Eines informàtiques per a les matemàtiques

Part IV: Programació en C

1er de grau de Matemàtiques
Universitat Autònoma de Barcelona

29 d'abril de 2016

Resum

Treball de programació en `C` sobre el tema:
Compilació i avaluació de la Notació Polonesa Inversa.

Índex

1 Introducció [1]

La notació polonesa inversa (**RPN** en anglès, Reverse Polish Notation) o notació postfix és un mètode d'introducció de dades alternatiu a l'algebraic. Va ser creat pel filòsof i científic de la computació australià Charles Leonard Hamblin a mitjans dels anys 1950. Deriva de la notació polonesa inventada pel matemàtic polonès Jan Łukasiewicz.
A la dècada dels 60 aquest mètode va ser introduït als ordinadors. Posteriorment, Hewlett-Packard (HP) el va aplicar per primera vegada a la calculadora de sobretaula HP-9100A el 1968.
El seu principi és el d'avaluar les dades directament quan s'introdueixen i manejar-les dintre d'una "pila LIFO" (Last In First Out), mètode que optimitza els processos a l'hora de programar. Bàsicament les diferències amb el mètode algebraic són que, per avaluar les dades directament quan s'introdueixen, no és necessari ordenar-ne l'avaluació, i que per a executar una ordre, primer s'han d'introduir tots els seus arguments. Així, per a fer una simple suma $a + b = c$ el RPN ho expressaria

```
a b +
```

presentant el resultat c directament.
Cal tenir en compte que la notació polonesa inversa no és literalment la imatge especular de la notació polonesa: amb operadors no-commutatius (com la resta o la divisió), l'ordre dels operands ha de romandre constant. Així per tant, 6/2 es tradueix a la notació polonesa com

```
/ 6 2
```

i a la notació polonesa inversa com

```
6 2 /
```

L'objectiu d'aquest treball és escriure un programa que sigui capaç de rebre una *entrada* en notació polonesa inversa, *compilar-la* a un format de representació intern, i *avaluar-la* obtenint el valor de l'expressió. Podeu trobar informació abundantíssima a internet sobre la RPN, incloent programes en C per treballar-hi que us poden inspirar, però amb el que hi ha a la pàgina de Wikipèdia `https://ca.wikipedia.org/wiki/Notaci%C3%B3_polonesa_inversa` en teniu suficient per començar.

1.1 Representació interna

Una expressió RPN és una successió de *termes* separats per espais. El programa haurà de reconèixer i treballar amb quatre tipus de termes:

- Nombres. Per exemple: `6`, `-1.1`, `1.2e6`.
- Constants numèriques. Per exemple: `pi`, `e`.
- Operadors. Per exemple: `+`, `-`, `*`, `/`, `^`.
- Funcions. Per exemple: `exp`, `sin`, `cos`, `sqrt`.

[1] Informació teòrica i històrica extreta i adaptada de la Wikipèdia.

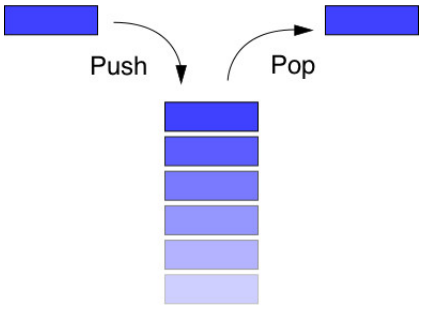

Figura 1: Representació d'una pila. [2]

Internament, el programa representarà l'expressió RPN com una matriu $m \times 2$, o vector de vectors de dues components `double` (un vector `{c,x}` per a cada terme) on `c` és un codi que valgui (per exemple) 0 si el terme és un nombre, 1 si és una constant, 2 si és un operador, i 3 si és una funció; i `x` és, en el primer cas, el nombre pròpiament, i en els altres novament un codi.

