Edgard de Paiva Melo Filho
Maicon Gomes Messias

Relatório de Pesquisa LP Pascal
Trabalho Teórico Prático
Disciplina: Linguagens de Programação

Belo Horizonte
Set, 2025

Edgard de Paiva Melo Filho
Maicon Gomes Messias

Relatório de Pesquisa LP Pascal
Trabalho Teórico Prático
Disciplina: Linguagens de Programação

Trabalho apresentado à disciplina de Linguagens de Programação do Curso de Engenharia de Computação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para avaliação.

Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Curso de Engenharia de Computação
Professor Orientador: Marco Rodrigo Costa

Belo Horizonte
Set, 2025

Sumário

1 Introdução

A linguagem de programação Pascal surgiu em 1970, criada por Niklaus Wirth na Universidade Federal de Zurique, com o objetivo de oferecer uma linguagem estruturada, clara e eficiente para o ensino da programação e da ciência da computação. Diferentemente de muitas linguagens de sua época, como Fortran e Algol, Pascal foi projetada não apenas para resolver problemas práticos, mas, sobretudo, para servir como uma ferramenta pedagógica capaz de transmitir os fundamentos da programação estruturada, promovendo a lógica, a organização e a disciplina no desenvolvimento de algoritmos.

Com uma sintaxe simples e legível, Pascal rapidamente se destacou como uma linguagem didática por excelência. Sua tipagem forte evitava erros comuns de programação, enquanto os recursos de modularização permitiam a construção de programas bem estruturados e de fácil manutenção. Esse conjunto de características fez com que Pascal fosse amplamente adotada em universidades, escolas técnicas e cursos de introdução à programação durante as décadas de 1970, 1980 e início de 1990, consolidando sua relevância no meio acadêmico.

Além de seu papel educacional, Pascal também foi utilizada em aplicações práticas. Compiladores eficientes e ambientes de desenvolvimento como o Turbo Pascal, lançado pela Borland em 1983, possibilitaram o uso da linguagem em projetos reais, incluindo softwares comerciais e sistemas de pequeno a médio porte. Esse avanço contribuiu para que Pascal deixasse de ser apenas uma linguagem acadêmica e alcançasse um espaço no mercado tecnológico da época.

Outro aspecto relevante é a sua influência histórica. Pascal serviu como base para a criação de outras linguagens, entre as quais se destacam Modula-2, Oberon e, principalmente, Delphi, que manteve a simplicidade e a clareza da linguagem original, mas incorporou recursos de orientação a objetos e interfaces gráficas, ampliando seu alcance para aplicações modernas.

Mesmo não sendo mais amplamente utilizada no cenário atual, dominado por linguagens como Python, Java e C++, o legado de Pascal permanece evidente. Seu impacto no ensino da programação estruturada formou gerações de profissionais, além de ter introduzido conceitos que se tornaram fundamentais para a evolução da ciência da computação.

Dessa forma, este relatório busca apresentar uma análise abrangente da linguagem de programação Pascal, abordando sua origem, características principais, aplicações educacionais e práticas, bem como sua influência histórica no desenvolvimento de novas tecnologias e metodologias de ensino.

2 Histórico sobre a Linguagem

O objetivo inicial de Wirth era criar uma linguagem de programação que fosse clara, simples e adequada ao ensino da **programação estruturada**. Diferentemente de outras linguagens de sua época, como Fortran e Algol, Pascal foi projetada com foco pedagógico, permitindo que estudantes aprendessem conceitos fundamentais, como estruturas de controle, modularização e tipagem forte.

Durante as décadas de 1970 e 1980, Pascal consolidou-se como uma das linguagens mais importantes em ambientes acadêmicos, sendo amplamente utilizada em universidades e escolas técnicas. Além do aspecto didático, a criação de ambientes de desenvolvimento como o **Turbo Pascal**, lançado pela Borland em 1983, possibilitou a aplicação prática da linguagem em softwares comerciais e sistemas de pequeno e médio porte.

Com o passar do tempo, Pascal perdeu espaço no mercado frente a linguagens como C e C++, mas continuou a desempenhar papel central na formação de gerações de programadores. Sua influência histórica é evidente no surgimento de outras linguagens, como Modula-2, Oberon e Delphi, que incorporaram e expandiram seus conceitos.

