Opera : un langage et un interpréteur pour effectuer des opérations arithmétiques mathématiquement exactes

Dans les applications informatiques habituelles, on utilise les nombres entiers et les flottants natifs qui sont de tailles fixes. Pour cette raison, la précision des calculs est limitée. En pratique, les approximations qui en résultent sont admises et suffisantes pour des calculs ordinaires en vie courante.

Pour effectuer des opérations arithmétiques exactes, les nombres utilisés par Opera sont représentés et mémorisés en tant que nombres rationnels, en fractions réduites, avec pour numérateurs et dénominateurs des entiers de tailles aussi grandes que nécessaires. Ce logiciel exécute les opérations à deux opérandes suivantes : élever à la puissance, diviser, multiplier, soustraire, additionner et comparer avec cet ordre spécifique de priorités. De plus, il reconnaît aussi automatiquement, d'après le contexte de son emploi, l'opérateur - ayant un seul opérande pour utiliser l'opposé du nombre désigné. Pour une expression arithmétique plus complexe l'emploi des parenthèses est reconnu et effectué. Les exponentiations qui sont irrationnelles sont détectées, approximées et signalées et celles qui sont impossibles sont signalées.

Le nom Opera de ce logiciel a pour origine une abréviation qui signifie : opérations arithmétiques mathématiquement exactes.

Opera a deux modes de fonctionnement : l'exécution d'un fichier de commandes et le mode conversationnel, ou interactif. L'interpréteur, dans les deux modes d'utilisation, effectue les calculs écrits en langage Opera. Il est disponible pour Windows et Linux et fonctionne dans une fenêtre en mode console.

Les bases du langage de programmation Opera

- Les constantes admises du langage sont soit des nombres entiers soit des nombres décimaux.
- Chaque variable représente un nombre rationnel ayant pour valeur une fraction réduite avec dénominateur positif. Les nombres entiers positifs ou négatifs sont reconnus : leurs dénominateurs sont égaux à 1.
- Toutes les variables sont du même type. Elles sont toutes locales dans le fichier de commandes où elles sont utilisées. Il n'existe pas de variables globales, ce qui facilite l'emploi des fichiers de commandes.
- Les variables qui sont des données ou des résultats en arguments d'un fichier de commandes sont indiquées et utilisées pour cela. La transmission se fait par copies.
- Un indice calculé avec un entier, exemple x[n], permet d'utiliser un tableau de variables.
- Un fichier de commandes peut en appeler un autre.
- Les branchements et débranchements conditionnels sont possibles.
- Les boucles itératives sont possibles, une boucle peut en contenir d'autres.
- On peut lire dans un fichier une suite de variables. On peut aussi sauvegarder toutes variables connues.
- Le caractère # débute un commentaire dans les fichiers de commandes.
- La variable last contient une copie du dernier résultat obtenu précédemment à son utilisation.
- Les fonctions reste, pgcd, ppcm, hasard, invmod, expmod, pnps, facteur et prem sont spécifiquement utilisables avec des variables ou des expressions ayant pour valeurs des entiers.

Un premier exemple

Voici un exemple simple d'utilisation de l'interpréteur Opera. Soit le fichier verif.txt suivant.

```
n = 366482613450380223856974383038858330841709595616663701388948644 73
p = 4103188409
q = 893165453106007309002842289513930850163184879519052329 7
prem p
prem q
v = p*q
valeur n-v
```

Si on lance Opera en mode conversationnel et si on écrit la commande : exec verif.txt on obtient un affichage qui montre que les nombres p et q sont premiers et que n et v sont égaux. En chiffrement RSA si on connait la clef publique d'un auteur pour découvrir sa clef privée il faut trouver les deux facteurs premiers d'un nombre entier ayant plusieurs centaines de chiffres décimaux. Opera n'a pas la possibilité de faire une telle décomposition, mais il peut très facilement en effectuer la vérification, si on le souhaite, c'est ce que fait cet exemple. En supplément, le fichier de commandes rsa.txt disponible dans la publication de Opera est un prototype de chiffrement RSA pour en montrer son utilisation dans l'envoi d'un document chiffré.

