Opera : des opérations arithmétiques exactes

Manuel de l'utilisateur

Plan du manuel :

1°) Introduction

Opera est un interpréteur qui effectue des opérations arithmétiques mathématiquement exactes. Dans les applications informatiques habituelles on utilise les 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 avec pour numérateurs et dénominateurs des entiers de tailles aussi grandes que nécessaires.

Opera a deux modes de fonctionnement : l'exécution d'un fichier de commandes et le mode conversationnel. C'est un programme interactif qui lit, interprète, calcule et archive les résultats des instructions écrites

selon les règles prévues. Le logiciel Opera effectue les calculs programmés en langage Opera.

Le logiciel est compilé par MicroSoft Visual Studio 2012 Express for Windows Desktop en mode Release et configuration Win32(x86) avec l'option /MT : cela ne nécessite aucune dll complémentaire. Il fonctionne dans une fenêtre de Windows en mode console. Le source développé en C++ est entièrement disponible. Il n'a qu'une seule dépendance fournie avec le source : GMP en variante développée par Paul Herman. Il y a deux structures principales. La première est la classe bigRa qui définit un rationnel avec deux entiers sans limite de grandeur : le numérateur et le dénominateur. La seconde est listera qui est une liste simplement chainée servant à gérer les rationnels utilisés pendant l'exécution du programme.

Le programme Opera ne met rien dans le système Windows ni dans la base de registre, il n'a pas d'installeur et il est portable. On peut l'utiliser avec une clef USB. Il est public et gratuit, son adresse Internet est : https://github.com/pgl10/Opera. La disponibilité du source permet de comprendre son fonctionnement et de faire assez facilement les corrections, les modifications et les améliorations que l'on souhaite.

2°) Les opérateurs et les opérandes

Les nombres rationnels utilisés sont définis par une variable ayant un nom et une valeur. Le nom commence par une lettre et comporte seulement les caractères : lettres et chiffres auxquels on joint le caractère _ pour pouvoir indiquer un nom composé de plusieurs mots. Les lettres admises sont les 26 lettres minuscules et majuscules de l'alphabet seulement. Ainsi, par exemple : Paris_Brest est un nom reconnu et utilisable. La valeur de chaque nombre rationnel est stockée en mémoire par deux entiers : le numérateur et le dénominateur aussi grands que nécessaires et ceux-ci sont réduits automatiquement sans aucun diviseur commun. Qui peut le plus, peut le moins : les calculs

avec les nombres entiers sont possibles aussi et de même pour les calculs qui utilisent à la fois les entiers et les rationnels.

Pendant son exécution le programme gère les variables qui sont activées. Il n'est pas possible de modifier directement la valeur d'une variable. Si on veut calculer et archiver plusieurs fois de suite la même variable il faut la supprimer avant chaque nouvel archivage. Mais on peut aussi en calculer plusieurs versions avec un suffixe servant de numéro d'ordre ou d'indice : ceci peut être effectué en nommant successivement les variables de cette suite, exemple : v1, v2, ... Contrairement à la phase initiale si on veut quand même pouvoir redéfinir directement une variable sans avoir la nécessité de la supprimer préalablement on peut utiliser l'instruction : redef oui et l'instruction : redef non permet de rétablir le fonctionnement initial. Cette commodité doit être utilisée avec précaution. De plus, on peut aussi renommer une variable à l'aide de la valeur actuelle entière positive ou nulle d'une autre variable connue, exemple : si n = 5 la variable x[n] est automatiquement renommée x_5 et de même pour x[5] ou x[i+2] avec i=3. Ce suffixe calculé est particulièrement utile dans les fichiers de commandes. Quand la session est terminée les variables encore actives sont perdues, mais on peut avant cela les sauvegarder dans un fichier et les recharger ensuite.