Exemple 1. Els termes `-1.1`, `pi`, `+`, `-`, `cos` es podrien codificar, per exemple, així:

- `"-1.1"` $\longmapsto$ `{0,-1.1}`
- `"pi"` $\longmapsto$ `{1,0}`
- `"+"` $\longmapsto$ `{2,0}`
- `"-"` $\longmapsto$ `{2,1}`
- `"cos"` $\longmapsto$ `{3,2}`

1.2 Estructura "pila LIFO"

En informàtica, la memòria en pila és una estructura de dades seqüencial (que conté elements ordenats) amb aquestes restriccions d'accés:

- Només es pot afegir elements al cim de la pila.
- Només es pot treure elements del cim de la pila.

Per analogia amb objectes quotidians, una operació empilar (push) equivaldria a posar un plat sobre una pila de plats, i una operació desempilar (pop) a retirar-lo. Una pila de nombres és l'estructura de dades que resulta escaient per implementar l'*avaluació* d'expressions en notació polonesa inversa. Us recomanem que implementeu la pila com un vector de nombres i una variable entera (o un apuntador) que indiqui el lloc del vector on es troba el cim de la pila.
Necessitareu funcions de `C` per a *empilar* o *desempilar* nombres, sempre prenent precaucions perquè si s'ha arribat a omplir el vector no es pugui empilar un nou nombre, o si la pila és buida no es pugui desempilar.

[2] Font: wikipèdia, by User:Boivie - made in Inkscape, by myself User:Boivie. Based on Image:Stack-sv.png, originally uploaded to the Swedish Wikipedia in 2004 by sv:User:Shrimp, Domini públic `https://commons.wikimedia.org/w/index.php?curid=1439935`

2 Requeriments del programa a fer

L'objectiu és fer un programa que donada una expressió en notació polonesa inversa *sintàcticament correcta* l'avaluï i presenti el valor resultant. En el cas d'introduir una expressió incorrecta, s'ha de presentar un missatge d'error.
L'entrada i sortida es farà per la consola (`STDIN`, `STDOUT`).

Exemple 2. Suposem que s'introdueix la següent expressió:

```
5 2 pi cos - 4 * + 3 -
```

La sortida hauria de ser `14.0000` (o un format numèric equivalent). Per passos, el programa hauria de compilar en primer lloc l'expressió "`5 2 pi cos - 4 * + 3 -`" per obtenir la representació interna, de l'estil de

```
{{0,5},{0,2},{1,0},{3,2},{2,1},{0,4},{2,2},{2,0},{0,3},{2,1}}.
```

A continuació s'ha d'avaluar aquesta expressió, el que pas a pas dóna lloc a:

1. Empilar 5.
2. Empilar 2.
3. Empilar el valor de `pi`, 3.14159265 amb la precisió que hàgiu decidit.
4. Desempilar un valor (pi), calcular-li el cosinus (-1) i empilar-lo.
5. Desempilar dos valors, calcular la resta ($2-(-1)=3$) i empilar-la.
6. Empilar 4.
7. Desempilar dos valors, calcular el producte ($3*4=12$) i empilar-lo.
8. Desempilar dos valors, calucular la suma ($5+12=17$) i empilar-la.
9. Empilar 3.
10. Desempilar dos valors, calcular la resta ($17-3=14$) i empilar-la.

Al final del procés es presenta per consola el valor que hi ha al cim de la pila (14). És instructiu representar el contingut de la pila pas a pas; assegureu-vos d'haver entès completament aquest exemple abans de començar a programar.

2.1 Suggeriments i pseudocodi

2.1.1 Lectura de l'expressió i compilació

És recomanable usar la funció `fgets` per llegir l'expressió RPN sencera, més que `scanf`. Llavors tindreu l'expressió en una cadena, la qual caldrà recórrer per traduir cada terme al codi intern. Per identificar els termes i decidir quins són nombres, constants, operadors o funcions, us poden ser d'utilitat les funcions `isspace`, `isdigit`, etc. del fitxer de capçalera `<ctype.h>`; podeu trobar-ne informació a la pàgina `http://www.cplusplus.com/reference/cctype/`. Per a convertir els nombres d'una cadena de caràcters a format numèric, recordeu la comanda `sscanf`. Per a identificar les constants, operadors i funcions, recordeu la comanda `strcmp`.