2.1 Cronologia da Linguagem Pascal

A seguir, apresenta-se a linha do tempo com os principais marcos da evolução da linguagem Pascal:

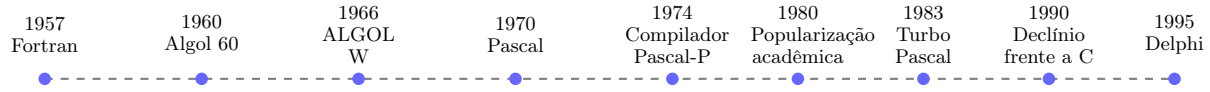

Figura 1 – Linha do tempo da evolução da linguagem Pascal

2.2 Genealogia da Linguagem Pascal

A genealogia de Pascal mostra suas influências diretas e as linguagens que dela derivaram. Pascal recebeu forte influência do Algol 60 e de sua extensão ALGOL W, além de incorporar princípios do Fortran. Por sua vez, originou linguagens importantes, como Modula-2, Oberon, Delphi/Object Pascal e influenciou Ada.

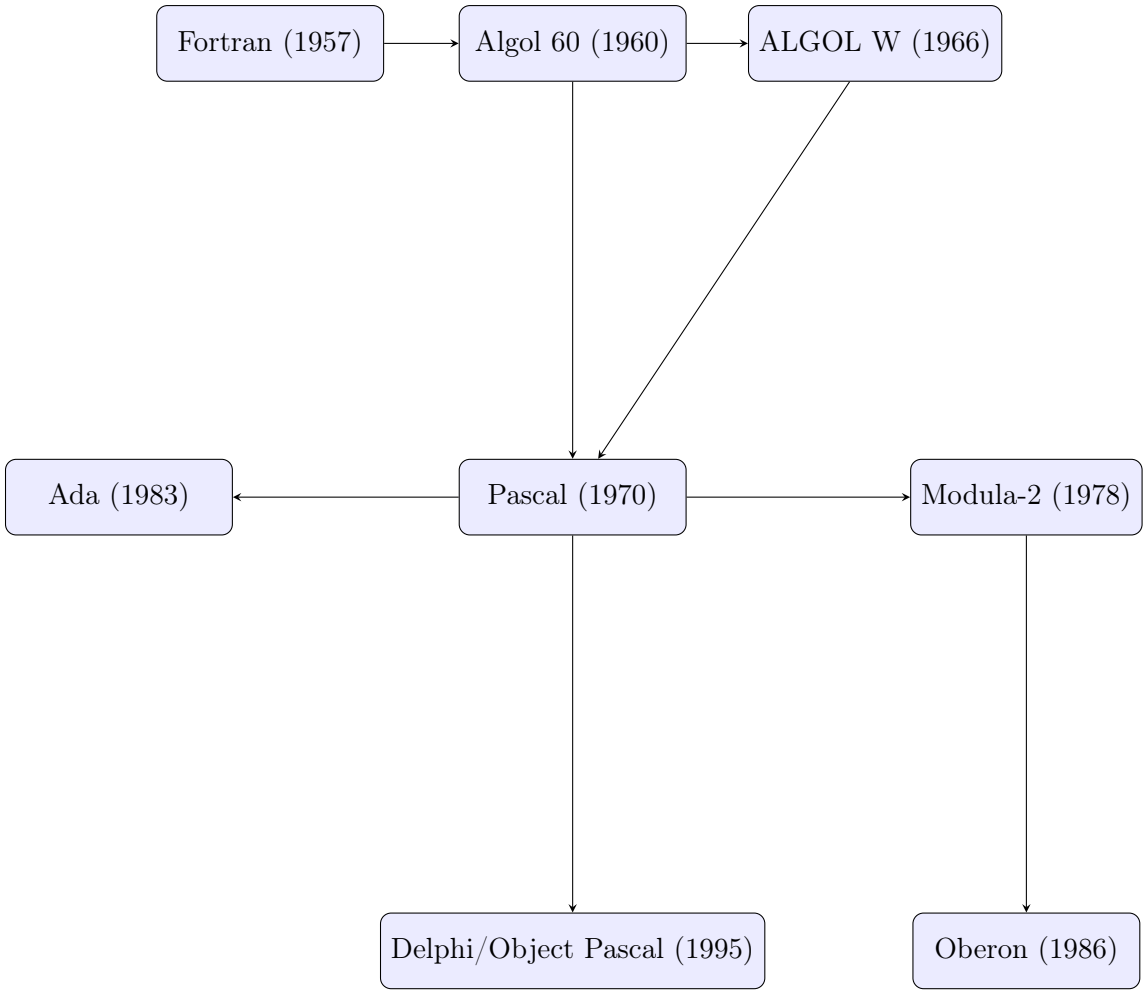

Figura 2 – Genealogia da linguagem Pascal e suas influências

3 Paradigmas a que Pertence

O Pascal é um marco na história da computação, conhecido por ser uma linguagem de programação didática e com um design bem pensado. Seus principais paradigmas são o **paradigma imperativo**, o **paradigma procedural** e o **paradigma estruturado**.

3.1 Paradigma Imperativo

O programa é descrito como uma sequência de comandos que alteram o estado do computador, tendo instruções claras de 'o que fazer' e 'como fazer'.

3.2 Paradigma Procedural

No paradigma procedural, o código é organizado em **procedimentos** e **funções**. Em vez de focar em objetos ou em lógica declarativa, o programador define uma série de passos ou instruções que devem ser executadas em uma ordem específica para atingir um resultado. O Pascal se encaixa perfeitamente nesse modelo:

- **Sub-rotinas:** O código é dividido em pequenos blocos, como `PROCEDURE` (para ações sem retorno de valor) e `FUNCTION` (para operações que retornam um valor). Isso ajuda a organizar o código e a evitar a repetição de tarefas.
- **Sequência de execução:** O programa é uma sequência de chamadas a essas sub-rotinas, seguindo um fluxo de controle linear e bem definido.
- **Variáveis globais:** O compartilhamento de dados entre diferentes partes do programa é frequentemente feito através de variáveis acessíveis por todo o código, embora isso possa gerar complexidade em projetos maiores.

Exemplo em Pascal:

```
procedure ExibirMensagem;
