Définition d'une classe "grand entier"

Mohamed Lokbani

LA DATE LIMITE EST LE **12 Novembre 1999 MINUIT**, avec une pénalité de 2 points par jours de retard.

À faire seul ou en équipe de deux.

BUT

Ce TP a pour but de vous faire pratiquer les matières suivantes :

- * Programmation modulaire procédurale en C++
- * Les notions de C++ (classes, constructeurs/destructeurs, surcharges des opérateurs, fonctions amies etc.)

Lisez l'énoncé de A à Z, puis en cas de problèmes contactez nous.

1. Présentation du problème

Les opérations arithmétiques, + (addition), - (soustraction), * (multiplication), / (division) etc. sont des opérations fondamentales en informatique. Dans de nombreux langages de programmation, la plus grande valeur d'une variable entière est souvent beaucoup trop petite pour représenter de grandes valeurs, exemple: la population mondiale ; le nombre d'occurrences de la lettre "e" dans l'œuvre complète d'Émile Zola ; en cryptographie où l'utilisation de grandes valeurs est nécessaire pour crypter une clef ou la vérifier.

Une variable positive entière représentée sur 16 bits (2 octets) peut prendre la valeur maximale de 32768. Sur 32 bits (4 octets), la valeur maximale est de : 2,147,483,647.

Même avec une représentation entière sur 4 octets, cette dernière reste trop petite pour les applications citées plus haut.

Exemple:

```
//exemple sur l'utilisation des entiers
```

```
/u/dift1166/Sujet/tp2/prg/limit_int.cpp
```

le programme exécutable est lui aussi disponible dans le même répertoire, i.e.:

```
/u/dift1166/Sujet/tp2/prg/limit_int.cpp
```

```

-----
#include <iostream>
#include <math.h>

int main() {
    int total, itt;

    unsigned int valeur;

    for (itt=0;itt<10;itt++) {
        total = (int) pow(itt,10);
        cout << "Pour une valeur = " << itt << " ; sa puissance par 10 vaut: " << total << endl;
    }

    cout << "la valeur réelle de 9 puissance 10 est .... " <<endl;
    valeur = (unsigned int) pow(9,10);
    cout << "valeur: " << valeur << endl;

    cout << "la valeur maximale permise pour un entier sur 4 octets est .... " <<endl;

    total = (int) (pow(2,31)-1.0);
    cout << "valeur: " << total << endl;

    return 0;
}

```

```

// sortie du programme ....
/*
Pour une valeur = 0 ; sa puissance par 10 vaut: 0
Pour une valeur = 1 ; sa puissance par 10 vaut: 1
Pour une valeur = 2 ; sa puissance par 10 vaut: 1024
Pour une valeur = 3 ; sa puissance par 10 vaut: 59049
Pour une valeur = 4 ; sa puissance par 10 vaut: 1048576
Pour une valeur = 5 ; sa puissance par 10 vaut: 9765625
Pour une valeur = 6 ; sa puissance par 10 vaut: 60466176
Pour une valeur = 7 ; sa puissance par 10 vaut: 282475249
Pour une valeur = 8 ; sa puissance par 10 vaut: 1073741824
Pour une valeur = 9 ; sa puissance par 10 vaut: -2147483648
la valeur réelle de 9 puissance 10 est ....
valeur: 3486784401
la valeur maximale permise pour un entier sur 4 octets est ....
valeur: 2147483647
*/

```

la valeur réelle devant être retournée pour 9 puissance 10 est égale à:

3486784401

la valeur maximale autorisée pour un entier signé sur 4 octets est de:

2147483647

la valeur de 9 puissance 10 est donc supérieure à la valeur maximale autorisée, d'où l'apparition du signe négatif.

2. Specification de GrdEnt

GrdEnt est une abréviation à : Grand Entier

2.1 Objectif du tp

L'objectif de ce tp, est de réaliser la classe grand entier et les opérations associées permettant de manipuler des entiers de taille quelconque. Exemple, l'addition de ces deux grands entiers:

```
7437380164651435143865874852299929767 +  
743738016465143514386587485229976589067546654695469845
```

2.2. Dimension du grand entier

Par dimension, on entend le nombre de chiffres que peut avoir un grand entier. Dans notre tp2, aucune restriction ne sera imposée sur la taille des grands entiers. La seule limite, est celle imposée par la capacité mémoire de la machine.

