TRABAJO PRÁCTICO Nº1 SSL 2024

K2055

Grupo 14

Integrantes:

- Lucas Martin
- Tobias Viale
- Facundo Gallardo
- Tomas Ruggiero

Profesora:

- Roxana Leituz

09/08/2024

Pascal y Prolog.

1)

PASCAL:

Historia:

Pascal es un lenguaje de alto nivel de uso general que fue desarrollado originalmente por Niklaus Wirth a principios de la década de 1970. Fue desarrollado para enseñar programación como disciplina sistemática y para desarrollar programas confiables y eficientes.

Pascal es un lenguaje basado en Algol e incluye muchas construcciones de Algol. Algol 60 es un subconjunto de Pascal. Pascal ofrece varios tipos de datos y estructuras de programación. Es fácil de entender y mantener los programas de Pascal.

Pascal ha ganado popularidad en el ámbito docente y académico por varias razones:

- Fácil de aprender.
- Lenguaje estructurado.
- Produce programas transparentes, eficientes y confiables.
- Se puede compilar en una variedad de plataformas informáticas.

Características del lenguaje Pascal

Pascal tiene las siguientes características:

- Pascal es un lenguaje fuertemente tipado.
- Ofrece una amplia comprobación de errores.
- Ofrece varios tipos de datos como matrices, registros, archivos y conjuntos.
- Ofrece una variedad de estructuras de programación.
- Soporta programación estructurada a través de funciones y procedimientos.
- Es compatible con la programación orientada a objetos.
- Pascal se basa en el estilo estructurado por bloques del lenguaje de programación Algol.

¿Por qué utilizar Pascal?

Pascal permite a los programadores definir tipos de datos estructurados complejos y construir estructuras de datos dinámicas y recursivas, como listas, árboles y gráficos. Pascal ofrece características como registros, enumeraciones, subrangos, variables asignadas dinámicamente con punteros y conjuntos asociados.

Pascal permite definiciones de procedimientos anidados a cualquier nivel de profundidad. Esto realmente proporciona un excelente entorno de programación para aprender a programar como una disciplina sistemática basada en los conceptos fundamentales.

Entre las implementaciones más sorprendentes de Pascal se encuentran:

- Skype
- Comandante total
- TeX
- Macromedia Captivate
- Manzana lisa
- Varios juegos de PC
- Sistemas embebidos

Tokens Básicos

- <token> ::= <palabra reservada> | <identificador> | <constante> | <literal de cadena> | <punctuator> | <operador> | <delimitador>

Palabras Reservadas

- <palabra reservada> ::= program | const | type | var | procedure | function | label | goto | if | then | else | case | of | while | do | repeat | until | for | to | downto | with | begin | end | and | or | not | div | mod

Identificadores

- <identificador> ::= <letra> {<letra o dígito>}
    - <letra> ::= a | b | c | ... | z | A | B | ... | Z | _
    - <letra o dígito> ::= <letra> | <dígito>
    - <dígito> ::= 0 | 1 | ... | 9

Constantes

- <constante> ::= <número entero> | <número real> | <cadena> | <identificador constante> | <número entero> .. <número entero> | <identificador constante> .. <identificador constante> | <cadena>
    - <número entero> ::= <dígito> {<dígito>}
    - <número real> ::= <número entero> . <número entero> | <número entero> . <número entero> E <factor de escala> | <número entero> E <factor de escala>
    - <factor de escala> ::= <número entero> | <signo> <número entero>
    - <signo> ::= + | -
    - <identificador constante> ::= <identificador>
    - <cadena> ::= '<carácter> {<carácter>}'

Punctuadores y Delimitadores

- <punctuator> ::= . | , | ; | [|] | : | (|) | := | ..
- <operador> ::= + | - | * | / | div | mod | and | or | not | = | <> | < | <= | >= | > | in
- <delimitador> ::= begin | end | if | then | else | case | of | while | do | repeat | until | for | to | downto | with | goto

SORTING (Ordenamiento):

En Pascal, no existe una función de ordenamiento directa, llamada “sort”, como por ejemplo en Python.

Sin embargo, existen ciertos algoritmos que cumplen la misma función.

1) **Ordenamiento Por Inserción (Insertion Sort):**

Este algoritmo es sencillo pero no muy eficiente para grandes conjuntos de datos.

