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, en fractions réduites, 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.

Il est disponible pour Windows et pour Linux. Compilé par MicroSoft Visual Studio 2012 Express for Windows Desktop en mode Release et configuration Win32(x86) avec l'option /MT : il 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 doublement 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 instructions arithmétiques sont écrites en utilisant des constantes, des variables et des opérateurs avec ou sans parenthèses pour préciser l'ordre des opérations à effectuer. Les seules constantes reconnues par Opera sont les nombres entiers et les nombres décimaux. 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.

Chaque variable est définie par son niveau d'exécution, son nom et sa valeur. En mode conversationnel on est au niveau 0 d'exécution. Dans les fichiers de commandes lancés en mode conversationnel on est au niveau 1 d'exécution. Les fichiers de commandes du niveau n+1 sont ceux qui sont lancés par un fichier de commandes du niveau n. Le niveau maximum utilisable est 9. Ceci permet l'existence sans conflit de variables différentes ayant le même nom dans plusieurs niveaux d'exécution. En pratique, la liste des variables connues se comporte comme plusieurs sous-listes de variables : une pour chaque niveau d'exécution. Dans un fichier de commandes les variables connues, crées ou mises à jour, sont toutes locales et du niveau d'exécution du fichier de commandes lui-même. L'utilisation de variables globales, déconseillée en programmation, est donc impossible.

Le nom de chaque variable commence par une lettre et comporte 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. Quand on souhaite utiliser plusieurs noms voisins on peut terminer le nom par un suffixe : v, v2, v_2 et v[2] sont des noms valides. Une variable ayant un nom indicé, exemple v[2], utilise un complément de nom entre crochets qui est soit un entier positif ou nul soit une expression à valeur positive ou nulle qui est évaluée au moment où l'instruction est effectuée, exemple : si n est un entier qui vaut 3 l'instruction v[2*n+1] = n+8 aura pour résultat v[7] = 11. Une variable ayant un nom indicé peut avoir plusieurs indices. L'emploi des noms indicés est particulièrement utile dans les boucles itératives. Quand l'exécution d'un fichier de commandes est terminée ou abandonnée toutes ses variables locales sont supprimées.

La valeur de chaque variable est définie par un nombre rationnel de la classe bigRa : le numérateur et le dénominateur n'ont aucun facteur premier commun. Sauf cas exceptionnel, le dénominateur est positif et

vaut 1 pour une variable à valeur entière. La variante de GMP développée par Paul Herman facilite la programmation de la classe bigRa.

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

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.

Chaque opérande est soit un entier soit un nombre décimal soit une variable et soit une expression arithmétique valide entre parenthèses.

Il existe une variable particulière nommée last qui contient une copie du résultat calculé ou même affiché précédemment.

Quelques exemples commentés d'expressions arithmétiques valides :

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

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

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

> 3^3^3 résultat : 19683 (3^3^3 = 27^3 = 19683)

3°) L'exponentiation

L'exponentiation x = a^b nécessite quelques explications spécifiques. Si b est un entier ou une expression à valeur entière il n'y a rien de plus à préciser. Mais si b = p/q avec q > 1 est un nombre rationnel 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 rationnel Opera pourra le calculer. Par contre, si le résultat est irrationnel Opera va calculer un nombre rationnel qui en sera une approximation. Dans une version précédente on avait utilisé pour cela les fonctions root() et dbl2ra() de la bibliothèque bigRa. On utilise maintenant la fonction nroot() qui est plus générale. Avec l'entier n quand x^(1/n) est un nombre irrationnel le calcul de son approximation r est effectué par dichotomie jusqu'à la condition : |(x-r^n)/x| < 10^(-64). Attention : si le résultat est irrationnel l'approximation obtenue n'est pas un résultat exact contrairement aux autres opérations arithmétiques effectuées par Opera. Il convient d'en tenir compte.

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

La fonction bigRa::puissance() permet d'effectuer 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 du langage Opera

4-1) La commande "exec", exemple : exec fic.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.

4-2) La commande "copier", exemple : copier x, y, z

Pour créer par copies les variables en arguments d'un fichier de commandes que l'on vient de lancer. C'est nécessairement la première instruction effective du fichier. On peut seulement écrire des lignes éventuelles de commentaires avant cette première instruction.

4-3) La commande "renvoyer", exemple : renvoyer z

Pour mettre à jour, dans le fichier de commandes appelant, une variable de type résultat dont la variable associée z dans le fichier de commandes appelé vient d'être calculée.

4-4) La commande "prochain", exemple prochain r

Pour créer, dans le fichier de commandes appelant, une variable indicée ayant la valeur actuelle de last et le nom associé à l'argument r avec l'indice associé à r incrémenté à chaque utilisation. Cette commande permet d'obtenir une suite de résultats.

4-5) La commande "recevoir", exemple : recevoir x, r

Pour obtenir dans la variable locale x une copie la variable r du niveau 0 d'exécution.

4-6) La commande "envoyer", exemple : envoyer x, r

Pour créer au niveau 0 d'exécution la variable r par copie de la variable locale x. Si r existe déjà on crée r[2] et si r[2] existe déjà on crée r[3] ...