begin
  writeln('Olá, mundo!');
end;

begin
  ExibirMensagem;
end.
```

3.3 Paradigma Estruturado

O paradigma estruturado, fortemente ligado ao procedural, busca criar programas mais lógicos, fáceis de ler e de dar manutenção. Ele surgiu como uma alternativa ao uso excessivo do `GOTO`, que criava um "código de espaguete"confuso. O Pascal foi explicitamente projetado para forçar o uso de estruturas de controle de fluxo mais claras e organizadas.

- **Estruturas de controle:** O Pascal oferece comandos como `IF...THEN...ELSE` (para decisões), `FOR` (para laços com contagem definida), `WHILE` e `REPEAT...UNTIL` (para laços com condição), que organizam o fluxo de execução do programa de forma previsível.
- **Blocos de código:** O uso das palavras-chave `BEGIN` e `END` para agrupar comandos permite que o programador defina claramente o escopo de cada estrutura de controle.
- **Modularidade:** A divisão do código em procedimentos e funções promove a **modularidade**, ou seja, a capacidade de dividir o programa em módulos ou partes independentes. Isso facilita a depuração e o reuso de código.

Exemplo em Pascal:

```
var
  i: integer;
begin
  for i := 1 to 5 do
    writeln('Número: ', i);
end.
```

Em resumo, Pascal é uma linguagem que exemplifica a programação **procedural** e **estruturada**, com ênfase na organização lógica do código através de sub-rotinas e estruturas de controle de fluxo bem definidas. Essa filosofia de design foi fundamental para ensinar boas práticas de programação e influenciou muitas linguagens modernas.

4 Características Marcantes da Linguagem Pascal

A linguagem Pascal possui diversas características que a tornam uma linguagem robusta, segura e didática. A seguir, detalhamos suas principais características:

4.1 Fortemente Tipada

Pascal exige que todas as variáveis tenham um tipo definido antes de serem usadas. Isso evita operações entre tipos incompatíveis, como somar inteiros com strings, garantindo maior segurança e confiabilidade do programa.

4.1.1 Tipos de Dados em Pascal

A linguagem Pascal dispõe dos tipos de dados apresentados na Tabela 1.