Funciona comparando cada elemento con los anteriores y moviendo el elemento actual a su posición correcta.

```pascal
program OrdenamientoPorInsercion;

var
  arr: array[1..10] of integer;
  i, j, key: integer;

begin
  // Inicializar el arreglo con números aleatorios
  Randomize;
  for i := 1 to 10 do
    arr[i] := Random(100);

  // Imprimir el arreglo antes de ordenarlo
  writeln('Arreglo antes de ordenarlo:');
  for i := 1 to 10 do
    write(arr[i], ' ');
  writeln;

  // Algoritmo de ordenamiento por inserción
  for i := 2 to 10 do
  begin
    key := arr[i];
    j := i - 1;

    // Mover los elementos del arr[1..i-1], que son mayores que la clave, a una posición adelante de su posición actual
    while (j > 0) and (arr[j] > key) do
    begin
      arr[j + 1] := arr[j];
      j := j - 1;
    end;

    arr[j + 1] := key;
  end;

  // Imprimir el arreglo después de ordenarlo
  writeln('Arreglo después de ordenarlo:');
  for i := 1 to 10 do
    write(arr[i], ' ');
  writeln;
end.
```

2) **Ordenamiento por Burbuja (Bubble Sort):**

Este algoritmo tambien es simple y sencillo, pero aún menos eficiente.

Este compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto.

Tambien, existen otros como por ejemplo el QuickSort o el MergeSort.

```
program OrdenamientoPorBurbuja;

var
  arr: array[1..10] of integer;
  i, j, temp: integer;

begin
  // Inicializar el arreglo con números aleatorios
  Randomize;
  for i := 1 to 10 do
    arr[i] := Random(100);

  // Imprimir el arreglo antes de ordenarlo
  writeln('Arreglo antes de ordenarlo:');
  for i := 1 to 10 do
    write(arr[i], ' ');
  writeln;

  // Algoritmo de ordenamiento de burbuja
  for i := 1 to 10 do
    for j := 1 to 10 - i do
      if arr[j] > arr[j + 1] then
      begin
        // Intercambiar arr[j] y arr[j + 1]
        temp := arr[j];
        arr[j] := arr[j + 1];
        arr[j + 1] := temp;
      end;

  // Imprimir el arreglo después de ordenarlo
  writeln('Arreglo después de ordenarlo:');
  for i := 1 to 10 do
    write(arr[i], ' ');
  writeln;
end.
```

2) PROLOG:

Historia:

Se trata de un lenguaje de programación ideado a principios de los años 70 en la Universidad de Aix-Marseille I (Marsella, Francia) por Alain Colmerauer y Philippe Roussel. Nació de un proyecto que no tenía como objetivo la traducción de un lenguaje de programación, sino el tratamiento algorítmico de lenguajes naturales. Esta primera versión de Prolog fue programada en ALGOL W y salió en 1972.

Inicialmente se trataba de un lenguaje totalmente interpretado hasta que, en 1983, David H.D. Warren desarrolló un compilador capaz de traducir Prolog en un conjunto

de instrucciones de una máquina abstracta denominada Warren Abstract Machine, o abreviadamente, WAM. Desde entonces Prolog es un lenguaje semi-interpretado.

Si bien en un principio se trataba de un lenguaje de uso reducido, la aparición de intérpretes del mismo para microordenadores de 8 bits (ej: micro-PROLOG) y para ordenadores domésticos de 16 bits (ej: Turbo Prolog de Borland, entre otros muchos), a lo largo de la década de 1980, contribuyó notablemente a su popularización. Otro importante factor en su difusión fue la adopción del mismo para el desarrollo del proyecto de la quinta generación de computadoras a principios de la década de los 80, en cuyo contexto se desarrolló la implementación paralelizada del lenguaje llamada KL1 y del que deriva parte del desarrollo moderno de Prolog.

Prolog se enmarca en el paradigma de los lenguajes lógicos y declarativos, lo que lo diferencia enormemente de otros lenguajes más populares tales como Fortran, Pascal, C o Java.

¿Por qué utilizar Prolog?

Prolog se utiliza principalmente en el campo de la inteligencia artificial, incluyendo áreas como el aprendizaje automático, el procesamiento del lenguaje natural y la construcción de sistemas expertos. Su enfoque basado en la lógica y las relaciones entre entidades lo convierte en una herramienta poderosa para resolver problemas complejos.

BNF:

<program> ::= <clause list> <query> | <query>
<clause list> ::= <clause> | <clause list> <clause>
<clause> ::= <predicate> . | <predicate> :- <predicate list>.
<predicate list> ::= <predicate> | <predicate list> , <predicate>
<predicate> ::= <atom> | <atom> (<term list>)
<term list> ::= <term> | <term list> , <term>
<term> ::= <numeral> | <atom> | <variable> | <structure>
<structure> ::= <atom> (<term list>)
<query> ::= ?- <predicate list>.
<atom> ::= <small atom> | ' <string> '
<small atom> ::= <lowercase letter> | <small atom> <character>
<variable> ::= <uppercase letter> | <variable> <character>
<lowercase letter> ::= a | b | c | ... | x | y | z
<uppercase letter> ::= A | B | C | ... | X | Y | Z | _
<numeral> ::= <digit> | <numeral> <digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<character> ::= <lowercase letter> | <uppercase letter> | <digit> | <special>

<special> ::= + | - | * | / | \ | ^ | ~ | : | . | ? | | # | $ | &
<string> ::= <character> | <string> <character>

Algoritmos de Ordenamiento (SORT):

En Prolog existen predicados preexistentes para el ordenamiento de listas. El sort/2, el más común de todos, se encarga de ordenar una lista de manera ascendente de manera simple. Por otro lado, existe la versión más completa, sort/4 que permite ordenar una lista de pares clave-valor, generando una lista ordenada basada en las claves. Este último también permite especificar si deseamos repetirlo o no.