Les instructions du langage Opera

Les sept opérateurs binaires ont pour symboles : ^ / * - + < > et ils ont cet ordre de priorités. La division par 0 est possible. L'instruction i = 1/0 définit l'infini positif et j = -1/0 définit l'infini négatif. Voici deux expressions valides : 1/2*3/4 résultat : 3/8 et : 3*(8-2)^2 résultat : 108. L'ordre des priorités des opérateurs est important pour écrire, et pour lire, les expressions arithmétiques. Si l'expression comporte plusieurs fois le même opérateur le calcul est effectué de gauche vers la droite. À titre d'exemples, on peut noter que : 1+3*4 = 3*4+1 = 13 et : 3^3^3 = 27^3 = 19683 et non pas 3^27 = 7625597484987. L'utilisation des parenthèses permet de modifier l'ordre des calculs effectués : (1+3)*4 = 16 et : 3*(4+1) = 15.

L'exponentiation, exemple : x = a^n, a un fonctionnement particulier. En effet, si n = p/q est un nombre rationnel avec q > 1 il y a des cas où le résultat x est un nombre rationnel et d'autres cas où il est irrationnel, exemples : (4/9)^(3/2) = 8/27 et 2^(1/2) est irrationnel. Si le résultat est irrationnel, Opera calcule une approximation du résultat et affiche une indication de cette exception. Il y a une fonction qui permet de choisir le niveau de précision de cette approximation. De plus, quand le résultat n'est pas un nombre réel, exemple : (-1)^(1/2) un message signale ce cas éventuel. Enfin, l'utilisation sans précaution de l'exponentiation peut provoquer un encombrement de la mémoire centrale quand n et a sont très grands, ce qu'il faut éviter.

Le manuel de l'utilisateur explique les autres instructions du langage parmi les quelles on trouve, par exemples, les quatre suivantes. La commande "exec" (exemple : exec fichier.txt a, b, c) pour indiquer en existence et valeur les données et en existence les résultats d'un fichier de commandes que l'on souhaite lancer. La commande "boucle" notifie l'instruction qui la précède puis si last est positif continue en séquence et si non elle va chercher l'instruction qui suit l'instruction "retour" qui lui correspond. On peut imbriquer une "boucle" dans une autre. La commande "facteur" (exemple : facteur n) calcule un facteur premier d'une variable ou d'une expression entière qui a 16 ou 17 chiffres décimaux au maximum pour être performant. La commande "copier" (exemple : copier n,r) est nécessairement la première instruction effective d'un fichier de commandes. En complément de la commande "exec" elle désigne les variables en données et les variables en résultats de son fichier de commandes et la transmission se fait par copies.

La programmation en C++ de l'interpréteur Opera

Le code source de Opera est programmé en C++ et n'a qu'une seule dépendance : la bibliothèque GMP qui est disponible avec les autres fichiers *.cpp. Opera ne met rien dans le système d'exploitation, il n'a pas d'installeur, il est portable, on peut l'utiliser dans une clef USB.

Trois classes C++ particulières sont importantes. La classe bigRa effectue le traitement arithmétique des nombres rationnels ayant la taille aussi grande que nécessaire. La classe listera est une liste doublement chainée servant à gérer les variables utilisées pendant l'exécution du programme Opera. Quand un fichier de commandes en appelle un autre il faut conserver les variables du fichier appelant pendant l'exécution du fichier appelé. Pour cette raison, le nom des variables gérées par la classe listera comporte un préfixe défini par le niveau d'exécution du fichier de commandes en cours, ce qui assure l'utilisation locale de ses variables internes. Et la classe Trans gère par copies la transmission des variables entre le fichier de commandes appelant et le fichier de commandes appelé. Quand le fichier appelé est terminé ses variables sont supprimées.

Pour interpréter une expression arithmétique il est habituel d'utiliser l'intermédiaire de la notation polonaise inversée. Opera ne l'utilise pas, il effectue directement le calcul du résultat. Quand l'expression comporte des parenthèses il en vérifie l'équilibre et ensuite il remplace le contenu des parenthèses les plus internes par une variable temporaire calculée pour cela. Ceci permet, en utilisant la récursivité du C++, d'évaluer seulement des expressions sans parenthèse : pour 3*(8-2)^2 on calcule : a = 8-2 puis : 3*a^2. En fait, les noms de ces variables intermédiaires sont obtenus automatiquement, exemples : &t1, &t2, ..., et ne peuvent pas être en conflit de nom avec les variables de l'utilisateur.

Pour évaluer la valeur d'une expression sans parenthèse il faut respecter la priorité des opérateurs utilisés. Pour l'opérateur "-" ayant un seul opérande il y a deux traitements différents selon les cas d'utilisation. Dans le premier cas, exemples : -123 ou -v, on effectue le changement : 0-123 ou 0-v. Dans le deuxième cas, exemple : 3*-v on effectue le changement temporaire : 3*(-v) qui deviendra ensuite : 3*t en raison du traitement des parenthèses. Après cela, l'expression finale a seulement des opérateurs ayant deux opérandes.