Opera reconnaît 7 opérateurs binaires ayant 2 opérandes et un opérateur ayant un seul opérande, c'est l'opérateur - qui permet d'utiliser directement la valeur opposée de son argument. Les sept opérateurs binaires sont : ^ / * - + < > et ils ont cet ordre de priorités. Ils permettent d'élever à la puissance, de diviser, de multiplier, de soustraire, d'additionner et de comparer. Les deux opérateurs - sont reconnus automatiquement d'après le contexte où ils sont employés. Quand l'expression arithmétique comporte plusieurs fois le même opérateur binaire le calcul est effectué de gauche vers la droite. Les opérandes sont soit des entiers soit des nombres décimaux soit des variables contenant un nombre rationnel pouvant être fractionnaire ou

entier et soit une expression valide entre parenthèses. Voici des exemples commentés d'expressions arithmétiques valides pour Opera :

> v pour afficher la valeur actuelle de la variable v

> 2+3/4 pour afficher la valeur de cette expression arithmétique

> a = 3/7 pour créer la variable a avec cette fraction

> p = 3.14 pour créer la variable p avec ce nombre décimal

> x = a^2 calcul et archivage de a^2

> y = b^n la valeur de l'exposant est un entier ou un rationnel

> z = a>b résultat : 1 si a>b, 0 si a=b et -1 si a<b

> t = a<b résultat : 1 si a<b, 0 si a=b et -1 si a>b

> -5--7 résultat : 2 (équivalent à : (-5)-(-7))

> 1/2*3/4 résultat : 3/8 (équivalent à : (1/2)*(3/4))

> 3*(8-2)/2 résultat : 9 (3*6/2 = 3*3 = 9)

> (a+5)/(n+1) ici la division est effectuée après les additions

Opera permet l'emploi de parenthèses pour préciser ou modifier l'ordre des opérations effectuées. Dans l'instruction x = a+b*c on effectue b*c en premier. Si on veut effectuer a+b en premier il faut faire x = (a+b)*c et on peut avoir des parenthèses sur plusieurs niveaux imbriqués. Les nombres décimaux utilisables ont un . et un seul qui n'est ni au début ni à la fin : 3.14 ou 3.0 ou 0.14 sont valides mais 3. et .14 sont invalides.

La division par 0 est possible. L'instruction i = 1/0 définit l'infini positif et j = -1/0 définit l'infini négatif. Il faut aussi noter qu'on a choisi arbitrairement que 0 est le résultat de l'opération 0/0 et celui de la partie entière de 1/0 : ceci est effectué sans aucun message d'accompagnement.

3°) L'exponentiation

L'exponentiation x = a^r nécessite quelques explications spécifiques. Si r est un entier ou une expression à valeur entière il n'y a rien de plus à préciser. Mais si r = 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 mais 2^(1/2) est irrationnel. Si le résultat est rationnel Opera pourra le calculer. Par contre, si le résultat est irrationnel il n'est pas possible d'obtenir un résultat exact. Dans ce cas, Opera va calculer un nombre rationnel qui sera une approximation du résultat en utilisant la fonction nroot() de la bibliothèque bigRa et un message spécifique sera affiché. Il convient d'utiliser ensuite l'approximation obtenue avec précaution parce que les opérations arithmétiques effectuées par Opera sont toujours exactes, sauf dans ce cas particulier.

Si a est négatif et si q est pair a^(1/q) est impossible puisqu'aucun nombre réel élevé à une puissance paire fournit un résultat négatif. Un message signale ce cas éventuel.

La fonction bigRa::puissance() permet d'exécuter x.puissance(n) pour tout x et tout n de type int de -INT_MAX à INT_MAX. Cela peut créer un encombrement de la mémoire centrale si x et n sont grands. On pourrait envisager d'interrompre automatiquement cette fonction en cas d'encombrement de la mémoire centrale par la taille du résultat intermédiaire obtenu pendant ce calcul avec un message d'annulation de l'instruction en cours. En l'état actuel cette précaution n'existe pas : le plantage de l'application peut survenir pour cette raison, mais il faut pour cela des valeurs exceptionnellement grandes.

4°) Autres instructions

En complément des instructions de base déjà expliquées il y a aussi les instructions suivantes :

