Estructura de Datos y de la Información

Pilas y expresiones aritméticas

LIDIA
Laboratorio de Investigación y desarrollo en Inteligencia Artificial

Departamento de Computación
Universidade da Coruña, España

Índice

1. **Introducción**
   - **Expresión aritmética**
   - **Notaciones infija, prefija y postfija**
2. **Algoritmos**
   - **Evaluación de una notación postfija**
   - **Conversión de infija a postfija**

Pilas

Notación Prefija, Infija y Postfija

- **Expresión aritmética:**
  - **Formada por operandos y operadores: A*B / (A+C)**
  - **Operandos: variables que toman valores enteros o reales**
  - **Operadores:**

Paréntesis	( )	Nivel mayor de prioridad
Potencia	^	↓
Multiplicación / División	* /	↓
Suma / Resta	+ -	Nivel menor de prioridad

  - **En caso de igualdad de prioridad**
    - **Son evaluados de izquierda a derecha (se evalúa primero el que primero aparece) ⇒ 5*4/2 = (5*4)/2 = 10**
    - **Cuando aparecen varios operadores de potenciación juntos la expresión se evalúa de derecha a izquierda ⇒ 2^3^2 = 2^(3^2) = 2^9 = 512**

- **Notación Infija**
  - **Es la notación ya vista que sitúa el operador entre sus operandos.**
  - **Ventaja: Es la forma natural de escribir expresiones aritméticas**
  - **Inconveniente: Muchas veces necesita de paréntesis para indicar el orden de evaluación: A*B/(A+C) ≠ A*B/A+C**

Pilas

Notación Prefija, Infija y Postfija

- **Notación Prefija o Polaca**
  - **En 1920 un matemático de origen polaco, Jan Lukasiewicz, desarrollo un sistema para especificar expresiones matemáticas sin paréntesis.**
  - **Esta notación se conoce como notación prefija o polaca (en honor a la nacionalidad de Lukasiewicz) y consiste en situar al operador ANTES que los operandos.**
  - **Ejemplo: la expresión infija A*B / (A+C) se representaría en notación prefija como: /*AB+AC**

- **Notación Postfija o Polaca Inversa**
  - **La notación postfija o polaca inversa es una variación de la notación prefija de forma que el operador de pone DESPUÉS de los operandos.**
  - **Ejemplo: la expresión infija A*B / (A+C) se representaría en notación postfija como: AB*AC+/**
  - **Ventajas:**
    - **La notación postfija (como la prefija) no necesita paréntesis.**
    - **La notación postfija es más utilizada por los computadores ya que permite una forma muy sencilla y eficiente de evaluar expresiones aritméticas (con pilas).**

Pilas

Evaluación de una Notación Postfija

- **Pseudocódigo**
  1. **Inicializar la pila**
  2. **Repetir hasta que no haya caracteres en la expresión a evaluar**
     - **2.1 Obtener el siguiente item de la expresión**
     - **2.2 Si el elemento es un operando se mete en la pila**
     - **2.3 Si el elemento es un operador (denominado &) entonces:**
       - **2.3.1 Se extraen los dos elementos superiores de la pila, denominados Op2 y Op1 respectivamente.**
       - **2.3.2 Se evalúa el resultado de Op1 & Op2 y se almacena en Z**
       - **2.3.3 Se introduce Z en la cima de la pila**
  3. **Obtener el valor de la expresión de la cima de la pila**

Pilas

Evaluación de una Notación Postfija

- **Ejemplo:**
  - **Expresión aritmética infija:** **A*B / (A+C)**
  - **Expresión aritmética postfija:** **AB*AC+/**
  - **Valores A=4, B=5 y C=6:** **45*46+/**

Cima

Carácter leído	Acción	Pila
4	Meter(4)	4
5	Meter(5)	5, 4
*	Op2=Sacar → 5 Op1=Sacar → 4 4 * 5 = 20 Meter(20)	20
4	Meter(4)	4, 20
6	Meter(6)	6, 4, 20
+	Op2=Sacar → 6 Op1=Sacar → 4 4 + 6 = 10 Meter(10)	10, 20
/	Op2=Sacar → 10 Op1=Sacar → 20 20 / 10 = 2 Meter(2)	2
Fin cadena	Fin evaluación	2

Pilas

Evaluación de una Notación Postfija

- **Código:**
  - **Precondición: La expresión postfija es correcta y consiste en un string donde cada carácter es o un operando o un operador.**
  - **Utilizamos una pila que almacena valores reales**

```
function EvaluaPostfija (ExpPostfija: string): real;
var
  P: tPila;
  Op1, Op2, Z: real;
  i: integer;
begin
  PilaVacia(P);
  for i:=1 to Length(ExpPostFija) do // Recorremos la expresion postfija
    begin
      // Si es un operando lo metemos en la pila
      if ExpPostfija[i] in ['0'..'9'] then
        Meter(P,StrToInt(ExpPostfija[i]))
      else begin           // Si es un operador sacamos los dos primeros elementos
         Cima(P,Op2);Sacar(P); // de la pila y realizamos la operación indicada con
         Cima(P,Op1);Sacar(P); // ellos. El resultado lo volvemos a meter en la pila
         Z:=Evalua(Op1, ExpPostfija[i], Op2);
         Meter(P,Z)
      end
    end;
  // Al final la pila queda con un único elemento que es el resultado de la evaluación
  Cima(P,EvaluaPostfija);
end;
```

