Avaluar expressions amb variables — X68760_ca

PRELIMINARS:
En aquest exercici assumim que ja heu resolt un exercici anterior a on havieu d'avaluar expressions sense variables. Haureu d'adaptar la solució d'aquell exercici al cas en que la expressió a avaluar té variables. Els valors de les variables es passaran com a paràmetre en un `map<string,int>`.

INTRODUCCIÓ:
Considerarem arbres que representen expressions sobre els operadors `+`, `-`, `*`, i sobre operands naturals i variables enteres (una variable serà una seqüència de lletres minúscules). Per exemple, l'arbre `-(+(3,*(4,x)),y)` representa l'expressió `3+4*x-y`.
Per a guardar els valors assignats sobre les variables usarem un map d'identificadors de variables a enters declarat així en el nostre programa C++:

```
#include <map>
...
map<string,int> variable2value;
```

Aquest tipus de dades es pot utilitzar així en C++:

```
// Guarda 3 com a valor associat a l'string "hola":
variable2value["hola"] = 3;
// Suma 2 al valor associat a l'string "hola":
variable2value["hola"] = variable2value["hola"] + 2;
// Escriu el valor associat a l'string "hola" sobre la sortida estandard
// (en aquest cas escriuria 5):
cout << variable2value["hola"];
```

EXERCICI:
Implementeu una funció que, donats els valors actuals de les variables, i donat un arbre binari d'strings que representa una expressió correcta sobre naturals i variables enteres, i operadors `+`, `-`, `*`, retorna la seva avaluació. Aquesta és la capcelera:

```
// Pre:  t és un arbre no buit que representa una expressió correcta
//       sobre naturals i variables enteres, i els operadors +,-,*.
//       Totes les variables que apareixen a t estan definides a variable2value.
//       Les operacions no produeixen errors d'overflow.
// Post: Retorna l'avaluació de l'expressió representada per t.
int evaluate(map<string,int> &variable2value, const BinaryTree<string> &t);
```

Aquí tenim un exemple de paràmetre d'entrada de la funció i la corresponent sortida:

```
evaluate({x:2, y:3}, *(+(1,x),-(5,y))) = 6
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `evaluate.hpp`, `utils.hpp`, `utils.cpp`. Us falta crear el fitxer `evaluate.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. 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 evaluate.cpp
```

Entrada

L'entrada té una seqüència de casos, on cada cas té una primera linia amb una descripció dels valors de les variables (una seqüència de parelles <variable,valor>), i una segona linia amb 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

Per a cada cas, la sortida té una linia amb la corresponent avaluació de l'arbre. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta avaluació. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
a 2 b 2
+(-(-(a,5),-(7,b)),+(9,5))
a 6 b 7 c 1
*(+(*(5,7),a),+(-(-(9,1),-(7,b)),-(-(c,a),1)))
a 3
*(a,2)

1
a 3 b 8
*(a,+(2,*(-(b,8),*(a,5))))
a 4
*(a,6)

*(6,2)
a 3
+(*(a,4),-(3,a))
a 5
*(+(+(7,*(7,a)),*(+(3,a),2)),3)

-(2,8)
```

Exemple de sortida 1

```
6
-164
6
7
6
24
12
12
174
-6
```

Exemple d'entrada 2

```
a 82
-(*(-(+(42,6),-(*(56,9),+(45,92))),-(76,-(+(a,78),+(40,29)))),*(41,a))
a 10 b 88
-(-(a,b),7)
a 64
-(15,*(-(+(a,38),+(64,75)),a))
a 59 b 30
+(-(-(*(5,a),+(55,22)),-(*(78,11),+(a,b))),87)
a 60
+(-(+(56,92),*(a,7)),10)
a 16 b 74
-(-(+(+(45,31),+(a,b)),*(53,68)),-(-(-(a,47),*(93,17)),b))
a 60 b 53 aa 94
-(-(-(*(41,20),*(a,b)),+(-(60,aa),-(+(88,*(98,71)),*(-(a,62),+(aa,a))))),+(-(aa,31),-(94,71)))
a 64 b 90 aa 38
+(+(-(-(a,41),*(-(1,+(82,8)),-(-(b,aa),90))),b),-(95,59))
a 22 b 40
-(+(*(39,94),*(52,a)),*(-(4,a),b))
a 95 b 98 aa 39
-(-(-(-(*(89,52),-(85,+(83,78))),*(45,40)),-(-(*(35,13),a),-(*(b,63),*(b,aa)))),-(-(a,a),+(+(83,
a 83 b 3
+(92,-(*(56,83),-(-(-(*(66,6),-(47,17)),-(9,+(64,67))),
a 71 b 76
+(*(*(-(45,73),*(35,8)),*(59,-(a,a))),+(-(*(a,44),+(b,
a 27 b 69 aa 30
-(-(-(*(17,97),88),+(+(-(a,a),*(21,84)),55)),+(*(b,+(b,
a 98
-(*(33,-(a,75)),*(23,a))
a 22 b 63 aa 20
*(a,*(*(+(98,30),-(a,a)),*(+(3,b),aa)))
a 5
-(+(52,89),a)
a 60 b 54
+(-(a,38),+(49,*(-(a,-(41,b)),a)))
a 96 b 46
+(+(+(62,79),-(*(69,34),88)),-(*(66,13),-(86,-(a,b))))
a 34 b 83 ab 47
-(-(87,31),-(a,-(*(91,b),+(51,ab))))
a 66
+(+(+(69,a),*(69,44)),+(+(81,23),-(*(5,a),12)))
```

Exemple de sortida 2

```
1423
-85
2383
-464
-262
-1752
-9766
-3233
5530
```

```
5198
1961
3055
-7356
-1495
0
136
4451
3221
7659
3593
```

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:21:59.255Z