> ilyatil v résultat : 1 si la variable v existe et 0 si non

> del var pour supprimer la variable var

> out sauv.txt pour sauvegarder toutes les variables actuelles

> exec fic.txt pour exécuter le fichier de commandes fic.txt

> convrt vr pour convertir et afficher en nombre réel la variable vr

> convrt x/123 pour convertir en réel une expression arithmétique

> nbch vr nombre de chiffres décimaux de la partie entière de vr

> enti vr pour calculer et afficher la partie entière de la variable vr

> enti x*100000 la partie entière de cette expression arithmétique

> frac vr la partie fractionnaire de vr ou d'une expression

> num vr le numérateur de la variable vr ou d'une expression

> den vr le dénominateur de la variable vr ou d'une expression

> pgcd e1,e2 le pgcd de deux expressions à valeurs entières

> ppcm e1,e2 le ppcm de deux expressions à valeurs entières

> prem e résultat : 1 si e est premier et 0 si non

> ndiv e un facteur premier d'une expression à valeur entière

> lister pour afficher toutes les variables actuelles

> aide pour afficher un petit rappel des instructions utilisables

> exit pour terminer la session

La commande convrt affiche en forme usuelle un nombre réel qui donne une bonne valeur approchée d'une variable ou d'une expression arithmétique. Si on veut voir un grand nombre de chiffres significatifs exacts obtenus il faut utiliser en complément la commande enti qui calcule et affiche la partie entière d'une variable ou d'une expression arithmétique. Selon les cas on pourra faire par exemples : enti v*1000

ou enti v*1000000 etc. Ce qui permet facilement d'afficher autant de chiffres significatifs que l'on veut.

Quand on calcule ou que l'on affiche une variable ou une expression il y a une variable particulière nommée last qui contient une copie du résultat. Il en est de même quand on affiche la valeur réelle d'une variable ou d'une expression avec la commande convrt. Et les commandes ilyatil, nbch, enti, frac, num, den, pgcd, ppcm, prem et ndiv ont aussi la même propriété avec la variable last. Si a et d sont deux variables ayant pour valeurs deux entiers, pour obtenir le quotient faire : enti a/d et : q = last et pour le reste : r = a-q*d.

La commande prem utilise la méthode de Miller et Rabin incluse dans GMP. Et la commande ndiv utilise la méthode de l'algorithme rho de John M. Pollard qui est performante pour trouver un petit facteur premier. Il est possible mais extrêmement rare que l'algorithme rho soit en échec et dans ce cas le résultat est nul. Avec cette méthode simple qui a une durée d'exécution imprévisible il faut éviter une recherche très difficile : pour information, le 7-ième nombre de Fermat 2^128 + 1 est le produit de 2 nombres premiers ayant 17 et 22 chiffres décimaux et sur un PC avec un Intel i7 sa factorisation a duré 43 minutes. En pratique, il est préférable d'éviter de dépasser environ 17 chiffres décimaux pour le plus petit facteur premier à rechercher.

5°) Les fichiers de commandes

Le mode fichier de commandes est utilisable au début de la session par l'intermédiaire d'un fichier batch qui désigne le fichier des instructions à exécuter. Le mode conversationnel est automatiquement utilisé ensuite jusqu'à la fin de la session. Si on exécute seulement le fichier opera.exe on commence directement en phase conversationnelle. L'utilisation d'un fichier de commandes pendant la phase conversationnelle est tout à fait possible aussi.

Les lignes de commentaires commencent par # et tous les caractères y sont acceptés. Les commentaires en fin de ligne sont admis. Un fichier de commandes peut en appeler un autre. Les noms des variables internes au fichier de commandes ne doivent pas être en conflit avec les noms des variables actives au moment de l'appel. Les débranchements conditionnels et les boucles itératives permettent d'effectuer de véritables petits programmes de calculs arithmétiques exacts. Pour assurer la lecture correcte de la dernière ligne utile d'un fichier commandes il convient de la faire suivre d'une et une seule ligne vide.

La commande boucle notifie l'instruction qui la précède puis si last est positive continue en séquence et si non elle va chercher l'instruction qui suit l'instruction retour qui lui correspond. Et la commande retour revient à l'instruction notifiée qui précède la commande boucle qui lui correspond. On peut effectuer une boucle imbriquée dans une autre.

La commande quitter effectue l'abandon du fichier de commandes si la variable last est positive et continue en séquence si non. La commande continuer continue en séquence si la variable last est positive et effectue l'abandon du fichier de commandes si non.