2.1.2 Avaluació de l'expressió compilada

El següent pseudocodi us pot ser d'utilitat.

```
Entrada: RPN[] és una expressió, codificada com un vector de vectors {c,x}.
t és el nombre de termes a RPN[].
pila[] és una pila de nombres, inicialment buida.
Sortida: 0 si l'expressió és correcta, 1 en cas contrari.
La pila conté el valor de l'expressió, en sortir de la funció.
Algorisme: avalua(RPN,pila) és
var
constants[]: real //llista de constants
op1, op2: real
i: enter
fivar
i=0;
mentre i<t fer
si RPN[i][0]=0 // és un nombre
pila=empilar(pila,RPN[i][1]);
sino si RPN[i][0]=1 // és una constant
pila=empila(pila,constants[(int)RPN[i][1]]);
sino si RPN[i][0]=2 // és una operació
si esbuida(pila) retorna 1
sino op2=desempilar(pila);
fisi
si esbuida(pila) retorna 1
sino op1=desempilar(pila);
fisi
empilar(operació_RPN[i][1] (op1,op2));
sino // és una funció
si esbuida(pila) retorna 1
sino op1=desempilar(pila)
fisi
empilar(funció_RPN[i][1] (op1));
fisi
fimentre
si esbuida(pila) retorna 1
sino retorna 0
fisi
fialgorisme
```

Fixeu-vos que alguns punts són molt simplificats. Haureu de pensar com es determinen `operació_RPN[i][1]` i `funció_RPN[i][1]`.

2.2 Variants i puntuació

La part més important del programa és l'algoritme d'avaluació de la RPN. Una variant simplificada consistiria en fer un programa que avalua directament l'expressió (sense compilar-la al "format intern") llegint terme a terme amb `scanf`. (El programa resultant es comportaria de forma semblant a una calculadora hp com `http://hp15c.com/web/hp15c.html`). Un treball així podrà optar com a màxim a una qualificació de 5; una possibilitat és començar escrivint aquest per assegurar-se que avalua correctament les expressions, i anar-lo modificant fins aconseguir el treball final demanat.
En la puntuació del programa entregat es valorarà principalment que es compili correctament sense errors i faci allò demanat. Es tindran també en compte aspectes formals, com l'estructura de funcions, la llegibilitat del codi, l'absència de *warnings* en compilar, i funcionals, com la varietat de constants, operacions i funcions acceptades, el tractament dels errors, les possibles restriccions en les expressions acceptades.

3 Lliurament i avaluació

- Us recomanem que rellegiu la informació de la Guia Docent sobre Avaluació i en especial el paràgraf sobre els **treballs individuals**. Concretament, *no compartiu arxius*: entregar un programa copiat total o parcialment pot implicar suspendre *l'assignatura*.

- Els algorismes proposats són coneguts i en podeu trobar ajuda per internet: pseudocodis, videos explicatius... Compte amb copiar *codi*! Haureu d'explicar com funciona el programa en una entrevista.

- Tots els fitxers que entregueu han de començar per `NomCognom` (on `Nom` vol dir el vostre nom i `Cognom` el vostre cognom). Per exemple, el fitxer font es pot dir `JoanGarcia.c`.

- Heu de lliurar el codi del programa com un fitxer `.c`. El codi lliurat s'haurà de poder compilar i executar als ordinadors del campus, amb la instrucció

```
gcc -Wall -O3 -o NomCognom NomCognom.c -lm
```

- La data límit és el 3 de juny de 2016 a les 23:55. El lliurament s'ha de fer a través del moodle del campus virtual.

- La nota sortirà del programa i l'entrevista obligatòria posterior a l'entrega.