Tabela 1 – Tipos de dados do Pascal

Denominação	**Conjunto de valores**
Char	Caracteres codificados no código ASCII (letras, dígitos e caracteres especiais como `:`, `?`, `*`, etc.)
Shortint	Números inteiros do intervalo $[-128, 127]$
Integer	Números inteiros do intervalo $[-32768, 32767]$
Longint	Números inteiros do intervalo $[-2\,147\,483\,648, 2\,147\,483\,647]$
Byte	Números inteiros do intervalo $[0, 255]$
Word	Números inteiros do intervalo $[0, 65535]$
Real	Números reais do conjunto $[-3.4 \times 10^{38}, -3.4 \times 10^{-38}] \cup [3.4 \times 10^{-38}, 3.4 \times 10^{38}]$
Boolean	Conjunto dos valores `false` (falso) e `true` (verdadeiro)

À exceção do tipo `real`, os tipos de dados acima são ordenados no sentido de que existe um primeiro elemento e existem as ideias de sucessor e de antecessor. A ordenação dos tipos de dados que são subconjuntos dos inteiros é a ordenação natural da matemática; a do tipo de dado `char` é dada pelo Código ASCII e a do tipo de dado `boolean` é dada por {`false`, `true`}. Uma observação importante é que o tipo `real`, rigorosamente falando, não armazena números reais e sim números de um *sistema de ponto flutuante*, que não contém todos os reais, mas apenas os representáveis. O estudo de sistemas de ponto flutuante foge ao escopo deste trabalho e é feito em disciplinas como *Organização e Arquitetura de Computadores* e *Cálculo Numérico*.

4.2 Palavras Reservadas da Linguagem Pascal

Assim como toda linguagem de programação, Pascal possui identificadores específicos para sua sintaxe, conhecidos como palavras reservadas e, portanto, não podem ser utilizados como nomes de variáveis, funções ou outros elementos do programa. A seguir, apresenta-se a listagem das palavras reservadas:

Tabela 2 – Palavras reservadas da linguagem Pascal

and	downto	in	repeat	then	xor
asm	else	inline	packed	to	case
array	begin	exports	label	program	until
const	constructor	for	mod	new	type
div	do	implementation	of	string	var
end	file	nil	record	unit	with
if	object	set	while		

Se faz ainda importante assim como em todas linguagens de programação, a consideração da sintaxe utilizada para comentarios no Pascal, que utiliza de chaves '{ }' para preparar comentarios no código, exemplos de utilização podem ser vistos nos exemplos apresentados no capitulo 6.

4.3 Operadores e Expressões em Pascal

Os compiladores da linguagem Pascal oferecem suporte a **expressões aritméticas** e **relacionais**, permitindo a realização de cálculos e comparações entre valores.

4.3.1 Operadores Aritméticos

As operações aritméticas envolvem soma, subtração, multiplicação e divisão, além da troca de sinal (operador unário). Operadores como `+`, `-`, `*`, `/` podem atuar sobre operandos dos tipos `integer` ou `real`, enquanto `div` e `mod` são específicos para inteiros, representando respectivamente o quociente e o resto da divisão inteira.

Tabela 3 – Operadores aritméticos em Pascal

Operador	**Operação**
+	Adição
-	Subtração
*	Multiplicação
/	Divisão (resulta em `real`)
div	Divisão inteira (quociente)
mod	Resto da divisão inteira

A avaliação das expressões segue uma hierarquia de precedência semelhante à da matemática, onde multiplicação, divisão, `div` e `mod` possuem maior prioridade que soma e subtração. Parênteses podem alterar essa ordem.

4.3.2 Operadores Relacionais

As expressões relacionais permitem comparações entre valores numéricos e também entre caracteres (de acordo com a ordem ASCII). O resultado de uma relação é sempre um valor do tipo `boolean`, podendo ser `true` (verdadeiro) ou `false` (falso).

Tabela 4 – Operadores relacionais em Pascal

Operador	**Operação**
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
=	Igual
<>	Diferente

Assim como na matemática, a prioridade dos operadores relacionais é inferior à dos operadores aritméticos. Dessa forma, em expressões mistas, primeiro ocorre a avaliação dos cálculos, e em seguida, a verificação da relação.

4.4 Estruturada

A linguagem segue o paradigma estruturado, utilizando blocos de código delimitados por `begin` e `end`. Isso favorece um fluxo lógico claro e reduz o uso do comando `goto`, facilitando a leitura e manutenção do código, conforme demonstrado nos exemplos expostos anteriormente e na seção de exemplos (Seção 6).

4.5 Paradigma Procedural

Pascal é uma linguagem procedural, ou seja, os programas são compostos por procedimentos e funções que executam tarefas específicas. A modularização do código permite a reutilização de funções e simplifica a depuração.