4-7) La commande "supprimer", exemple : supprimer v

Pour supprimer la variable locale v.

4-8) La commande "aide"

Pour un rappel des commandes disponibles de Opera.

4-9) La commande "pause"

Pour stopper temporairement le défilement des affichages obtenus pendant l'exécution d'un fichier de commandes. Faire <return> pour continuer.

4-10) La commande "lister"

Cette commande permet d'afficher toutes les variables locales actuelles du niveau actuel.

4-11) La commande "noter", exemple : noter mesnotes.txt

Cette commande permet de noter dans un fichier toutes les variables actuelles d'un fichier de commandes. Si le fichier désigné existe déjà l'archivage est refusé.

4-12) La commande "garder", exemple : garder monarchive.txt

Cette commande permet de garder dans un fichier toutes les variables actuelles d'un fichier de commandes. Si le fichier désigné existe déjà il est remplacé.

4-13) La commande "lire", exemple : lire monfichier.txt

Cette commande permet de restituer au niveau actuel toutes les variables disponibles dans un fichier obtenu avec "noter" ou "garder", ou bien d'entrer facilement des données que l'on souhaite examiner.

4-14) La commande "valeur", exemple : valeur expr

Pour un affichage traditionnel, mais limité, de la valeur d'une variable ou d'une expression arithmétique.

4-15) La commande "nbch", exemple : nbch expr

Pour afficher le nombre de chiffres décimaux de la partie entière d'une expression arithmétique.

4-16) La commande "enti", exemple : enti expr

Pour calculer la partie entière d'une expression arithmétique.

4-17) La commande "frac", exemple : frac expr

Pour calculer la partie fractionnaire d'une expression arithmétique.

4-18) La commande "num", exemple : num expr

Pour calculer le numérateur d'une expression arithmétique.

4-19) La commande "den", exemple : den expr

Pour calculer le dénominateur d'une expression arithmétique.

4-20) La commande "continuer"

Pour continuer si last est positif et abandonner si non.

4-21) La commande "quitter"

Pour quitter si last est positif et continuer si non.

4-22) La commande "si", exemple : si expr

Pour aller en séquence si l'expression est positive et pour sauter une instruction si l'expression est nulle ou négative.

4-23) La commande "boucle"

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.

4-24) La commande "retour"

La commande retour revient à l'instruction notifiée qui précède la commande boucle qui lui correspond.

4-25) La commande "exit"

La commande exit termine la session en cours d'exécution.

4-26) La commande "pgcd", exemple : pgcd m, n

Pour calculer le pgcd de deux variables ou expressions arithmétiques ayant pour valeurs deux entiers.

4-27) La commande "ppcm", exemple : ppcm m, n

Pour calculer le ppcm de deux variables ou expressions arithmétiques ayant pour valeurs deux entiers.

4-28) La commande "facteur", exemple : facteur n

Pour calculer un facteur premier d'une variable ou d'une expression arithmétique ayant pour valeur un entier. On utilise l'algorithme rho de John M. Pollard qui est performant pour trouver un petit facteur premier et qui est très rarement en échec. En pratique, il est préférable d'éviter de dépasser environ 16 ou 17 chiffres décimaux pour le facteur premier à rechercher.

4-29) La commande "prem", exemple : prem n

Pour calculer la primalité d'une variable ou d'une expression arithmétique ayant pour valeur un entier. On utilise la méthode de Miller et Rabin incluse dans GMP.

La variable last est automatiquement mise à jour avec le résultat calculé ou affiché d'une expression arithmétique, elle est redéfinie aussi quand on emploie les commandes : valeur, nbch, enti, frac, num, den, pgcd, ppcm, facteur et prem. 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.

En mode conversationnel les commandes suivantes n'effectuent aucune action : copier, renvoyer, prochain, recevoir, envoyer, pause, continuer, quitter, si, boucle et retour.

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 initiale ou pendant la phase conversationnelle est tout à fait possible aussi. Un fichier de commandes peut en appeler un autre.

Les lignes de commentaires commencent par # et tous les caractères y sont acceptés. Les commentaires en fin de ligne sont admis.

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.

5°) 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.

Quelques astuces de programmation :

# Pour calculer max : 	
# le maximum de x et y 	
max = x
si y > x
max = y
	# La parité p de l'entier n
	p = 2
	den n/2
	si last > 1
	p = 1

# 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

6°) 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 .txt ne sont plus utilisables directement avec le Block-note Notepad. Pour rétablir cette utilisation habituelle il faut ouvrir les fichiers de type .txt avec Wordpad et faire "Enregistrer". Il est conseillé et parfois nécessaire de faire cela pour les fichiers de commandes. L'encodage ANSI des fichiers de commandes est nécessaire pour les lire correctement.

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.

7°) 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 Notepad2 sous Windows. Les fichiers .bat sont remplacés par leurs équivalents .sh. Les deux seules fonctions C++ différentes sous Linux sont : pause() et aout(). 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. Le sous-répertoire exe contient les exemples et l'exécutable obtenu avec Linux Mint.