D'autres exemples du langage Opera

Voici un programme en langage Opera qui vérifie la démonstration de Zagier qui prouve que le nombre 157 est congruent.

```
# Ce fichier de commandes montre que 157 est congruent.
# Un nombre entier n est congruent s'il existe 3 nombres
# rationnels a, b et c tels que a² + b² = c² et n = ab/2
# Voir : https://fr.wikipedia.org/wiki/Nombre_congruent
# Zagier a démontré que 157 est congruent.
# Voir : http://serge.mehl.free.fr/chrono/Zagier.html
# Le triplet (a, b, c) approprié pour n = 157 est :
a = 6803298487826435051217540/411340519227716149383203
b = 411340519227716149383203/21666555693714761309610
n = 224403517704336969924557513090674863160948472041
d = 8912332268928859588025535178967163570016480830
c = n/d
a2 = a*a
b2 = b*b
c2 = c*c
d2 = (a2 + b2) - c2
n = (a*b)/2
# Comme prévu on obtient  : d2 = 0 et n = 157
```

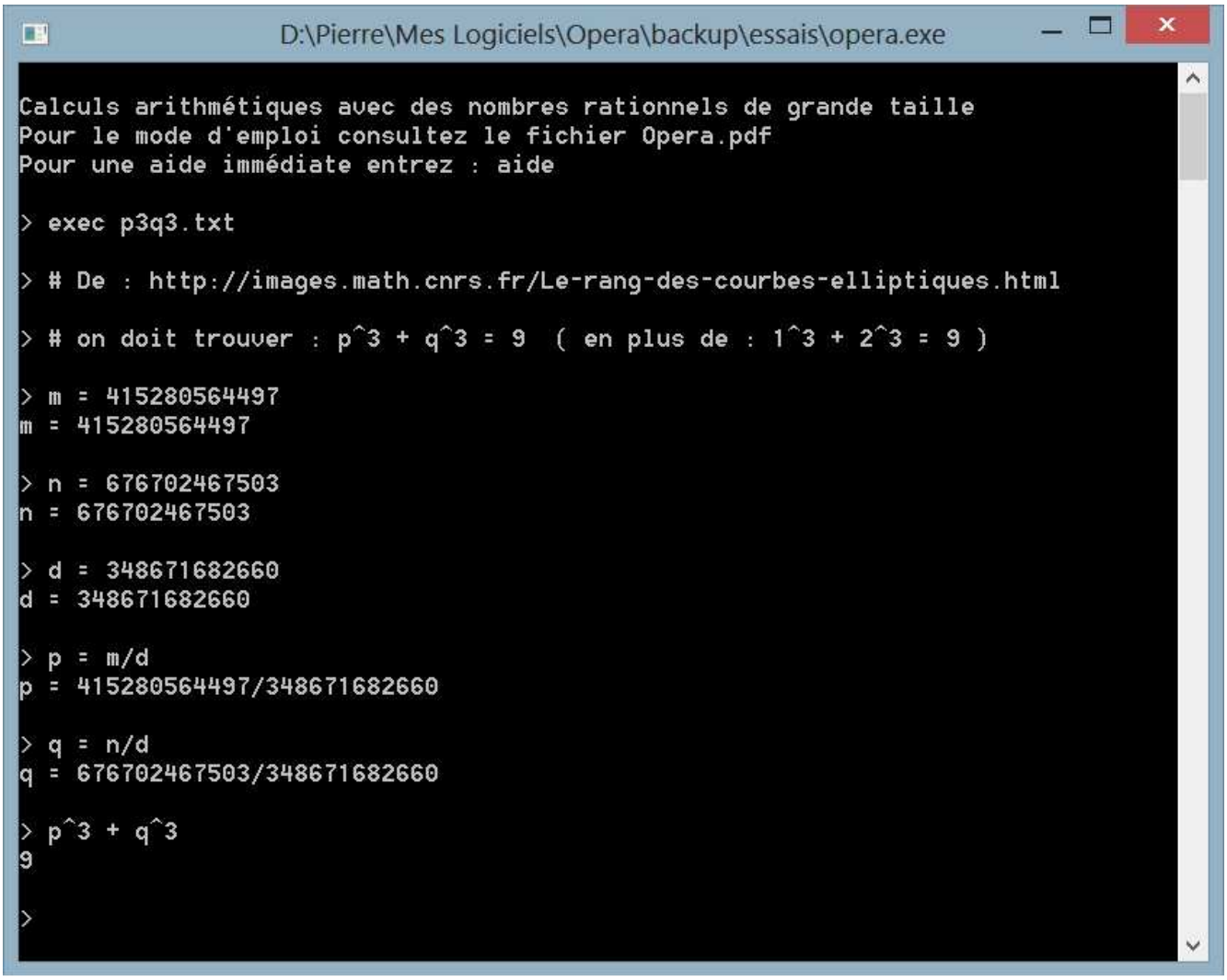
Voici, un programme écrit en langage Opera qui calcule une suite d'approximations du nombre pi avec une formule due à Jean-Henri Lambert jusqu'à obtenir n décimales exactes.

```
# Ce programme en langage Opera calcule une suite d'approximations de pi
# jusqu'à celle ayant c décimales exactes avec la formule suivante.
# pi = 4 / (1 + 1² / (3 + 2² / (5 + 3² / (7 + 4² / (9 + ... )))))
# Pour l'utiliser on peut écrire :
#    n = 16
#    exec mpi.txt n
# Ensuite on peut lire le fichier mpi-out.txt obtenu pour afficher
# toutes les variables utilisées où la dernière a la précision demandée.
# Mais on peut aussi écrire seulement :
#    lire mpi-out.txt
#    valeur f[i]
# Et on vérifie que l'approximation finale a 16 décimales exactes.
copier c
f[1] = 3
```

```
f[2] = 19/6
delta = f[2]-f[1]
i=2
delta > f[i]/10^c
boucle
   i = i + 1
   k = i
   d =  (2*k-1)+k^2/(2*k+1)
   k > 1
   boucle
      k = k-1
      d = (2*k-1)+k^2/d
   retour
   f[i] = 4/d
   delta = f[i]-f[i-1]
   si delta < 0
      delta = -delta
retour
enti f[i]*10^c
mpi = last
garder mpi-out.txt
```

