Expresii în forma poloneză

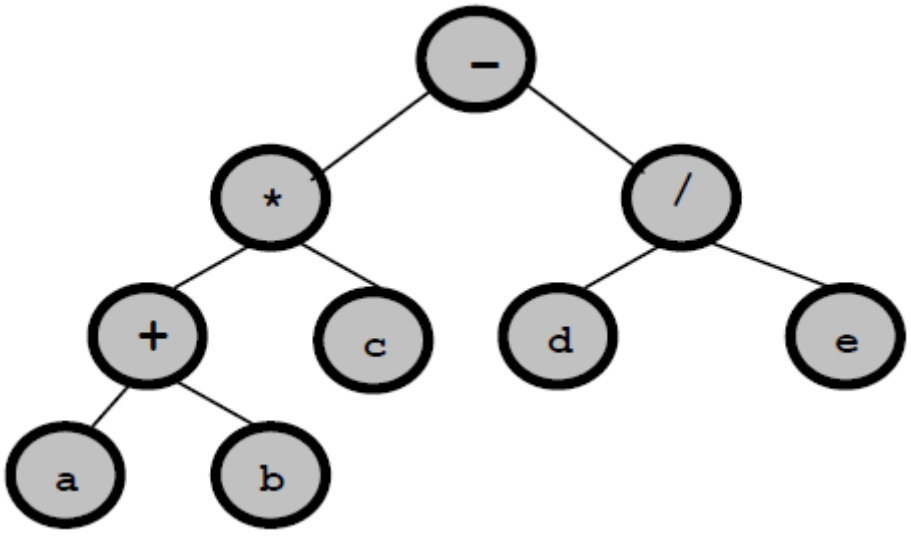
Se dă o expresie aritmetică ce foloseşte operatorii: +,-, *, / ,(,).
Fiecare operand este o literă. Se cere să se convertească această expresie într-o expresie în forma poloneză (postfixată, expresie scrisă fără paranteze).
Definim expresia recursiv, astfel:
- un operand este o expresie în forma poloneză;
- dacă E1 şi E2 sunt două expresii în formă poloneză şi # un operator atunci E1 E2 # este o expresie în forma poloneză;
- orice expresie în forma poloneză se obţine aplicând regulile de mai sus, de un număr finit de ori.
exemple:

a+b - > ab+
a*(b+c) -> abc+*
a*(b+c)-e/(a+d)+h -> abc+*ead+/- h +

Expresiile aritmetice pot fi reprezentate utilizând arbori binari, respectând următoarele reguli:

- nod neterminal - operație
- nod terminal - valoare
- pentru fiecare nod neterminal subarborele din stânga şi cel din dreapta reprezintă, în această ordine, cei doi operanzi;
- rădăcina corespunde ultimei operaţii executate la evaluarea expresiei.

expresia (a+b)*c-d/e are forma [poloneza ab+c*de/- si reprezentarea grafica:



Pentru construcţia arborelui, vom acorda priorităţi operatorilor şi operanzilor (mai puţin parantezelor), după cum urmează:
- prioritatea iniţială a operatorilor '+', '-' este 1;
- prioritatea iniţială a operatorilor '*', '/' este 10;
- la prioritatea unui operator se adună 10 pentru fiecare pereche de paranteze între care se găseşte;
- prioritatea unui operand este 1000.

În program, acest lucru se realizează astfel:
- se citeşte expresia aritmetică în variabila e;

- se utilizează o variabilă j care indică ce număr se adaugă la prioritatea iniţială a unui operator (la întâlnirea unei paranteze deschise j creşte cu 10, iar la întâlnirea unei paranteze închise j scade cu 10);

- se parcurge expresia caracter cu caracter şi se pun în vectorul p priorităţile acestor operatori şi operanzi (mai puţin ale parantezelor);
- în efp se construieşte expresia fără paranteze (la expresia aritmetică iniţială lipsesc parantezele), iar în pfp se obţine vectorul priorităţilor, din care, spre deosebire de p, lipsesc componentele corespunzătoare parantezelor (acestea nu aveau nici o valoare).

Utilizând efp şi pfp, cu ajutorul funcţiei arb, se construieşte arborele ataşat expresiei aritmetice. Un nod al acestui arbore are ca informaţie utilă un operator sau un operand.

(a+b)*c -d*8+ (9/5-4)/3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
e	(	a	+	b	)	*	c	-	d	*	8	+	(	9	/	5	-	4	)	/	3
p	0	1000	11	1000	0	10	1000	1	1000	10	1000	1	0	1000	20	1000	11	1000	0	10	1000