4.5.1 Modularidade devido ao paradigma

Pascal suporta unidades (`units`), que permitem separar o programa em módulos distintos. Cada módulo pode conter suas funções e procedimentos, facilitando o desenvolvimento e a manutenção de programas grandes.

4.6 Legibilidade e Sintaxe Clara

A sintaxe de Pascal é próxima da linguagem natural, com comandos como `if`, `for`, `while`, `repeat` e `case`. Essa característica torna a linguagem ideal para o ensino de programação e para iniciantes, pois facilita a compreensão da lógica do programa.

4.7 Segurança e Controle de Erros

A tipagem forte e a estrutura organizada da linguagem previnem erros comuns, garantindo que operações inválidas sejam detectadas em tempo de compilação e promovendo programação segura.

4.8 Suporte a Estruturas de Dados

Pascal permite a manipulação de arrays, registros (`records`), conjuntos (`sets`) e arquivos, oferecendo flexibilidade para implementação de algoritmos e organização eficiente dos dados.

4.9 Facilidade de Aprendizado e Ensino

Devido à clareza da sintaxe, tipagem rigorosa e fluxo lógico bem definido, Pascal é uma excelente linguagem didática, utilizada amplamente para ensinar algoritmos, lógica de programação e boas práticas de desenvolvimento.

4.10 Portabilidade

Apesar de ser uma linguagem antiga, Pascal possui compiladores para diversas plataformas. Programas bem estruturados podem ser facilmente adaptados ou migrados para outras linguagens estruturadas modernas.

4.11 Resumo Visual das Características

Tabela 5 – Características marcantes da linguagem Pascal

Característica	**Descrição**
Fortemente Tipada	Garante que todas as variáveis tenham tipos definidos, prevenindo erros de operação entre tipos incompatíveis.
Estruturada	Utiliza blocos de código (`begin` ... `end`) e fluxo lógico claro, evitando uso excessivo de `goto`.
Paradigma Procedural	Organização do código em procedimentos e funções, permitindo modularização e reutilização de código.
Modularidade	Suporte a unidades (`units`) que separam o programa em módulos, facilitando manutenção.
Legibilidade	Sintaxe clara, próxima da linguagem natural, ideal para ensino e iniciantes.
Segurança	Reduz erros comuns por tipagem forte e estrutura organizada do código.
Suporte a Estruturas de Dados	Arrays, registros, conjuntos e arquivos, permitindo organização eficiente de dados.
Facilidade de Aprendizado	Linguagem didática, adequada para ensinar algoritmos e lógica de programação.
Portabilidade	Código estruturado pode ser adaptado para outras linguagens e plataformas.

5 Linguagens Relacionadas

A linguagem Pascal foi diretamente influenciada por linguagens anteriores e serviu como base para o desenvolvimento de outras linguagens modernas, desempenhando um papel central na evolução da programação estruturada.

5.1 Linguagens que Influenciaram Pascal

- **Algol 60 (1960):** Pascal herdou a sintaxe clara e o conceito de programação estruturada de Algol 60, uma das primeiras linguagens a introduzir blocos de código e estruturas de controle modernas.
- **ALGOL W (1966):** Desenvolvida por Niklaus Wirth, ALGOL W foi um precursor direto do Pascal, com melhorias em tipos de dados e modularidade que foram refinadas no Pascal.
- **Fortran (1957):** Embora menos estruturado, Fortran influenciou Pascal em aspectos como manipulação de arrays e operações aritméticas.

5.2 Linguagens Influenciadas por Pascal

- **Modula-2 (1978):** Criada por Wirth como uma evolução do Pascal, Modula-2 introduziu módulos mais robustos e suporte a programação concorrente.
- **Oberon (1986):** Outra criação de Wirth, Oberon simplificou o Pascal, mantendo sua clareza e adicionando recursos para sistemas operacionais.
- **Delphi/Object Pascal (1995):** Desenvolvida pela Borland, Delphi estendeu o Pascal com orientação a objetos e interfaces gráficas, tornando-se popular para desenvolvimento de aplicações Windows.
- **Ada (1983):** Embora não seja uma derivação direta, Ada foi fortemente influenciada pelos conceitos de tipagem forte e modularidade do Pascal.