f[22] = 51359350521741312/16348189019048887 est l'approximation finale du nombre pi obtenue avec le programme mpi.txt présenté ci-dessus. Pour 32 décimales exactes on aurait obtenu :
f[43] = 377007404451960464990387692952593891328/120005184001549872126533838222292079175

Voici une image de la console Windows avec un petit exemple d'utilisation du langage Opera.



Utilisations du langage Opera

Opera est public et gratuit. La publication comporte la documentation avec un manuel de l'utilisateur, le code source pour Windows et pour Linux, des commandes de compilation du code source, l'exécutable pour Windows et celui pour Linux et de nombreux exemples d'utilisation.

Opera est disponible à https://github.com/pgl10/Opera. Son exécutable pour Windows est obtenu avec Visual Studio et celui pour Linux est obtenu avec gcc. On peut aussi compiler le code source avec d'autres compilateurs C++ pour les mêmes operating systems ou bien pour d'autres.

Il faut noter que les espaces dans les instructions Opera ne sont pas effectifs sauf, bien sûr, dans les commentaires. Ce qui permet d'écrire aussi bien : n = 3 + 4*5 que : n=3+4*5. En effet, au début de l'interprétation de l'instruction les espaces sont supprimés, sauf dans les commentaires. Dans un fichier de commandes l'indentation des instructions n'est pas indispensable, mais c'est une disposition commode. De plus, pour lire et effectuer la dernière ligne effective d'un fichier de commandes il est nécessaire d'ajouter ensuite une ligne vide. Les exemples ci-joints montrent cette nécessité.

Le langage Opera a trois instructions pour effectuer des entrées ou des sorties. La commande "noter" (exemple : noter fichier.txt) permet de noter dans un fichier toutes les variables actuelles. Si le fichier désigné existe déjà l'archivage est refusé. La commande "garder" (exemple : garder fichier.txt) permet de garder dans un fichier toutes les variables actuelles. Si le fichier désigné existe déjà il est remplacé. La commande "lire" (exemple : lire fichier.txt) permet de créer au niveau actuel les variables définies dans ce fichier, ce qui est très commode quand on souhaite faire divers essais avec les mêmes nombres.

Opera a divers concurrents. Le premier est le langage C++ lui-même avec la bibliothèque GMP où se trouvent les entités mpq_t. Il y a aussi les divers langages formels, exemples : Maple, MatLab ou Eigenmath. Opera est moins puissant que ceux-ci mais dans bien des cas il est plus commode. Et le code source de Opera comporte des éléments pouvant servir à d'autres développements analogues.