Paradigmas de Linguagem de Programação

Pascal

Sistema bancário

Criação e Contexto Tecnológico

O Pascal foi criado por Niklaus Wirth e lançado em 1970, em um cenário onde linguagens de baixo nível, como Assembly, eram complexas e difíceis de usar, e linguagens de alto nível, como Fortran e COBOL, atendiam a nichos específicos, mas não eram adequadas para ensino.

Nesse contexto, o Pascal surgiu como uma alternativa estruturada, clara e eficaz para fins educacionais, oferecendo simplicidade e clareza para o aprendizado de programação.

Motivação e Objetivo Principal

A criação do Pascal foi motivada pelo desejo de ensinar boas práticas de programação estruturada e tornar os conceitos algorítmicos acessíveis, especialmente para iniciantes. A linguagem foi projetada para resolver problemas de legibilidade, redigibilidade e simplicidade, comuns nas ferramentas disponíveis na época.

Seu principal objetivo era introduzir a programação estruturada, com foco em sub-rotinas bem definidas, controle de fluxo previsível e padronização de conceitos como loops e condicionais.

Influências e Decisões-Chave

O Pascal foi fortemente influenciado pelo Algol 60, do qual herdou a clareza sintática, mas trouxe melhorias práticas.

Entre as decisões-chave, destaca-se a rejeição ao uso do comando Goto, substituído por estruturas como if-then-else, while e for, promovendo um código mais organizado e livre de "spaghetti code". Além disso, a adoção de tipagem estática e declarações explícitas reduziu erros e incentivou um planejamento estruturado e disciplinado.

Características

- Suas principais características incluem **modularidade**, por meio do uso de funções e procedimentos;
- **clareza**, garantindo que o código fosse **legível** e **organizado**;
- e um enfoque em ensinar **disciplina** e **planejamento** aos programadores.

Impacto e Evolução

O impacto do Pascal foi significativo, especialmente na educação, sendo amplamente adotado em universidades para ensinar programação estruturada.

O sucesso do Pascal levou ao desenvolvimento de versões como o Turbo Pascal, que expandiram seu uso para o mercado, e sua evolução posterior resultou no Object Pascal e no Delphi, ainda relevantes em nichos de desenvolvimento.

Compilação e Uso

- Pascal é uma linguagem **compilada**, gerando executáveis a partir do código-fonte.
- Compiladores populares: **Free Pascal** e **Turbo Pascal**.
- Excelente para ensino de programação estruturada e lógica computacional.

Instalação do Free Pascal

- A linguagem Pascal está disponível para sistemas operacionais como Windows, Linux, macOS e outros, graças a implementações como o Free Pascal.
- Este compilador é amplamente utilizado devido à sua compatibilidade com diversos sistemas e dispositivos. Para começar a usar o Free Pascal, basta acessar o site oficial e selecionar o sistema operacional desejado.
- O processo de instalação é simples: basta baixar o executável correspondente ao seu sistema e seguir as instruções fornecidas na plataforma.
- Você pode encontrar mais informações e o download do Free Pascal no link oficial: **Free Pascal - Download**.

Tradução para Linguagem de Máquina

1) **Análise Léxica**
O código Pascal é analisado para identificar tokens, como palavras-chave (ex: program, begin), operadores e identificadores.

2) Análise Sintática
Os tokens são organizados em uma árvore sintática, onde a estrutura do código é validada conforme a gramática da linguagem Pascal.

3) Análise Semântica
Verificação da consistência semântica, como tipos de dados e escopos de variáveis.

4) Geração de Código Intermediário
O código Pascal é transformado em uma forma intermediária, independente da arquitetura de hardware, para facilitar a otimização.

5) Otimização
O código intermediário é otimizado para melhorar o desempenho ou reduzir o tamanho do executável.

6) Geração de Código de Máquina
O código intermediário é traduzido para o código de máquina específico da arquitetura do sistema, criando o executável final.

Palavras reservadas ou pré-definidas

A linguagem Pascal possui um conjunto de palavras reservadas (ou pré-definidas).

Essas palavras possuem significados especiais para o compilador e não podem ser usadas como identificadores, ou seja, não podem ser utilizadas para nomear variáveis, funções ou procedimentos. Elas têm funções específicas na linguagem e são parte da sintaxe.

- **program:** Indica o início de um programa Pascal.
- **begin e end:** Delimitam blocos de código.
- **if, then, else**: Estruturas condicionais.
- **for, to, do:** Loops com controle de índice.
- **while, do**: Loops condicionais.
- **repeat, until:** Loops com verificação no final.
- **var**: Declaração de variáveis.
- **type**: Definição de tipos de dados.
- **function, procedure**: Para definição de funções e procedimentos.

