Pascal como você nunca viu?

Um projeto de implementação de um subconjunto da linguagem Pascal com geração de código para C moderno.

Murilo Henrique Alves

Universidade Tecnológica Federal do Paraná

31 de dezembro de 1979

Outline

O que é o Projeto? (Visão Geral)

Este trabalho consiste na construção de um compilador da linguagem Pascal para código C, resolvi fazer isso porque o pascal ISO 7185[1] é de 1983 e os compiladores modernos de C (gcc/clang) podem adicionar anos luz de evolução pra essa linguagem.

[1] International Organization for Standardization, *ISO 7185:1990 — Programming Languages — Pascal*.

O que foi usado?

Para obter êxito nesse projeto as ferramentas **Flex**[2] foi usada para análise léxica e geração dos tokens da linguagem, já o **Bison**[3] foi usado para análise sintática de acordo com a BNF (Backus-Naur Form) da linguagem[4].
Ferramentas como **Make** e os compiladores modernos **GCC** e **Clang** também foram utilizados.

[2] Westes e Project, *Flex - Fast Lexical Analyzer*.
[3] Foundation, *GNU Bison - The Yacc-compatible Parser Generator*.
[4] Alves, *BNF Grammar for Pascal Subset — caes101-languages-and-compilers*.

As particularidades da construção

O compilador foi desenhado de forma modular, seguindo fases bem clássicas da compilação, o que facilita muito a manutenção e a extensibilidade:

- **Análise léxica**: o scanner, implementado com Flex, lê o código fonte **Pascal(m)** do **stdin** ou de um arquivo de input fornecido para o compilador com a tag **-f** e o converte em um fluxo de tokens (palavra-chaves, identificadores, operadores, literais, etc.)
- **Análise sintática**: o parser, implementado com Bison, e linkado com as saídas do flex, lê os tokens e valida se a estrutura do programa está de acordo com a BNF da linguagem. Nessa fase uma árvore sintática abstrata **AST** foi construída para apoiar as próximas etapas.

- **Análise semântica**: esta é se uma das fases, se não a mais, mais críticas do projeto. O analisador percorre a AST para verificar a ordem semântica do código, suas particularidades são:
  - **Tabela de símbolos com escopos**: utiliza uma tabela de símbolos implementada com hash tables, que gerencia escopos para variáveis globais, locais e parâmetros de funções.
  - **Validação de tipos**: verifica a compatbilidade dos tipos em atribuições e expressões.
- **Geração de código**: apos a validação semântica, a AST é percorrida por uma última vez para traduzir as estruturas do Pascal(m) em código C equivalente.

O que o compilador suporta?

O compilador implementa um subconjunto significativo de Pascal, e adiciona algumas coisas à mais, o foco é garantir a expressividade da linguagem.

- **Tipos de dados abrangentes**:
  - Suporte à tipos primitivos: `integer`, `real`, `char`, `boolean`
  - Suporte à tipos estruturados, incluindo `ARRAY`, `SET` e `RECORD`, com suporte incluse à **parte variante dos registros**, uma característica poderosa do Pascal.
- **Estruturas de controle completas**:
  - Condicionais `IF / THEN / ELSE` e de múltipla escola `CASE`.
  - Laços de repetição `FOR`, `WHILE` e `REPEAT UNTIL`.

- **Sub-rotinas avançadas**:
  - Implementação completa de `FUNCTION` e `PROCEDURE`.
  - Suporte a **passagem de parâmetros por valor** e **por referência (VAR)**, traduzida de forma segura para o C usando ponteiros (`&` na chamada, `*` na declaração).

Inovações e particularidades refletidas na gramática (BNF)

A robustez deste compilador começa na definição de sua gramática formal, a qual foi definida para suportar construções poderosas do Pascal. Os exemplos de regras da BNF abaixo, demonstram algumas das funcionalidades-chave.

- **Passagem de parâmetros por referência (VAR)** A capacidade de modificar variáveis externamente de forma segura é definida diretamente na seção de parâmetros formais. A gramática permite explicitamente o uso da palavra-chave VAR.

BNF Snippet:

Passagem de Parâmetros por Referência (VAR)

```
formal_parameter_section ::= identifier_list ':'
    type_identifier
                           | 'VAR' identifier_list ':'
                             type_identifier
                           | 'FUNCTION' identifier_list
                             parameters ':'
                             type_identifier
                           | 'PROCEDURE' identifier_list
                             parameters
```

- **Records com Parte Variante (Estruturas de Dados Flexíveis)** Uma das funcionalidades mais avançadas do Pascal é o `record` com partes variantes, que permite que uma mesma variável armazene diferentes estruturas de dados de forma eficiente.

BNF Snippet:

Records com Parte Variante

```
field_list ::= fixed_part [ ';' variant_part ] |
    variant_part

variant_part ::= 'CASE' tag_field 'OF' variant_list
```

Destaque

Esta estrutura gramatical permite a criação de `records` que se comportam de maneira similar a uma **union** em C, mas com a segurança adicional de um campo tag, tornando o código mais robusto e com melhor aproveitamento de memória.

- **Construção de Conjuntos (SET)** A sintaxe para a criação de conjuntos `sets` em Pascal é declarativa e de alto nível. A gramática define uma forma clara de construir conjuntos, que o compilador traduz para operações bitwise eficientes em C.

BNF Snippet:

Construção de Conjuntos (SET)

```
primary_expression ::= '[' element_list ']' | ...

element_list ::= element { ',' element }

element ::= expression [ '..' expression ]
```

Destaque

A sintaxe `[ elemento, outro_elemento, faixa_inicio .. faixa_fim ]` é uma inovação do Pascal que simplifica a manipulação de coleções. Este compilador implementa essa abstração, gerando código C performático para representar e operar sobre esses conjuntos.

Referências I


Alves, Murilo. *BNF Grammar for Pascal Subset — caes101-languages-and-compilers*. https://github.com/omurilo/caes101-languages-and-compilers/blob/main/final_work/grammar/BNF. Accessed on June 22, 2025. 2025.


Foundation, Free Software. *GNU Bison - The Yacc-compatible Parser Generator*. https://www.gnu.org/software/bison/. Accessed on June 22, 2025. 2025.


International Organization for Standardization. *ISO 7185:1990 — Programming Languages — Pascal*. Rel. técn. Available at https://www.cs.bilkent.edu.tr/~guvenir/courses/CS315/iso7185pascal.pdf. Accessed on June 22, 2025. Geneva: ISO, 1990.

Referências II

 Westes, Vern Paxson e The Flex Project. *Flex - Fast Lexical Analyzer*. https://github.com/westes/flex/blob/master/README.md. Accessed on June 22, 2025. 2025.

The End