5.3 Linguagens Similares

Linguagens como **C** e **C++** compartilham semelhanças com Pascal no uso de programação estruturada e procedural, mas diferem na flexibilidade (C é menos rígido na tipagem) e na complexidade (C++ adiciona orientação a objetos).

6 Exemplo(s) de Programa(s)

A seguir, apresentamos dois exemplos de programas em Pascal que ilustram suas características estruturadas e didáticas.

6.1 Exemplo 1: Cálculo da Média de Três Números

Este programa solicita três números ao usuário, calcula a média e exibe o resultado.

```
program CalculaMedia;
var
  num1, num2, num3, media: real;
begin
  writeln('Digite três números:');
  readln(num1);
  readln(num2);
  readln(num3);
  media := (num1 + num2 + num3) / 3;
  writeln('A média é: ', media:0:2);
end.
```

6.2 Exemplo 2: FizzBuzz

O clássico problema FizzBuzz imprime números de 1 a 20, substituindo múltiplos de 3 por "Fizz", múltiplos de 5 por "Buzz"e múltiplos de ambos por "FizzBuzz".

```
program FizzBuzz;
var
  i: integer;
begin
  for i := 1 to 20 do
  begin
    if (i mod 3 = 0) and (i mod 5 = 0) then
      writeln('FizzBuzz')
    else if i mod 3 = 0 then
      writeln('Fizz')
    else if i mod 5 = 0 then
      writeln('Buzz')
    else
```

```
      writeln(i);
  end;
end.
```

Esses exemplos demonstram a clareza da sintaxe do Pascal, com uso de estruturas de controle (`if`, `for`) e modularidade, um exemplo de maior complexidade seria por exemplo a criação de um CRUD, que se encontra disponivel como adicional ao fim desde relatorio.

7 Prática: Tutoriais de Instalação, Uso e Programação

7.1 Instalação do Free Pascal

O Free Pascal é um compilador moderno e gratuito para a linguagem Pascal, disponível em diversas plataformas (Windows, Linux, macOS). Siga os passos abaixo para instalá-lo:

1. **Baixe o Free Pascal:** Acesse o site oficial `https://www.freepascal.org` e faça o download da versão mais recente para seu sistema operacional.
2. **Instalação no Windows:**
   - Execute o instalador (`.exe`) e siga as instruções.
   - O compilador será instalado por padrão em `C:`.
3. **Instalação no Linux:**
   - Use o gerenciador de pacotes (ex.: `sudo apt install fpc` no Ubuntu).
   - Verifique a instalação com `fpc -version`.
4. **Ambiente de Desenvolvimento:** O Lazarus (`https://www.lazarus-ide.org`) é uma IDE recomendada que integra o Free Pascal e suporta desenvolvimento gráfico.
5. **Opção online:** Para familiarização com a linguagem e somente testes simples, pode-se utilizar um compilador online, para este estudo, foi utilizado o `https://www.onlinegdb.com/online_pascal_compiler`

7.2 Compilando e Executando um Programa

1. Crie um arquivo `programa.pas` com o código desejado (ex.: o programa FizzBuzz acima).
2. Abra o terminal, navegue até o diretório do arquivo e compile com: `fpc programa.pas`.
3. Execute o programa gerado: `./programa` (Linux/macOS) ou `programa.exe` (Windows).

7.3 Tutorial Prático: Criando um Programa Simples

Crie um programa que verifica se um número é par ou ímpar:

1. Crie um arquivo `par_impar.pas`.
2. Adicione o seguinte código:

```
program ParImpar;
var
  numero: integer;
begin
  writeln('Digite um número:');
  readln(numero);
  if numero mod 2 = 0 then
    writeln('O número é par')
  else
    writeln('O número é ímpar');
end.
```

3. Compile e execute conforme descrito acima.

8 Considerações Finais

O estudo da linguagem Pascal revelou sua importância histórica e pedagógica na ciência da computação. Projetada por Niklaus Wirth com foco no ensino, Pascal introduziu conceitos de programação estruturada e tipagem forte que moldaram gerações de programadores e influenciaram linguagens modernas como Modula-2, Oberon e Delphi. Sua sintaxe clara e organização lógica a tornam uma ferramenta valiosa para iniciantes, enquanto sua robustez permitiu aplicações práticas, especialmente com o Turbo Pascal.

