
Scanner

X25570_ca

INTRODUCCIÓ:

En aquest exercici considerarem un llenguatge de programació amb expressions i instruccions. Les expressions són sobre els operadors $+$, $-$, $*$, i sobre operands naturals i variables enteres. Hi ha dos tipus d'instruccions:

- Instrucció d'assignació $x = e$, on x és una variable i e és una expressió. Per exemple $x=3+4*y$.
- Instrucció d'escriure per la sortida estandard $\text{Print}(e)$, on e és una expressió. Per exemple $\text{Print}(3+4*y)$.

Cada instrucció apareix en una nova línia. Aquí tenim un exemple de programa:

```
x = 3
y=3+x
z=2 * (x+y)
Print(z - y+x)
```

El nostre objectiu serà escanejar cada línia, identificar els elements bàsics que hi apareixen i retornar una llista amb aquests elements. Per exemple, donat l'string `"z=2 * (x+y)"`, haurem de produir la llista de parelles d'strings:

```
<"variable", "z">, <"operator", "=">, <"number", "2">, <"operator", "*">, <"marker", "(">,
<"variable", "x">, <"operator", "+">, <"variable", "y">, <"marker", ")">
```

EXERCICI:

Implementeu una funció que, donat un string `line` amb una instrucció en el nostre llenguatge de programació, retorna un `list<pair<string, string> >` que representa la seqüència d'elements atòmics del llenguatge que hi apareixen. Aquesta és la capcelera:

```
// Pre: "line" conté una instrucció correcta del llenguatge.
// Post: retorna la llista d'elements atòmics del llenguatge que apareixen a "line"
list<pair<string, string> > scanner(string line);
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `scanner.hpp`, `utils.hpp`, `utils.cpp`. Us falta crear el fitxer `scanner.cpp` amb els corresponents `includes` i implementar-hi la funció `scanner` que hem explicat. Valdrà la pena que utilitzeu algunes de les funcions oferides a `utils.hpp`. Quan pugeu la vostra solució al jutge, només cal que pugeu un tar construït així:

```
tar cf solution.tar scanner.cpp
```

Entrada

L'entrada té una seqüència de línies, on cadascuna és una instrucció del llenguatge. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquesta entrada. Només cal que implementeu la funció abans esmentada.

Sortida

El programa dona com a sortida, per a cada línia, la llista d'elements atòmics que hi apareixen. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
a=3*2+1
Print (a-(7-a)-(2*a))
a=a+6-3
Print (a-(3*a))
b=a*a-(a*2)
Print (4*a)
Print (b-(2-4))
Print (3*b)
b=7-4-(a*4)
Print (a+3-b+(7-3))
```

Exemple d'entrada 2

```
Print ( 4+7+( 6-8)-(8-2))
a=1- 1
Print ((a-7* a)*(a*1))
b=a+1
b=3
Print (8+(7- 2))
b=b-3
Print ((b-2-b)*(6-(5-a)+ (a-3-(4+9))))
Print ( 9)
Print ((3* 9+4-9*(5- a)) *(b-a*7))
```

Exemple d'entrada 3

```
a=36*87+ 50
b=63- (a-a-( 43-a))-99- ( 27-81)
a=9+ (85-(100-a))
Print (2+ (41+b+ (b-b)-( 84+ 25+4*71))
a=38-(75+59)-19*44-(44- (5*29-97*44))
a=a-21-27-83*25-( a+(74+34*61-(b+ a+41*85))
a=(21-23)* 82-(a-(a-41) )
Print (a-85*43+ (a-a) - (b-a+(b+91))
Print (52+b)
Print (42* 34+39-85)
Print ( 8+b-b)
Print (40+64-(32-83))
Print (82-(32+a)+(a- (a+ b+(b-51) -(35-76-(94-6))
b=b- b
bb=b-a-a-80*( 73+35) +(a+15+(a-55-87))
bb=21-b*(bb-bb-(10+a) )
a= bb*82- (bb+26)*93
bb= b+86-a-(93-26*bb)-(26* 71- 9-(a-72-(a+38)14,(78)-58-18)
bb=63- ( a- 16)-(92-38)
Print (55*76-13+( 61- 15)*(84+b) )
```

Exemple de sortida 1

```
<"variable", "a"> <"operator", "="> <"number", "3"> <"oper
<"function", "Print"> <"marker", "("> <"variable", "a"> <
<"variable", "a"> <"operator", "="> <"variable", "a"> <"op
<"function", "Print"> <"marker", "("> <"variable", "a"> <
<"variable", "b"> <"operator", "="> <"variable", "a"> <"op
<"function", "Print"> <"marker", "("> <"number", "4"> <"op
<"function", "Print"> <"marker", "("> <"variable", "b"> <
<"function", "Print"> <"marker", "("> <"number", "3"> <"op
<"variable", "b"> <"operator", "="> <"number", "7"> <"oper
<"function", "Print"> <"marker", "("> <"variable", "a"> <
```

Exemple de sortida 2

```
<"function", "Print"> <"marker", "("> <"number", "4"> <"op
<"variable", "a"> <"operator", "="> <"number", "1"> <"oper
<"function", "Print"> <"marker", "("> <"marker", "("> <"va
<"variable", "b"> <"operator", "="> <"variable", "a"> <"op
<"variable", "b"> <"operator", "="> <"number", "3">
<"function", "Print"> <"marker", "("> <"number", "8"> <"op
<"variable", "b"> <"operator", "="> <"variable", "b"> <"op
<"function", "Print"> <"marker", "("> <"marker", "("> <"va
<"function", "Print"> <"marker", "("> <"number", "9"> <"ma
<"function", "Print"> <"marker", "("> <"marker", "("> <"nu
```

Exemple de sortida 3

```
<"variable", "a"> <"operator", "="> <"number", "36"> <"ope
<"variable", "b"> <"operator", "="> <"number", "63"> <"ope
<"variable", "a"> <"operator", "="> <"number", "9"> <"oper
<"function", "Print"> <"marker", "("> <"number", "2"> <"op
<"variable", "a"> <"operator", "="> <"number", "38"> <"ope
<"variable", "a"> <"operator", "="> <"variable", "a"> <"op
<"variable", "a"> <"operator", "="> <"marker", "("> <"numk
<"function", "Print"> <"marker", "("> <"number", "52"> <"c
<"function", "Print"> <"marker", "("> <"number", "42"> <"c
<"function", "Print"> <"marker", "("> <"number", "8"> <"op
<"function", "Print"> <"marker", "("> <"number", "40"> <"c
<"function", "Print"> <"marker", "("> <"number", "82"> <"c
<"variable", "b"> <"operator", "="> <"variable", "b"> <"op
<"variable", "bb"> <"operator", "="> <"variable", "b"> <"c
<"variable", "bb"> <"operator", "="> <"number", "21"> <"op
<"variable", "a"> <"operator", "="> <"variable", "bb"> <"c
<"variable", "bb"> <"operator", "="> <"variable", "b"> <"c
<"variable", "bb"> <"operator", "="> <"number", "63"> <"op
<"function", "Print"> <"marker", "("> <"number", "55"> <"c
```

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:05:28.103Z

© *Jutge.org*, 2006–2026.
<https://jutge.org>