Toutes les variables sont globales. Il n'y a aucun dispositif pour distinguer les variables internes du fichier de commandes relativement aux variables externes et de plus, les variables pouvant être des données ou bien des résultats du fichier de commandes en cours ne sont pas spécifiées pour cela. Il en résulte la nécessité de veiller à éviter un éventuel conflit de nom entre les variables internes au fichier de commandes et les autres variables actives au moment de son appel. L'utilisation de la commande redef oui est une commodité parfois nécessaire. Si on autorise la redéfinition des variables, une variable interne peut remplacer une autre variable qui est donc perdue dans ce cas. Si on n'autorise pas la redéfinition, une variable externe peut

empêcher l'exécution correcte du fichier de commandes ayant une variable interne du même nom.

Les exemples joints montrent divers fichiers de commandes.

6°) Exemples, trucs et astuces

Plusieurs exemples de fichiers de commandes disponibles avec la publication de Opera sont joints avec leurs fichiers .bat correspondants qui permettent de lancer Opera en mode fichier de commandes dès la phase initiale et d'autres exemples sont prévus pour être utilisés pendant la phase conversationnelle. La lecture des exemples avec les commentaires inclus et leur utilisation montrent les fonctions actuellement disponibles avec Opera, en particulier la nécessité d'utiliser des nombres rationnels de tailles aussi grandes que nécessaires dans certaines circonstances mais pas uniquement : il y a aussi des calculs avec de grands entiers et d'autres calculs plus simples qui sont faisables facilement.

Voici en complément quelques astuces de programmation :

# Pour calculer abs : # la valeur absolue de x redef oui abs = x abs < 0 boucle abs = -x retour	# Pour calculer min : # le minimum de x et y redef oui min = x y < min boucle min = y retour
# Pour vérifier que n est entier frac n 1+(last<0)*(last>0) continuer	# Pour vérifier que a = b a < b 1+(last<0)*(last>0) continuer

7°) Compléments

Si vous avez téléchargé à : https://github.com/pgl10/Opera le fichier Opera-master.zip après avoir cliqué sur "Clone or download" vous pouvez dézipper l'ensemble des fichiers disponibles, mais sous Windows les nombreux fichiers de type .bat .txt .cpp ou .hpp ne sont plus utilisables directement avec le Block-note Notepad. Pour rétablir cette utilisation habituelle il faut les ouvrir avec Wordpad et faire "Enregistrer". Ceci est dû à l'utilisation de GitHub et du logiciel Git développé par Linus Torvalds. Il est particulièrement nécessaire de faire cela pour les fichiers de commandes. L'encodage ANSI des fichiers de commandes est indispensable pour les lire correctement, l'encodage Unicode n'est pas reconnu dans la version Windows.

Il est souvent commode d'utiliser la fenêtre de la console Windows avec une hauteur assez grande. On peut l'obtenir avec la souris sur le bord inférieur ou bien avec le bouton <Agrandir>. Il y a une autre méthode plus complète pour l'effectuer : souris sur le bord supérieur et clic <droite> => Propriétés => Configuration => Taille de la fenêtre, ce qui permet aussi de faire d'autres réglages et de plus : Windows conservera ces réglages pour la prochaine fois, mais cela oblige à rétablir les réglages qu'on a modifiés si on ne veut pas les garder.

Il existe d'autres logiciels qui utilisent les nombres rationnels de tailles aussi grandes que nécessaires et il existe aussi des logiciels beaucoup plus puissants que Opera, par exemples : Maple de l'entreprise canadienne Maplesoft ou bien Eigenmath de George Weigt. Il est bien peu constructif de vouloir en faire des comparaisons par contre, pouvoir examiner les constituants disponibles de Opera peut intéresser certains utilisateurs ou programmeurs scientifiques.

8°) Linux

La version Linux de Opera est presque identique à la version Windows. Les deux changements principaux sont le remplacement du retour à la ligne Windows(CR-LF) par Unix(LF) et celui de l'encodage ANSI par UTF-8(sans BOM). Diverses solutions sont disponibles pour effectuer ces deux changements, par exemple Notepap2 sous Windows. Les fichiers .bat sont remplacés par leurs équivalents .sh. La nouvelle fonction pause() concerne le fichier main.cpp et la nouvelle fonction aout() concerne les fichiers aout.cpp, eval.cpp, listera.cpp et main.cpp. Dans le source on a indiqué comment installer la bibliothèque GMP et on a mis le fichier Makefile pour pouvoir refaire la compilation.

Le sous-répertoire src contient le source et Makefile. Les exemples et l'exécutable obtenu sur Linux Mint sont dans le sous-répertoire exe.