Sintaxis:

Sort/2: ? sort(+List, -SortedList)
Sort/4: ? sort(+Mode, +KeyPos, +List, -SortedList)

Por otro lado, también podemos generar nuestros propios predicados de ordenamiento de listas. Veremos dos ejemplos de ordenamiento:

Ordenamiento por Inserción (Insertion Sort)

```
% Ordenamiento por Insercion
insertion_sort([], []).
insertion_sort([Head | Tail], Sorted) :-
    insertion_sort(Tail, SortedTail),
    insert(Head, SortedTail, Sorted).

insert(X, [], [X]).
insert(X, [Y | Rest], [X, Y | Rest]) :- X =< Y.
insert(X, [Y | Rest], [Y | RestSorted]) :- X > Y, insert(X, Rest, RestSorted).

% Main Program
main:-
        process,
        halt.

process:-
        nl,
        List = [4, 3, 2, 1], % LISTA DE EJEMPLO
        insertion_sort(List, Sorted),
        write('Sorted List: '),
        write(Sorted).

:- main.
```

Ordenamiento por Burbuja (Bubble Sort)

```
% Ordenamiento por Burbuja
bubble_sort(List,Sorted):-b_sort(List,[],Sorted).
b_sort([],Acc,Acc).
b_sort([H|T],Acc,Sorted):-bubble(H,T,NT,Max),b_sort(NT,[Max|Acc],Sorted).

bubble(X,[],[],X).
bubble(X,[Y|T],[Y|NT],Max):-X>Y,bubble(X,T,NT,Max).
bubble(X,[Y|T],[X|NT],Max):-X=<Y,bubble(Y,T,NT,Max).

main:-
        process,
        halt.

process:-
        write('Hello World'),
        nl,
        List = [4, 3, 2, 1],
        bubble_sort(List, Sorted),
        write('Sorted List: '),
        write(Sorted).

:- main.
```

Resumen

En este trabajo, hemos comparado dos lenguajes de programación: Pascal y Prolog.

Pascal es un lenguaje de programación de alto nivel desarrollado por Niklaus Wirth en la década de 1970. Fue diseñado para enseñar programación de manera sistemática y para desarrollar programas confiables y eficientes. Pascal es un lenguaje fuertemente tipado que ofrece una amplia comprobación de errores y soporta programación estructurada y orientada a objetos. En Pascal, no existe una función de ordenamiento directa, pero se pueden implementar algoritmos de ordenamiento como el de inserción y el de burbuja.

Por otro lado, Prolog es un lenguaje de programación lógico y declarativo desarrollado en la Universidad de Aix-Marseille I en la década de 1970. A diferencia de Pascal, Prolog es un lenguaje semi-interpretado. Prolog se utiliza principalmente en inteligencia artificial y lingüística computacional. En Prolog, existen predicados preexistentes para el ordenamiento de listas, como sort/2 y sort/4. También es posible implementar algoritmos de ordenamiento personalizados.

En resumen, ambos lenguajes tienen sus propias fortalezas y debilidades, y su elección depende del problema que se quiera resolver. Pascal es ideal para aprender programación de manera sistemática y desarrollar programas confiables y eficientes, mientras que Prolog es más adecuado para problemas que requieren un enfoque lógico y declarativo, como la inteligencia artificial y la lingüística computacional.

Bibliografia:

https://wiki.freepascal.org/sorting_algorithm

https://www.aprendeaprogramar.com/mod/forum/discuss.php?d=1509

https://www.nachocabanes.com/pascal/curso/cupasamp03.php

https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento

Tutorial completo sobre la creación de Pascal: historia y desarrollo - Triunfa Emprendiendo

Programación en Pascal/Historia - Wikilibros (wikibooks.org)

Pascal - Guía rápida (isolution.pro)

PROLOG: a brief history (mta.ca)

Prolog (lenguaje de programación) - EcuRed

Prolog - Wikipedia, la enciclopedia libre

El lenguaje Prolog: un ejemplo del paradigma de programación lógica - EducaciónIT (educacionit.com)

El lenguaje Prolog: un ejemplo del paradigma de programación lógica (genbeta.com)

Prolog ▷ Información, Historia, Biografía y más. (wikidat.com)