Pilas

Evaluación de una Notación Postfija

- **Código:**
  - **Utilizamos la función evalúa para realizar la operación indicada por el símbolo del operador**

```
function Evalua(Op1:real; Operador:char; Op2:real):real;
begin
  Case Operador of
    '^': Evalua:=Power(Op1,Op2);
    '*': Evalua:=Op1*Op2;
    '/': Evalua:=Op1/Op2;
    '+': Evalua:=Op1+Op2;
    '-': Evalua:=Op1-Op2;
  end
end;
```

Pilas

Conversión de Infija a Postfija

- **Pseudocódigo**
  1. **Inicializar la pila**
  2. **Repetir hasta que no haya caracteres en la expresión de entrada**
     - **2.1 Leer un carácter de la expresión**
     - **2.2 Si es un operando se pasa a la expresión postfija de salida**
     - **2.3 Si el elemento es un operador distinto de ‘)’ entonces:**
       - **2.3.1 Si la pila está vacía se mete en la pila.**
       - **2.3.2 Si la pila NO está vacía**
         - **Si la prioridad del operador es mayor que la prioridad del operador de la cima de la pila ⇒ se mete en la pila**
         - **Si la prioridad del operador es menor o igual que la prioridad del operador de la cima de la pila ⇒ se saca el operador de la cima y se coloca en la expresión postfija. Volvemos a 2.3**
     - **2.4 Si el elemento es el operador ‘)’ entonces:**
       - **2.4.1 Se sacan operadores de la pila hasta encontrar el paréntesis ‘(‘ que se elimina (las expresiones postfijas no llevan paréntesis)**
  3. **Al finalizar el recorrido por la expresión aritmética se pasa todo el contenido de la pila a la expresión postfija**

Pilas

Conversión de Infija a Postfija

- **Pseudocódigo**
  - **Prioridad de los operadores**

Operador	Prioridad en la expresión infija	Prioridad en la pila
^	4	3
*	2	2
/	2	2
+	1	1
-	1	1
(	5	0
)	no definida	no definida

  - **Notas:**
    - **La prioridad de la potencia es menor en la pila que en la expresión infija para evaluar varios operadores de potenciación de derecha a izquierda (se evalúa primero lo último encontrado)**
    - **Esta variación no afecta a los otros operadores ya que la prioridad de la potencia siempre es mayor**
    - **El paréntesis izquierdo pasa a tener prioridad cero ya que sólo se extrae de la pila (para eliminarlo) cuando aparece un paréntesis derecho.**
    - **La prioridad del paréntesis derecho no está definida porque nunca entre a formar parte de las comparaciones**

Pilas

Conversión de Infija a Postfija

- **Ejemplo:**
  - **Expresión aritmética infija: A*B / (A+C)**
  - **Expresión aritmética postfija: AB*AC+/**

Cima

Carácter leído	Acción	Pila	Expresión postfija
A	Pasar a postfija		A
*	Meter('*')	*	A
B	Pasar a postfija	*	AB
/	PInfija('/') <= PPila('*') ⇒ Sacar→('*') y pasamos a postfija Como la pila está vacía ⇒ Meter ('/')	/	AB*
(	PInfija('(') > PPila('/') ⇒ Meter('(')	(, /	AB*
A	Pasar a postfija	(, /	AB*A
+	PInfija('+') > PPila('(') ⇒ Meter ('+')	+, (, /	AB*A
C	Pasar a postfija	+, (, /	AB*AC
)	Pasamos los elementos de la pila a postfija hasta encontrar '(' que se elimina	/	AB*AC+
Fin entrada	Pasamos todo lo que queda en la pila a la expresión postfija		AB*AC+/

Conversión de Infija a Postfija

```
function InfijaToPostfija(Infija:string):string;
uses TADPila; // Usamos una pila de carácteres
var
  Postfija: string;
  P: tPila;  aux: char;
  i: integer;
  salir:boolean;
begin
  Postfija:=''; PilaVacia(P);
  for i:=1 to Length(Infija) do begin // Recorremos la expresión infija
     if Infija[i] in ['0'..'9']       // Un operando se pasa a la expresión postfija
     then  Postfija:=Postfija + Infija[i]
     else      if Infija[i] in ['^','*','/','+','-','(']
               then begin // Si es un operador
                salir:=false;
                while not salir do begin
                   // Comparamos la prio del operador de la expresion con ultimo guardado
                   Cima(P, aux);
                   if EsPilaVacia(P) or
                     (PrioridadInfija(Infija[i])>PrioridadPila(aux)
                    then begin
                       Meter(P,Infija[i]); // se mete el operador en la pila
                        salir:=true;
                    end
                    else begin
                      // Si es < o = se saca el operador de la pila a la expresión
                      Cima(P, aux); Sacar(P);
                      Postfija:=Postfija + aux;
                    end
                end //while
               end
               else  if Infija[i]=')'       // Si encontramos un paréntesis dch se sacan
                     then repeat            // operadores de la pila hasta encontrar un
                         Cima(P, aux); Sacar(P);      // paréntesis izqdo que se elimina.
                         if (aux <> '(')
                         then Postfija:=Postfija + aux
                       until (aux='(');
   end; // Cuando se acaba la expresión infija se vacía la pila en la expresión postfija
  while not EsPilaVacia(P) do begin
       Cima(P, aux); Sacar(P);
       Postfija:=Postfija + aux;
  end;
  InfijaToPostfija:=Postfija;
end;
```