Dados primitivos suportados

Inteiro: Dados numéricos positivos ou negativos, excluindo-se qualquer número fracionário.

Tipo de Dado	Faixa de Abrangência	Tamanho (bytes)
shortint	de –128 até 127	1 byte
integer	de –32.768 até 32.767	2 bytes
longint	de –2.147.483.648 até 2.147.483.647	4 bytes
byte	de 0 até 255	1 byte
word	de 0 até 65.535	2 bytes

Real: O tipo de dado real permite trabalhar com números fracionários, tanto positivos como negativos.

Tipo de Dado	Faixa de Abrangência	Tamanho (bytes)
real	de 2.9 e-39 até 1.7 e38	6 bytes
single	de 1.5 e-45 até 3.4 e38	4 bytes
double	de 5.0 e-324 até 1.7 e308	8 bytes
extended	de 3.4 e-4.932 até 1.1 e4.932	10 bytes
comp	de –9.2 e18 até 9.2 e18	8 bytes

Dados primitivos suportados

Caracteres: São considerados tipos caracteres, as sequências contendo letras, números e símbolos especiais. Uma sequência de caracteres, em Pascal, deve ser representada entre apóstrofos (‘’). Este tipo de dado é referenciado pelo identificador string, podendo armazenar de 1 até 255 caracteres. Podemos ainda identificar um tamanho menor do que 255 caracteres permitidos.

Lógicos: São caracterizados tipos lógicos, os dados com valores True (verdadeiro) e false (Falso). Este tipo de dado também é conhecido como tipo boleano. Representado pelo identificador boolean.

Acesso e manipulação de endereços de memória das variáveis

A linguagem Pascal permite o acesso e manipulação de endereços de memória das variáveis, embora seja uma linguagem de alto nível. Isso é possível através do uso de ponteiros e alocação dinâmica de memória, que são recursos importantes para trabalhar com endereços de memória diretamente.

Em Pascal, os ponteiros são variáveis que armazenam o endereço de memória de outra variável. Eles são definidos usando o símbolo ^ após o tipo base. Quando uma variável de ponteiro é criada, ela pode apontar para dados desse tipo específico. O acesso ao conteúdo apontado pelo ponteiro é feito usando o operador ^.

Acesso e manipulação de endereços de memória das variáveis

```
1 var
2    p1, p2: ^integer;
3
4 p1^ := 234;
5
6 p1 := @variavel;
7
```

Estrutura de um Programa Pascal

- Todo programa começa com **program** seguido pelo nome do programa.
- Seções de declaração (variáveis e tipos) antecedem o bloco principal.

```
1 program HelloWorld;
2 begin
3   writeln('Hello, world!');
4 end.
5
```

Declaração e Tipos de Variáveis

- Pascal usa tipagem estática: variáveis devem ser declaradas antes do uso.
- Sintaxe: **var nome_variavel: tipo**;

```
var
  idade: integer;
  nome: string;

```

Tipos comuns:

- Integer: Números inteiros.
- Real: Números com ponto flutuante.
- Char: Caracteres individuais.
- String: Sequências de caracteres.
- Boolean: Verdadeiro ou falso.

Estruturas de Controle

Laços

```
for i := 1 to 10 do
  writeln(i);

while idade < 18 do
begin
  writeln('Ainda menor de idade');
  idade := idade + 1;
end;

repeat
  writeln('Idade atual: ', idade);
  idade := idade + 1;
until idade = 18;
```

Condicional

```
if idade >= 18 then
  writeln('Maior de idade')
else
  writeln('Menor de idade');

```

Funções e Procedimentos

- **Procedimento:** Executa uma tarefa, mas não retorna valores.

```
1 procedure ExibirMensagem;
2 begin
3   writeln('Este é um procedimento');
4 end;
5
```

- **Função:** Retorna um valor.

```
1 function Somar(a, b: integer): integer;
2 begin
3   Somar := a + b;
4 end;
5
6 var
7   resultado: integer;
8 begin
9   resultado := Somar(3, 5);
10  writeln('Resultado: ', resultado);
11 end;
12
```

Estruturas de Dados

- **Arrays:** Coleções de elementos do mesmo tipo.

```
1 var
2   notas: array[1..5] of integer;
3
```

- **Registros:** Agrupam dados de tipos diferentes.

```
1 type
2   Pessoa = record
3     nome: string;
4     idade: integer;
5   end;
6 var
7   aluno: Pessoa;
8
```

Thank You

GitHub www.github.com