Rapport calculatrice à notation polonaise inversée

Ahmed NDIAYE et Jean-Baptiste DEGIOVANNI

Introduction

Vous pourrez trouver dans ce rapport l'analyse de notre travail et les explications nécessaires à sa pleine compréhension.

Vous pourrez de plus retrouver :

- Nos codes sources sur le dépôt google code suivant : https://lo21-ahmed-pitouli.googlecode.com/svn/trunk/
- Notre Dioxygen sur le site suivant : http://www.pitouli.fr/lo21/Dioxygen/html/index.html
- Les modélisations UML sont elles aussi sur le dépôt, tout comme ce fichier.

Justification modélisation

Les nombres

Lors de la phase de conception, nous avions envisagé deux méthodes de modélisation des nombres.

Une première méthode où tous les nombres auraient hérités d'un « Complexe », qui aurait comporté 6 attributs :

- 4 int : DenominateurReel, DenominateurImaginaire,PartieEntiereReel et PartieEntiereImaginaire
- 2 double : PartieDecimaleReel et PartieDecimaleImaginaire (compris entre -1 et +1)

La programmation aurait été simplifiée, en ayant seulement à définir les opérations une seule fois, et des conversions d'un type à l'autre complètement transparent (l'héritage se faisant par spécialisation, par exemple en mettant à 0 les partie décimal pour faire les complexes rationnels). Cependant, cela se faisait au détriment de la mémoire, car chaque nombre aurait comporté jusqu'à 5 attributs inutiles (par exemple les entiers non complexes, où les dénominateurs sont fixés à 1, et toutes les autres valeurs sauf la partie entière réelle à 0).

Dans le cadre de notre projet scolaire, qui a avant tout un but didactique, nous avons préféré éviter le "simple" héritage, dont nous pensons maîtriser la théorie, et avons préféré une méthode plus complexe, mais qui a le mérite de ne pas avoir d'attributs inutiles, et surtout de rajouter un véritable challenge, et donc de se révéler bien plus instructive. Elle a fait en particulier apparaître l'intéressante difficulté de toujours devoir retrouver sur quel nombre on travaille pour pouvoir le caster et agir en conséquence.

Dans ce modèle, on observe en particulier que les classes Rationnel et surtout Complexe ont la particularité d'être définie comme composition d'autres nombres.

Nous avons malgré tout rapidement développé les fondements de la classe "Complex" qui correspond à la "Super Classe" depuis laquelle tous les autres nombres auraient hérités. Vous pourrez la trouver dans le même dépôt SVN que les autres classes, dans les fichiers numbers.h et numbers.cpp

L'interface

Nous avons choisi de faire une calculatrice compacte, où toutes les options sont accessibles via des boutons visibles si besoin, mais qui n'encombrent pas le champ visuel, grâce à un rangement dans un panneau à onglet.

Des raccourcis permettent un usage complet au clavier, des raccourcis spécifiques permettant d'accéder à chaque panneau (basiqUe, dIvers,trigO et Pile), et permettant alors d'utiliser les raccourcis spécifique aux boutons alors visibles. Cependant, les fondamentaux sont eux toujours accessibles, visibles ou non (les numériques et les opérateurs arithmétiques classiques).

Nous avons décidé que les expressions concrètes n'avaient d'intérêt que si elles s'opposaient à des expressions calculées en continues. Ainsi, nous avons un mode "calcul automatique" qui peut être activé ou désactivé à volonté. Lors qu'il est activée, l'utilisateur peut écrire autant de formules qu'il le souhaite dans la barre de saisie. Quand il le désactive, le résultat est empilé sous forme d'une expression concrète qui n'attendra plus que d'être évaluée. En mode automatique, dès la fin d'écriture d'un nombre ou l'exécution d'un opérateur, l'opération/l'empilement est effectué.

Concernant l'UI, nous avons pris un soin tout particulier à développer un système d'écriture guidée, évitant à l'utilisateur d'écrire des nombres mal formés, et lui faisant gagner du temps. Concrètement, après avoir choisi dans quel mode il souhaite écrire, il n'a plus qu'à écrire les chiffres, et appuyer sur espace. En fonction de sa saisie, [espace] rajoutera un point, un slash, un dollar, voire une valeur par défaut (ainsi, en mode complexe rationnel, je peux écrire le nombre 15/1$0/1 en tapant simplement [15][espace][espace][espace]). Cependant, à des fins de confort d'utilisateur, la touche [point] et la touche [$] ont été définies comme des alias de [espace]. C'est aussi le cas de la touche [slash] quand on est en cours d'écriture d'un nombre rationnel (la même touche étant le raccourci de la division dans tous les autres cas).

