas como variáveis locais, não poderão ser utilizadas fora dos mesmos e nenhuma variável local poderá ter o mesmo nome de alguma outra variável local e vice-versa.

O exemplo a seguir mostra como um procedimento deve ser inserido no programa principal.

```
programa ILUSTRA_PROCEDIMENTO;
        { Cabeçalho do programa principal (ILUSTRA_PROCEDIMENTO) }
variavel
        { Seção de declaração de variáveis globais }
        K : inteiro;
procedimento CALCULAR;
        { Cabeçalho do procedimento CALCULAR }
inicio_proc
        { Comandos do procedimento CALCULAR }
fim_proc;

inicio
        { Inicio dos comandos do programa principal }
```

```
    CALCULAR;  { Chamada ao procedimento }
    K := K + 1;
fim. { Fim do programa }
```

A3.13 Comentários

Os comentários, cuja importância é indiscutível dentro de um programa, serão delimitados por "{ }" (chaves) e podem aparecer em qualquer lugar do programa. Dentro dos delimitadores de comentário não pode haver outros delimitadores, ou seja, não são permitidos comentários aninhados. Os comentários não influenciarão no funcionamento do programa pois os analisadores do PCP irão ignorar tudo que estiver contido dentro dos delimitadores "{ }". Portanto, recomendamos a utilização constante dos comentários pois estes não afetarão em nada no programa exceto promover maiores esclarecimentos sobre determinadas partes do algoritmo.

Uma restrição apontada é que os comentários deverão iniciar e finalizar numa mesma linha, ou seja, os delimitadores de inicio e fim de comentários "{ }" deverão aparecer na mesma linha. Caso seja necessário acrescentar comentários que não caibam numa única linha, é permitido escrever diversas linhas consecutivas com comentários, bastando inserir os delimitadores em cada linha.

Exemplos de comentários:

```
    leia(a,b);   { Leitura das variáveis A e B }
    imprima_ln('A soma de A e B = ',soma);      {Escreve a soma dos valores}
                                                {informados pelo usuário}
```

Bibliografia

[AHO 74] AHO, Alfred V.; ULLMAN, Jeffrey D. **The theory of parsing, translation, and compiling**. Englewood Cliffs: Prentice-Hall, 1974.

[BEC 97] BECK, Leland L. **Desenvolvimento de software básico:** Assemblers, linkers, loaders, compiladores, sistemas operacionais, bancos de dados e processadores de textos. Rio de Janeiro: Campus, 1994.

[BEH 93] BEHAR, Patrícia Alejandra. **Avaliação de softwares educacionais no processo de ensino-aprendizagem computadorizado:** estudo de caso. Porto Alegre: CPGCC da UFRGS, 1993. Dissertação de Mestrado.

[BEH 98] BEHAR, Patrícia Alejandra. **Análise operatória de ferramentas computacionais de uso individual e cooperativo**. Porto Alegre: CPGCC da UFRGS, 1998. Tese de Doutorado.

[BJO 85] BJORNER, Dines; JONES, Cliff B. **Formal specification & software development**. England: Prentice-Hall, 1985.

[CAR 88] CARROL, David W. **Programação em Turbo Pascal**. São Paulo: McGraw-Hill, 1988.

[CLA 94] CLAUDIO, Dalcidio Moraes; VICCARI, Rosa Maria. **Compêncio de trabalhos sobre informática na educação**. Porto Alegre: CPGCC da UFRGS, 1994. 2v.

[DOB 86] DOBB, Dr. **Toolbook of C**. New York: Brady Book, 1986.

[FAR 89] FARRER, H. et al. **Algoritmos estruturados**. 2. ed. São Paulo: Guanabara, 1989. p. 23-25

[GLO 9?] GLOOR, Peter; DYNES, Scott; LEE, Irenea. **Animated Algorithms:** A hypermedia learning environment for introduction to algorithms. Cambridge: Mit Press, [199?].

[GRI 93] GRILLO, Maria Célia Arruda. **Turbo Pascal**. 5. ed. Rio de Janeiro: LTC, 1993.

[GUI 91] GUILHERME, Vera Maria. **Produção e avaliação de software educacionais:** relação entre teoria e prática. Porto Alegre: Faculdade de Educação, Universidade Federal do Rio Grande do Sul, 1991. Dissertação de Mestrado.

[GUI 95] GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e Estruturas de Dados**. Rio de Janeiro: LTC, 1995.

Algoritmos e Estruturas de Dados. Rio de Janeiro: LTC, 1995.

[HUN 87] HUNTER, Robin. **Compiladores – Sua concepção e programação em Pascal**. Lisboa: Presença, 1987.

[JOS 87] JOSÉ NETO, João. **Introdução à compilação.** Rio de Janeiro: LTC, 1987.

[RIN 93] RINALDI, Roberto. **Turbo Pascal 7.0:** comandos e funções. São Paulo: Érica, 1993.

[SAL 98] SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen. **Algoritmos**. São Paulo: Makron Books, 1998.

[SCH 88] SCHILDT, Herbert. **Turbo C – Guia do usuário**. 2. ed. São Paulo: McGraw-Hill, 1988.

[SCH 90] SCHILDT, Herbert. **C completo e total.** São Paulo: Makron Books, MacGraw-Hill, 1990.

[SCH 90] SCHILDT, Herbert. **Turbo C avançado – Guia do usuário**. São Paulo: McGraw-Hill, 1990.

[SEB 99] SEBESTA, Robert W. **Concepts of programming languages**. 4th ed. USA: Addison-Wesley, 1999.

[SET 89] SETZER, Valdemar W.; MELO, Inês S. Homem de. **A construção de um compilador**. 3. ed. Rio de Janeiro: Campus, 1989.

[SIP 92] SIPPU, Seppo; SOISALON-SOININEN, Eljas. **Parsing Theory:** Languages and Parsing. Helsinki: Springer-Verlag, 1992. v.1.

[UFR 99] UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. Instituto de Informática.Biblioteca. **Normas para apresentação de monografias do Instituto de Informática e do CPGCC**. Porto Alegre, 1997. Disponível em: <http://www.inf.ufrgs.br/biblioteca/html/normas.htm>. Acesso em: 10 jan. 1999.

[VEL 87] VELOSO, Paulo A. S. **Estrutura e verificação de programas com tipos de dados**. São Paulo: E. Blucher, 1987.