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
E	a	+	b	*	c	-	d	*	8	+	9	/	5	-	4	/	3
pr	1000	11	1000	10	1000	1	1000	10	1000	1	1000	20	1000	11	1000	10	1000

http://www.lbi.ro/~leuciuc/formaPoloneza.pdf

ab+c*d8*-95/4-3/+

```
#include <iostream>
using namespace std;
struct Nod
{
    char op;
    Nod *as,*ad;
};
char e[30],E[30],a;
int pr[30],p[30],i,j,n;
Nod *c;
Nod* arb(int li,int ls,char E[30],int pr[30])
{
    Nod* c;
    int i,j,mini=pr[ls];
    i=ls;
    for (j=ls; j>=li; j--)
        if (pr[j]<mini)
        {
            mini=pr[j];
```

```
            i=j;
        }
    c=new Nod;
    c->op=E[i];
    if(li==ls) c->as=c->ad=0;
    else
    {
        c->as=arb(li,i-1,E,pr);
        c->ad=arb(i+1,ls,E,pr);
    }
    return c;
}
void parc(Nod* c)
{
    if (c)
    {
        parc(c->as);
        parc(c->ad);
        cout<<(c->op);
    }
}
main()
{
    j=0;n=1;
    cin>>a;
    while (a!='.')
    {
        e[n++]=a;
        cin>>a;
    }
    n--;
    for (i=1; i<=n; i++)
        switch(e[i])
        {
        case ')': j-=10; break;
        case '(': j+=10; break;
        case '+': p[i]=j+1;  break;
        case '-': p[i]=j+1;  break;
        case '*': p[i]=j+10; break;
        case '/': p[i]=j+10;  break;
        default:  p[i]=1000;
        }
    j=1;
    for (i=1; i<=n; i++)
        if ( e[i]!=')' && e[i]!='(' )
              E[j]=e[i], pr[j]=p[i], j++;
    for(int i=1;i<=n;i++) cout<<p[i]<<' ';
    cout<<endl;
    for(int i=1;i<=n;i++) cout<<e[i]<<' ';
    cout<<endl;
    for(int i=1;i<=n;i++) cout<<E[i]<<' ';
    cout<<endl;
    c=arb(1,j-1,E,pr);
```

```
    parc(c);
}

///  (a+b)*c -d*8+ (9/5-4)/3.
```

Se da o expresie aritmetica in forma poloneza prefixata. Expresia este formata din operatorii + - / * %, iar operanzii sunt litere mici.
Afisati expresia in forma normala (infixata) si in forma postfixata.

Exemplu:

expresie.in	expresie.out
-*bb**4ac	((b*b)-((4*a)*c)) bb*4a*c*-

```
#include <fstream>
#include <cstring>

using namespace std;

ifstream fin("expresie.in");
ofstream fout("expresie.out");

char op[]="+-*/%";

struct nod
{
    char info;
    nod *st;
    nod *dr;
};

void read(nod* &r)
{
    char x;
    fin>>x;
    if(strchr(op,x)==NULL)
    {
        r=new nod;
        r->info=x;
        r->st=NULL;
        r->dr=NULL;
    }
    else
```

```
    {
        r=new nod;
        r->info=x;
        read(r->st);
        read(r->dr);
    }
}
void SRD(nod *r)
{
    if(strchr(op,r->info)) fout<<"(";
    if(r->st!=NULL) SRD(r->st);
    fout<<r->info;
    if(r->dr!=NULL) SRD(r->dr);
    if(strchr(op,r->info)) fout<<")";
}
void SDR(nod *r)
{
    if(r->st!=NULL) SDR(r->st);
    if(r->dr!=NULL) SDR(r->dr);
    fout<<r->info;
}
int main()
{
    nod *r;
    read(r);
    SRD(r);
    fout<<endl;
    SDR(r);
    return 0;
}
```

Se da o expresie aritmetica in forma poloneza prefixata. Expresia este formata din operatorii + - / * %, iar operanzii sunt dintr-un singur caracter.
Construiti arborele binat asociat expresiei citite si afisati expresia in forma normala (infixata).
Exemplu:
/-+*5314+27
in forma normala este (((5*3)+1)-4)/(2+7)

```
#include <fstream>
#include <cstring>
using namespace std;
ifstream fin("date.in");
ofstream fout("date.out");
```

```
int S[100],D[100],T[100],n;
char A[100];
void SRD(int v)
{
    if(strchr("+/-*%",A[v]) && v!=0)fout<<"(";
    if(S[v]) SRD(S[v]);
    fout<<A[v];
    if(D[v]) SRD(D[v]);
    if(strchr("+/-*%",A[v]) && v!=0)fout<<")";
}
int main()
{
    int n,v,pus;
    fin>>A; n=strlen(A)-1; T[0]=-1;
    for(int i=1;i<strlen(A);i++)
    {
        pus=0; v=0;
        while(!pus)
        {
          while(S[v] && strchr("/*-+%",A[S[v]])) v=S[v];
          if(S[v]==0) S[v]=i,pus=1,T[i]=v;
          else
            if(D[v] && strchr("/*-+%",A[D[v]])) v=D[v];
            else
            {
              if(D[v]==0) D[v]=i, pus=1,T[i]=v;
              else
              {
                v=T[v];
                while(T[v]!=-1&&!(D[v]==0||strchr("/*-+%",A[D[v]])))
                          v=T[v];
                if(D[v]==0) D[v]=i,pus=1,T[i]=v;
                else
                   if(strchr("/*-+%",A[D[v]])) v=D[v];
              }
           }
        }
    }
    SRD(0);
    return 0;
}
```