Embora tenha perdido espaço no mercado para linguagens mais versáteis como C++ e Python, o legado de Pascal permanece vivo em compiladores como o Free Pascal e em sua influência no ensino de programação, mesmo que limitando-se atualmente ao ambiente academico como sua primaria função.

Referências

EVARISTO, Jaime. *Programando com Pascal: a linguagem do Turbo Pascal e do Delphi*. 2. ed. revista. São Paulo: E-Book Express, 2000.

JENSEN, Kathleen; WIRTH, Niklaus. *Pascal user manual and report*. 4. ed. Revisão de Andrew B. Mickel; James F. Miner. Berlin: Springer-Verlag, 1974.

FREE PASCAL. Disponível em: <https://www.freepascal.org>. Acesso em: 10 set. 2025.

A Manifestação Individual

A.0.1 Edgard de Paiva Melo Filho

Foi com consideravel surpresa que durante a pesquisa foi visto que as primeiras versões de programas relativamente recentes em comparação com a idade da linguagem de programação, foram feitos utilizando variações de Pascal, por exemplo o Skype, onde o cliente desktop inicial foi rapidamente prototipado usando Object Pascal no ambiente de desenvolvimento Delphi. Essa escolha foi impulsionada pela experiência do desenvolvedor principal de interface, que aproveitou as vantagens do Delphi para criar aplicativos Windows nativos eficientes, com uma experiência de usuário refinada que era essencial para a pequena equipe buscando entrar rapidamente no mercado. Além disso,a linguagem Pascal mostra uma abordagem bastante didática. A sua tipagem forte e a sintaxe clara facilitam a compreensão de conceitos fundamentais, como estruturas de controle e modularidade, que são essenciais para iniciantes. O uso de `begin` e `end` para delimitar blocos de código reforça a organização lógica, tornando assim bastante claro e intuitivo sua utilização.

A.0.2 Maicon Gomes Messias

Sinceramente, eu vejo o Pascal como uma linguagem que marcou época. Ela pode parecer "antiga"hoje em dia, mas teve um papel muito importante: ajudou muita gente a aprender lógica de programação de forma organizada. O legal do Pascal é que ele é bem estruturado, então não dá pra sair escrevendo código bagunçado; ele praticamente "obriga"a programar com disciplina.