Un problème est cependant apparu lors de la saisie des nombres négatifs. En effet, la touche [moins] est le raccourci de la soustraction. Dans les calculatrices classiques du type windows, la solution est simple : ils rajoutent un éventuel zéro en début de calcul, et considèrent que faire un nombre négatif revient bel et bien à faire une soustraction. Mais chez nous, le problème vient des nombres complexes, dont la structure "fermée" (le complexe tel que défini dans le sujet n'est pas considéré comme une somme de partie réelle et de partie complexe, mais comme un "tout"). Et la méthode "calculatrice windows" ne nous aurait pas permis de créer dès la saisie un nombre ayant une partie complexe de signe opposé à la partie imaginaire (il aurait fallu rentrer le nombre en deux fois, avec à chaque fois l'une des parties nulles). Nous nous sommes alors penché sur la manière dont les calculatrices scientifiques gèrent le problème. Malheureusement pour nous, elles implémentent un bouton spécifiques pour l'opérateur d'opposition. Nous avons donc envisagé de définir un bouton supplémentaire tenant ce rôle. Mais au moment de choisir le raccourci clavier, il nous est apparu qu'il n'était pas du tout intuitif d'utiliser autre chose que la touche [moins]. Et nous avons alors décidé, quitte à écrire en notation inverse, que le signe moins devrait attendre que l'écriture du nombre ait débutée pour être écrit. Ainsi, comme pour le [slash], le comportement de [moins] change lors de l'écriture d'un nombre. Ici, il aura pour effet de switcher le signe de la partie en cours d'écriture (ainsi en mode complexe entier, -15$-12 s'écrit simplement [15][moins][espace||dollar][12][moins]).

Enfin, pour nous assurer encore une fois de l'intégrité des données saisies, nous avons ajouté deux autre "protections" : lors de l'écriture d'un nombre, passée la partie entière réelle, il n'est plus possible de modifier le type du nombre tant que le calcul n'est pas fini. De même, pour toujours assurer à l'utilisateur un gain de temps et une excellente intégrité de saisie, le retour arrière a pour effet de supprimer un mot complet à chaque fois, évitant que les "EVAL" ne deviennent des "EVA", ou que le superbe "15/78$12/65" ne devienne un "15/78$1".

Dernière sécurité : si malgré tout un utilisateur ayant décidé de détourner les protections mises pour lui arrivait à soumettre une requête mal formée, notre implacable regexp ferait son devoir et annoncerait le problème.

Les petits plus

Outre ceux déjà annoncés, nous voulons mettre en avant quelques petits choix au service d'une meilleure calculatrice.

- notre empilement montre tous les termes empilés, pour être sûr que l'utilisateur ne perd jamais le fil. La calculatrice est d'ailleurs élégamment mettable en plein écran pour laisser à la pile la possibilité d'afficher en pleine vue même les nombreuses saisies ou les expressions concrètes longues. Si jamais vous deviez dépasser malgré tout en hauteur, la pile scrollera jusqu'au dernier nombre à chaque empilement.
- notre sauvegarde lors de la fermeture se souvient aussi des états précédents. Ainsi, lors de la réouverture de votre calculatrice, vous pourrez faire votre Ctrl-Z que vous aviez oublié de faire en allant chercher votre dernière édition de "Les Maths des Stars".
- le CLEAR (voir panneau Pile) en plus de vider la pile efface la sauvegarde. C'est bon, vous pouvez dormir tranquille, vos secrets seront illisibles ! Le raccourci (C) n'est pas effectuable depuis tous les panneaux pour éviter un appui malencontreux.
- dans la section trigonométrique, en plus de pouvoir choisir le mode de calcul degré ou radian, vous avez deux boutons pour vous permettre de faire les conversions que vous voulez. Ainsi qu'un nombre Pi. Bien que peu souvent utilisés, c'est 3 boutons ont aussi leur raccourcis clavier : pour RAD to DEG, c'est (1 2 PI * / 180 *), et DEG to RAD (1 180 / 2 PI * *). Pour PI, vous en avez une infinité en fonction du niveau de précision désiré. Par exemple, un simple [3] en mode entier peut parfois suffire. Sinon, un [3.14] ou [3.1415]. Ou encore un [3.14159265].
- lors de la saisie d'un trop grand ou trop petit nombre, il est automatiquement converti en notation scientifique. Après avoir combattu cet état de fait qui faisait planter la calculatrice lors de la recharge, nous avons finalement décidé d'améliorer encore plus notre calculette. Maintenant, la factory est capable de créer des nombres à partir d'une notation scientifique. Donc, n'ayez pas peur des grands nombres, ils seront sauvegardés comme les autres (NB : notez cependant que la précision maximale est dans tous les cas de 15 digits. Mais la puissance de 10 n'influera pas).

La conclusion

En conclusion, ce projet fut très formateur. Nous ne regrettons pas notre choix de conception pour ce projet scolaire, même s'il nous a conduit à faire une erreur lors des opérations de division complexe/complexe. C'est en faisant des erreurs qu'on apprend, et le moins qu'on puisse dire, c'est portant la barre à ce niveau, nous avons beaucoup appris ! Et nous avons donc fait de longues nuit de débugging... Nous avons aussi le plaisir de vous annoncer à l'occasion des commits finaux de la modélisation UML que nous avons franchi la barre des 100 commit, marque de notre véritable travail collaboratif.

Et maintenant, plus qu'une chose : dormir. Et réviser pour les finaux.

Merci !