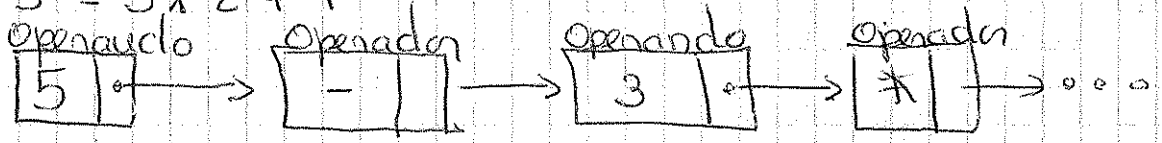
Cátia Gonçalo nº 58903 4/5/2010

Co_Produto Cartesiano

data AvBin A = Node A AnvBin A AnvBin A
| Nil

Exemplo:

5 - 3 * 2 + 7



↳ Union

```
#define operando 1001
#define operador 1002
typedef union uElem
        {
            int operando;
            Char operador;
        } Elemento;

typedef struct sExp
        {
            Elemento elem;
            Struct sExp * Seg;
        } * Exp, No exp;

int main()
{
    Exp e1 = Null , e2 = Null;
    e1 = insein...
    show Exp (e1);
}

void
```

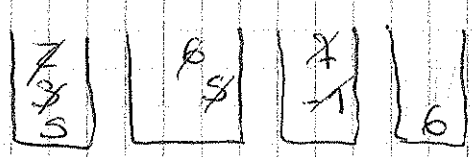
```
void showExp(Exp e)
{
  if (e)
  {
    showElem(e->elem);
    showExp(e->seg);
  }
}

typedef struct sElem
{
  int tipo;
  union uValor
  {
    int operando;
    char operador;
  } valor;
} Elemento;

void showElem(Elemento elem)
{
  switch(elem.tipo)
  {
    case Operando: printf("%d", elem.valor.operando);
                   break;
    case operador: printf("%c", elem.valor.operador);
                   break;
  }
}
```

5 - 3*2 + 7 ↘ posfixo

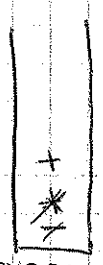
5 3 2 * - 7 +



Algoritmo: Infixo → Pós-fixo

Input
5 - 3*2 + 7

output
5 3 2 * - 7 +

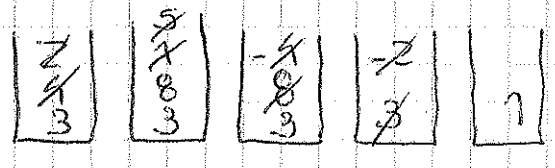


operadores

Input = 3 + 4 * 2 / (1 - 5)

Token	Ação	output	stack
3	Copia	3	
+	Push	3	+
4	copia	3 4	+
*	Push	3 4	* +
2	copia	3 4 2	* +
/	Pop	3 4 2 *	+
	Push	3 4 2 *	/ +
(	Push	3 4 2 *	(/ +
1	copia	3 4 2 * 1	(/ +
-	Push	3 4 2 * 1	- (/ +
5	copia	3 4 2 * 1 5	- (/ +
)	Popate "("	3 4 2 * 1 5 -	/ +
Null	Pop ate vazia	3 4 2 * 1 5 - / +	

3 4 2 * 1 5 - / +



```
int calc_exp (Exp e)
{ int opn, opz;
  Int stack = Null;
while (e)
{
  switch ( e -> elem. tipo )
  {
    case operando : stack = push (stack, e -> elem. valor.
                                  operando);
                    break;
    case Operador : switch (e -> elem. valor. operador)
                    {
                      case '+' : opz = top ( stack); stack = pop
                                                             (stack);
                                 opn = top(stack); stack = pop(stack);
                                 stack = push (stack, opn + opz);
                                 break;
                      ...
    e = e -> seg;
}
{
return top(stack);
```