Interopérabilité

TP Boost

Interopérabilité Python - C++

Eric Ramat
ramat@lil.univ-littoral.fr

26 février 2007

Durée : 9 heures

---

1 Objectif

L'objectif de ce TP est de réaliser une calculatrice HP. Les calculatrices HP ont la particularité d'utiliser la notation inverse polonaise pour exprimer les calculs. Par exemple, l'expression : 3 * (4 + 7) s'écrira : 4 7 + 3 *

En pratique, sur une calculatrice en notation inverse polonaise, le calcul sera saisi de la manière suivante : "4", "enter", "7", "enter", "+", "3", "enter" et "*".

La réalisation de calculatrices en notation inverse polonaise est basée sur l'utilisation d'une pile ; c'est-à-dire, que les opérandes sont ajoutés en haut de la pile, et les résultats des calculs sont retournés en haut de la pile.

Du point de vue pratique, l'interface graphique et le moteur de calcul doivent être totalement séparés. Dans le cadre de ce TP, nous allons surtout insister sur cet aspect en déléguant l'interface à Python et la librairie PyGTK et le moteur de calcul à C++/Boost/STL. La liaison sera réalisée grâce à la librairie Boost : :python.

2 Interface graphique en Python

L'interface graphique s'occupe uniquement des éléments d'interaction (touches et affichage). Le langage utilisé est uniquement Python et le package pygtk. Aucun élément de calcul ou de précalcul doivent être présent dans cette partie. Les échanges entre d'interface et le moteur de calcul se font exclusivement sous forme de chaînes de caractères.

L'interface graphique se compose :
- d'un ensemble de touches :
  - numériques : de 0 à 9 et π ;
  - opérations : +, -, *, / ;
  - fonctions trigonométriques : sin, cos et tan ;
  - fonctions puissances : x^2, $\sqrt{x}$, x^y, e^x, lnx ;
  - changement de signe : "chs" ;
  - délimiteurs d'expression : « et » ;

  - "enter" (validation de la zone de saisie), "←" (annulation du dernier caractère saisi) et "clear" (effacement de la zone de saisie) ;
  - opérations de pile : "drop" (suppression du sommet de la pile) et "swap" (permutation des deux éléments de sommet de pile) ;
  - "undo" (annulation de la dernière opération) et "redo" (réapplique la dernière opération).
- d'une zone de saisie ;
- d'une zone d'affichage de la pile (4 niveaux) ;

Les touches numériques placent en sommet de pile un chiffre ou pi. Les touches d'opérations, de fonctions et de changement de signe dépilent les deux premiers niveaux de la pile, appliquent l'opération et place le résultat au sommet de pile.

Si la touche « est appuyée alors on entre en mode saisie d'expression et tout appui sur une touche s'inscrit dans la zone de saisie jusqu"à l'appui sur » (fin d'expression).

Le sommet de la pile est affiché en bas de la zone d'affichage de la pile. Lors de la validation d'un élément saisi par la touche "enter", ce dernier est placé au sommet de la pile donc en bas de la zone d'affichage. Les éléments déjà présents dans la pile sont décalés vers le haut. Si le nombre d'élements est supérieur à 4 alors les éléments au-delà de la position 4 n'apparaît pas.

Il faut prévoir une procédure particulière si le calcul est faux. Par exemple, s'il y a qu'un seul élément dans la pile alors que on veut appliquer un opérateur à 2 opérandes (par exemple, l'addition).

Attention, la pile ainsi que les traitements ne doivent pas être écrits en Python mais en C++ !

3 Moteur de calcul en C++

Le moteur de calcul se base principalement sur la pile. Cette pile contient exclusivement des réels. Néanmoins, il faudra ajouter des fonctions telles que le "undo" et "redo". Il ne faudra pas oublier de gérer les erreurs de calculs. Les erreurs doivent être gérées au niveau de la pile sous forme d'exception. Attention, certaines opérations comme drop, peuvent aussi soulever des exceptions.

Dans le cas de la saisie d'une expression, le moteur doit procéder à l'analyse d'une chaîne de caractères contenant l'expression. Après cette phase d'analyse, le moteur doit se comporter comme si l'utilisateur avait appuyer sur les touches. Il n'y a donc pas de traitement supplémentaire à imaginer.

Afin de faciliter le débuggage, il est demandé d'intégrer les contrôles de validité à l'aide des assertions. Par exemple, si vous développer une fonction admettant en paramètre une chaîne de caractères qui doit être non vide alors vous pouvez placer une assertion en début de fonctions afin de vérifier cette condition.

Exemple d'assertion :

dans un premier temps

```
<boost/assert.hpp>

...
BOOST_ASSERT(str.size() > 0)
...
```

Il est impératif d'écrire un minimum de code. On s'attachera donc à utiliser au maximum les librairies Boost ainsi que la STL. On identifiera pour chaque structure et/ou fonctionnalités le design pattern utilisé. Pour le "undo" et "redo", le pattern command semble tout indiqué.

4 Question supplémentaire

Il serait intéressant d'intégrer la possibilité de définir des variables. Pour cela, il faut ajouter à l'interface graphique :

- une série de touches avec les lettres de a à z ;
- une touche pour délimiter les noms des variables : " ' " ;
- une touche "sto" pour créer la variable ;

Lors de l'appui sur "sto", la pile doit contenir une valeur et le nom de la variable entre quotes (le nom étant au sommet de la pile). Les variables sont utilisables directement et à l'intérieur d'une expression. Dans ce cas, le nom n'est pas entouré de "quotes".

5 Références

- Boost : http://www.boost.org
- STL : http://www.sgi.com/tech/stl/
- PyGTK : http://www.pygtk.org