Pilas

Conversión de Infija a Postfija

- **Código:**
  - **Funciones que establecen la prioridad de los operadores**

```
function PrioridadInfija (c:char): integer;
begin
  Case c of
    '^': PrioridadInfija:=4;
    '*': PrioridadInfija:=2;
    '/': PrioridadInfija:=2;
    '+': PrioridadInfija:=1;
    '-': PrioridadInfija:=1;
    '(': PrioridadInfija:=5;
    // La prioridad para el paréntesis derecho no está definida
  end
end;
```

```
function PrioridadPila (c:char): integer;
begin
  Case c of
    '^': PrioridadPila:=3;
    '*': PrioridadPila:=2;
    '/': PrioridadPila:=2;
    '+': PrioridadPila:=1;
    '-': PrioridadPila:=1;
    '(': PrioridadPila:=0;
    // La prioridad para el paréntesis derecho no está definida
  end
end;
```

Pilas

Conversión de Infija a Postfija

- **Ejemplo: expresión infija: 4*(5+6-(8/2^3)-7)-1**

Cima

Carácter leído	Acción	Pila	Expresión postfija
4	Pasar a postfija		4
*	Meter('*')	*	4
(	PInfija('(') > PPila('*') ⇒ Meter('(')	(, *	4
5	Pasar a postfija	(, *	45
+	PInfija('+') > PPila('(') ⇒ Meter ('+')	+, (, *	45
6	Pasar a postfija	+, (, *	456
-	PInfija('-') <= PPila('+') ⇒ Sacar→('+') y pasamos a postfija PInfija('-') > PPila('(') ⇒ Meter ('-')	-, (, *	456+
(	PInfija('(') > PPila('-') ⇒ Meter('(')	(, -, (, *	456+
8	Pasar a postfija	(, -, (, *	456+8
/	PInfija('/') > PPila('(') ⇒ Meter ('/')	/, (, -, (, *	456+8
2	Pasar a postfija	/, (, -, (, *	456+82
^	PInfija('^') > PPila('/') ⇒ Meter ('^')	^, /, (, -, (, *	456+82
3	Pasar a postfija	^, /, (, -, (, *	456+823
)	Pasamos los elementos de la pila a postfija hasta encontrar '(' que se elimina	-, (, *	456+823^/
-	PInfija('-') <= PPila('-') ⇒ Sacar→('-') y pasamos a postfija PInfija('-') > PPila('(') ⇒ Meter ('-')	- , (, *	456+823^/-
7	Pasar a postfija	- , (, *	456+823^/-7
)	Pasamos los elementos de la pila a postfija hasta encontrar '(' que se elimina	*	456+823^/-7-
-	PInfija('-') <= PPila('*') ⇒ Sacar→('*') y pasamos a postfija Como la pila está vacía ⇒ Meter ('-')	-	456+823^/-7-*
1	Pasar a postfija	-	456+823^/-7-*1
Fin entrada	Pasamos todo lo que queda en la pila a la expresión postfija		456+823^/-7-*1-

Pilas

Evaluación de una Notación Postfija

- **Ejemplo:**
  - **Expresión aritmética postfija: 456+823^/-7-*1-**

Cima

Carácter leído	Acción	Pila
4	Meter(4)	4
5	Meter(5)	5, 4
6	Meter(6)	6, 5, 4
+	Op2=Sacar→6 Op1=Sacar→5 5 + 6 = 11 Meter(11)	11, 4
8	Meter(8)	8, 11, 4
2	Meter(2)	2, 8, 11, 4
3	Meter(3)	3, 2, 8, 11, 4
^	Op2=Sacar→3 Op1=Sacar→2 2 ^ 3 = 8 Meter(8)	8, 8, 11, 4
/	Op2=Sacar→8 Op1=Sacar→8 8 / 8 = 1 Meter (1)	1, 11, 4

Cima

Carácter leído	Acción	Pila
-	Op2=Sacar→1 Op1=Sacar→11 11 - 1 = 10 Meter(10)	10, 4
7	Meter(7)	7, 10, 4
-	Op2=Sacar→7 Op1=Sacar→10 10 - 7 = 3 Meter(3)	3, 4
*	Op2=Sacar→3 Op1=Sacar→4 4 * 3 = 12 Meter(12)	12
1	Meter(1)	1, 12
-	Op2=Sacar→1 Op1=Sacar→12 12 - 1 = 11 Meter(11)	11
Fin cadena	Fin evaluación	11