Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare

ARBORI BINARI PENTRU EXPRESII

Introducere

- Reprezentarea unei expresii (aritmetice, logice sau de alt tip) în compilatoare se poate face fie printr-un șir postfixat, fie printr-un arbore binar
- Arborele permite și optimizări la evaluarea expresiilor cu subexpresii comune
- Un șir postfixat este o altă reprezentare, liniară, a unui arbore binar

Introducere

- Reprezentarea expresiilor prin arbori rezolvă problema ordinii efectuării operațiilor, prin poziția operatorilor în arbore, fără a folosi paranteze sau priorități relative între operatori
- Operatorii sunt aplicați începând de la frunze către rădăcină, adică în ordine postfixată

Arbori binari pentru expresii

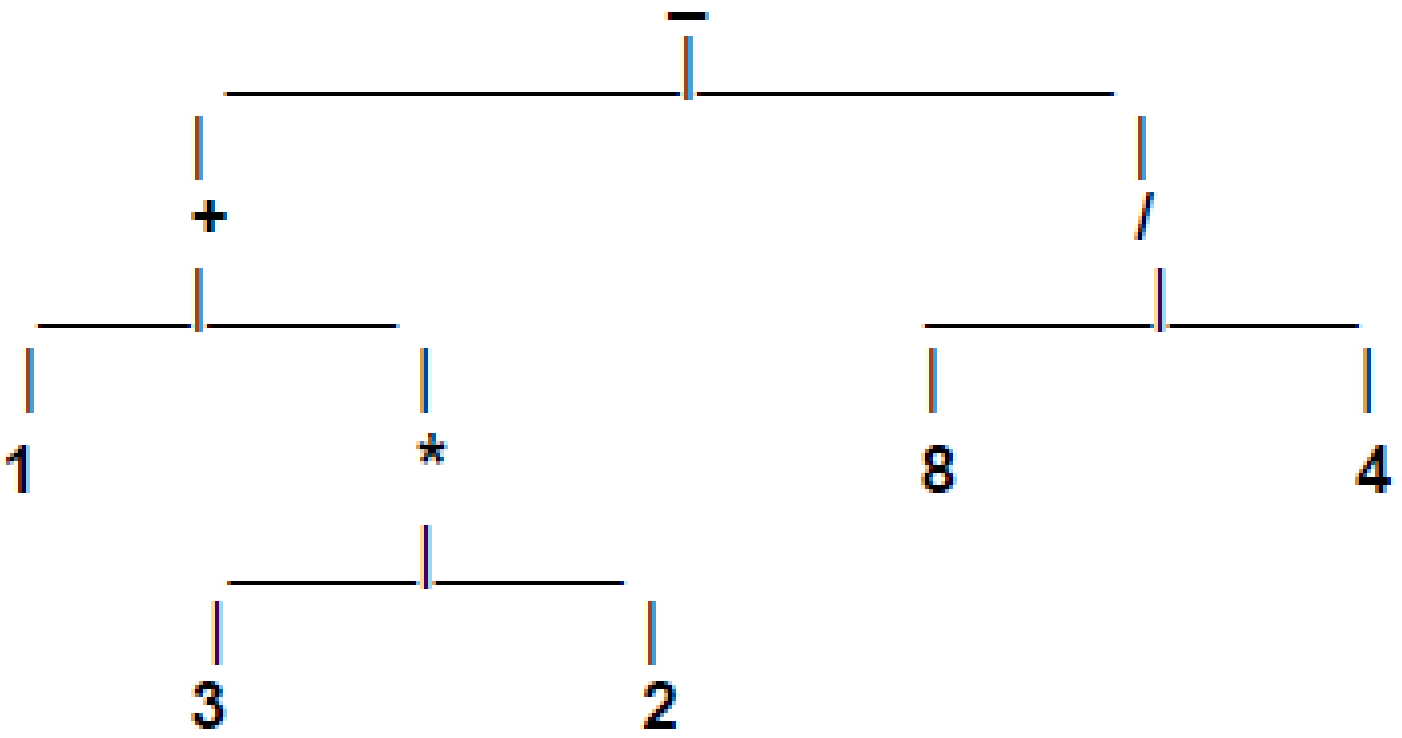
- Construcția arborelui este mai simplă dacă se pornește de la forma postfixată sau prefixată a expresiei, deoarece nu există problema priorității operatorilor și a parantezelor
- Construcția arborelui se face de la frunze spre rădăcină
- Un algoritm recursiv este mai potrivit dacă se pleacă de la șirul prefixat, iar un algoritm care folosește o stivă este mai potrivit dacă se pleacă de la șirul postfixat

Exemplu

- Se consideră expresii cu operanzi dintr-o singură cifră, cu operatorii aritmetici binari '+', '-', '*', '/' și fără spații între operanzi și operatori
- Eliminarea acestor restricții nu modifică problema sau soluția, dar complică implementarea ei

Exemplu

- Pentru expresia:
- 1 + 3 * 2 – 8 / 4 arborele echivalent este:



Observații

- Operanzii se află numai în noduri terminale, iar operatorii numai în noduri interne
- Evaluarea expresiei memorate într-un arbore binar este un caz particular de parcurgere postfixată a nodurilor arborelui și se poate face fie recursiv, fie folosind o stivă de pointeri la noduri
- Nodurile sunt interpretate diferit (operanzi sau operatori), fie după conținutul lor, fie după poziția lor în arbore (terminale sau neterminale)

Evaluarea recursivă a unui arbore

```
int eval (tnod * r) {
  int vst, vdr ;                  // valoare din subarbore stanga si dreapta
  if (r == NULL)
    return 0;
  if ( isdigit(r→val))             // daca este o cifra
      return r→val -'0';           // valoare operand
    // operator
  vst = eval(r→st);               // valoare din subarbore stanga
  vdr = eval(r→dr);               // valoare din subarbore dreapta
  switch (r→val) {                // r→val este un operator
    case '+':  return vst + vdr;
    case '*':   return vst * vdr;
    case '-':   return vst - vdr;
    case '/':   return vst / vdr;
  }
  return 0;
}
```

Observații

- Algoritmul de creare a unui arbore, pornind de la forma postfixată sau prefixată, seamănă cu algoritmul de evaluare a unei expresii postfixate sau prefixate

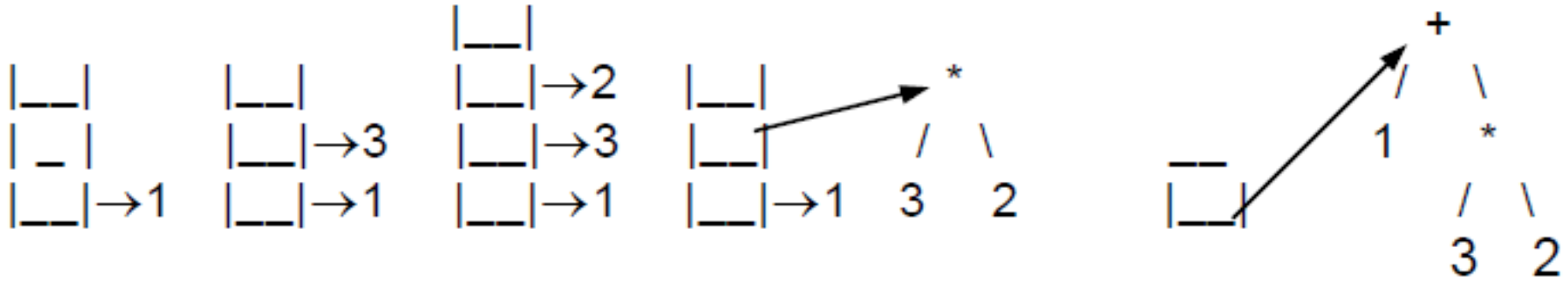
Exemplu

- Funcție care folosește o stivă de pointeri la noduri și creează (sub)arbori, care se combină treptat într-un singur arbore final

```
tnod * buidtree ( char * exp) {        // exp= sir postfixat terminat cu 0
  Stack s ; char ch;                   // s este o stiva de pointeri void*
  tnod* r=NULL;                        // r= adresa radacina subarbore
  initSt(s);                           // initializare stiva goala
  while (ch=*exp++) {                  // repeta pana la sfarsitul expresiei exp
     r=new tnode;                      // construire nod de arbore
     r→val=ch;                         // cu operand sau operator ca date
     if (isdigit(ch))                  // daca ch este operand
        r→st=r→dr=NULL;                // atunci nodul este o frunzã
     else {                            // daca ch este operator
       r→dr =(tnod*)pop (s);           // la dreapta un subarbore din stiva
       r→st= (tnod*)pop (s);           // la stanga un alt subarbore din stiva
     }
     push (s,r);                  // pune radacina noului subarbore in stiva
  }
  return r;                       // radacina arbore creat { return(tnod*)pop(s);}
}
```

Exemplu

- Pentru expresia postfixată:
- 1 3 2 * + 8 4 / -
- conținutul stivei după 5 pași este:



Exemplu

- Funcție care creează un arbore binar, pornind de la o expresie prefixată

```
tnod* build ( char p[], int & i) {          // p este sirul prefixat, terminat cu zero
  tnod* nou= (tnod*) malloc(sizeof (tnod));   // creare nod nou
  if (p[i]==0) return NULL;                  // daca sfarsit sir prefixat
  if ( isdigit(p[i])) {                      // daca este o cifra
      nou→val=p[i++];                        // se pune operand in nod
      nou→st=nou->dr=NULL;                   // nodul este o frunza
    }
    else {                                   // daca este operator
      nou→val=p[i++];                        // se pune operator in nod
      nou→st= build(p,i);                    // primul operand
      nou→dr= build (p,i);                   // al doilea operand
    }
    return nou;                              // nod creat (in final, radacina)
}
```