B Exemplo 3: CRUD simples // Edgard Melo

```
program CRUD_Estudantes;
uses crt;

const
  MAX_ESTUDANTES = 100; { Limite do array }

type
  TEstudante = record
    id: integer;
    nome: string[50];
    nota: integer;
    ativo: boolean; { Flag para indicar se o registro está ativo }
    {notem que a linguagem deixa muito bem definida os tipos das variaveis}
  end;

var
  estudantes: array[1..MAX_ESTUDANTES] of TEstudante;
  totalEstudantes: integer;

{ Procedimento para inicializar o array }
procedure Inicializar;
var
  i: integer;
begin
  totalEstudantes := 0;
  for i := 1 to MAX_ESTUDANTES do
    estudantes[i].ativo := false;
end;

{ Função para encontrar um slot livre }
function EncontrarSlotLivre: integer;
var
  i: integer;
begin
  for i := 1 to MAX_ESTUDANTES do
    if not estudantes[i].ativo then
```

```
    begin
      EncontrarSlotLivre := i;
      exit;
    end;
  EncontrarSlotLivre := 0; { Retorna 0 se não houver slot livre }
end;

{ Procedimento para criar um novo estudante }
procedure CriarEstudante;
var
  inputId, inputNota: integer;
  inputNome: string[50];
  slot: integer;
begin
  clrscr; {comando para limpar o console/janela atual}
  writeln('=== Criar Estudante ===');
  slot := EncontrarSlotLivre;
  if slot = 0 then
  begin
    writeln('Erro: Banco cheio!');
    readln;
    exit;
  end;
  writeln('Digite o ID: ');
  readln(inputId);
  writeln('Digite o nome: ');
  readln(inputNome);
  writeln('Digite a nota: ');
  readln(inputNota);
  with estudantes[slot] do
  begin
    id := inputId;
    nome := inputNome;
    nota := inputNota;
    ativo := true;
  end;
  totalEstudantes := totalEstudantes + 1;
  writeln('Estudante criado com sucesso!');
  readln;
```

```
end;

{ Procedimento para ler (listar) todos os estudantes }
procedure ListarEstudantes;
var
  i: integer;
begin
  clrscr; {comando para limpar o console/janela atual}
  writeln('=== Lista de Estudantes ===');
  if totalEstudantes = 0 then
  begin
    writeln('Nenhum estudante cadastrado.');
    readln;
    exit;
  end;
  for i := 1 to MAX_ESTUDANTES do
    if estudantes[i].ativo then
      writeln('ID: ', estudantes[i].id, ' | Nome: ', estudantes[i].nome, ' | Nota: ',
      {notem que não posso fazer uma 'quebra de linha nessa definição ou causa um erro
  readln;
end;

{ Função para buscar estudante por ID }
function BuscarEstudante(id: integer): integer;
var
  i: integer;
begin
  for i := 1 to MAX_ESTUDANTES do
    if (estudantes[i].ativo) and (estudantes[i].id = id) then
    begin
      BuscarEstudante := i;
      exit;
    end;
  BuscarEstudante := 0; { Retorna 0 se não encontrado }
end;

{ Procedimento para atualizar um estudante }
procedure AtualizarEstudante;
var
```

```
  inputId, inputNota: integer;
  inputNome: string[50];
  indice: integer;
begin
  clrscr; {comando para limpar o console/janela atual}
  writeln('=== Atualizar Estudante ===');
  writeln('Digite o ID do estudante: ');
  readln(inputId);
  indice := BuscarEstudante(inputId);
  if indice = 0 then
  begin
    writeln('Estudante não encontrado!');
    readln;
    exit;
  end;
  writeln('Digite o novo nome: ');
  readln(inputNome);
  writeln('Digite a nova nota: ');
  readln(inputNota);
  with estudantes[indice] do
  begin
    nome := inputNome;
    nota := inputNota;
  end;
  writeln('Estudante atualizado com sucesso!');
  readln;
end;

{ Procedimento para deletar um estudante }
procedure DeletarEstudante;
var
  id, indice: integer;
begin
  clrscr; {comando para limpar o console/janela atual}
  writeln('=== Deletar Estudante ===');
  writeln('Digite o ID do estudante: ');
  readln(id);
  indice := BuscarEstudante(id);
  if indice = 0 then
```

```
  begin
    writeln('Estudante não encontrado!');
    readln;
    exit;
  end;
  estudantes[indice].ativo := false;
  totalEstudantes := totalEstudantes - 1;
  writeln('Estudante deletado com sucesso!');
  readln;
end;

{ Procedimento para exibir o menu }
procedure ExibirMenu;
begin
  clrscr; {comando para limpar o console/janela atual}
  writeln('=== Sistema CRUD de Estudantes ===');
  writeln('1. Criar estudante');
  writeln('2. Listar estudantes');
  writeln('3. Atualizar estudante');
  writeln('4. Deletar estudante');
  writeln('5. Sair');
  writeln('Escolha uma opção: ');
end;

{ Programa principal }
{Como o professor explicou durante a apresentação, o programa em si, se resume a esta
{sendo tudo acima,modulos/blocos para o funcionamento do programa, e como dito a modul
var
  opcao: integer;
begin
  Inicializar;
  repeat
    ExibirMenu;
    readln(opcao);
    case opcao of
      1: CriarEstudante;
      2: ListarEstudantes;
      3: AtualizarEstudante;
      4: DeletarEstudante;
```

```
      5: writeln('Saindo...');
      else
      begin
        writeln('Opção inválida! Pressione Enter para continuar.');
        readln;
      end;
    end;
  until opcao = 5;
end.
```

C Exemplo 4: Sistema de notas de alunos//Maicon Gomes Messias

```
program NotasAlunos;
    uses crt;
    var
      nome: string;
      nota1, nota2, nota3, media: real;

    begin
      clrscr;

      writeln('Digite o nome do aluno:');
      readln(nome);

      writeln('Digite a primeira nota:');
      readln(nota1);

      writeln('Digite a segunda nota:');
      readln(nota2);

      writeln('Digite a terceira nota:');
      readln(nota3);

      media := (nota1 + nota2 + nota3) / 3;

      writeln('Aluno: ', nome);
      writeln('Media: ', media:0:2);

      if media >= 7 then
        writeln('Situacao: Aprovado')
      else if media >= 5 then
        writeln('Situacao: Recuperacao')
      else
        writeln('Situacao: Reprovado');

      readln;
end.
```