2.3. Opérations sur les grands entiers

Les méthodes à implanter sont:

Fondamentales

constructeurs

Entrees/sorties

operator<<
operator>>

Arithmetiques

operator+=
operator-=
operator=
operator+
operator-
operator**

Comparaisons

operator<
*operator<=
operator>
operator>=
operator==
operator!=*

2.4. Gestion des erreurs

Les erreurs doivent être capturées, un message d'erreur doit s'afficher à l'écran. Par la suite il faut sortir du programme. Exemples d'erreur: lorsqu'une personne entre une valeur alphanumérique: 24356tre6743643.

3. Spécification de la classe GrdEnt

La classe GrdEnt (Grand Entier) permet de représenter des nombres entiers (positifs, nuls, ou négatifs) d'une taille quelconque. La classe GrdEnt est munie de quelques fonctions et opérateurs permettant de faire certains calculs avec ces grands entiers.

La classe Représentation, dont l'interface est décrite plus bas dans le document, permet de gérer un tableau dynamique de caractères. Elle nous sert donc à représenter les chiffres d'un grand entier. Les chiffres sont représentés en base 10. Ainsi, le chiffre 6475 sera représenté comme suit:

le chiffre de droite au rang 0
le chiffre des dizaines au rang 1
le chiffre des centaines au rang 2 etc.

5	7	4	6
---	---	---	---

6475 devient 5746.

Lors de l'affichage du résultat, faite attention d'afficher dans le bon ordre, exemple, l'entier cent sera affiché 100 au lieu de 001 (à cause de la représentation).

Pour les algorithmes de calculs

l'addition et la soustraction, vous utiliserez ceux appris à l'école primaire!
la multiplication, par exemple le produit de axb peut être calculé en additionnant a , b fois. La voie est libre si vous voulez utiliser d'autres techniques.

4. Spécification formelle des membres de la classe GrdEnt

La classe GrdEnt contient les membres suivants :

Constructeurs

GrdEnt(); // post condition: initialise le grand entier à 0.

GrdEnt(int); // post condition: initialise le grand entier à une valeur entière

GrdEnt(const string &); // pré condition
s est un ensemble de digits, précédée (de manière optionnelle) par le signe + ou -
12564364367 ou +12564364367
// post condition
GrdEnt est initialisée avec l'entier représenté par la chaîne de caractères s, si
s contient une valeur alphanumérique, arrêt du programme avec un message d'erreur.

Entrées/sorties

operator << // post condition: GrdEnt insère dans un stream de sortie

operator >> //post condition: GrdEnt est extrait à partir d'un stream d'entrée, les blancs seront les délimiteurs.

Arithmétiques

Si A, B, C sont deux grands entiers, et k un entier (classique) alors:

operator += // A+=B; retourne un grand entier après addition par un GrdEnt
operator -= // A-=B; retourne un grand entier après soustraction par un GrdEnt
operator *= // A*=B; retourne un grand entier après multiplication par un GrdEnt
operator *= // A*=k; retourne un grand entier après multiplication par un entier.

Indice: Servez vous de ce qui précède pour faire :

operator + //A=B+C; addition de deux GrdEnt et retourne un GrdEnt
operator - //A=B-C; soustraction de deux GrdEnt et retourne un GrdEnt
operator * //A=B*C; multiplication de deux GrdEnt et retourne un GrdEnt
operator * //A=B*k; multiplication d'un GrdEnt par un entier et retourne un GrdEnt
operator * //A=k*B; multiplication d'un entier par un GrdEnt et retourne un GrdEnt

Comparaison

operator == // A et B sont-ils égaux? retourne un booléen
operator < // A est-il inférieur à B? retourne un booléen
operator != // A et B sont différents? retourne un booléen
operator > // A est supérieur à B? retourne un booléen
operator >= // A est supérieur ou égal à B? retourne un booléen
operator <= // A est inférieur ou égal à B? retourne un booléen

Membres privés de la classe

Signe monsigne // Signe{positif , négatif}

Representation Digits //classe pour stocker tous les chiffres du nombre

int NbreDigits //nombre de chiffres que le nombre contient

5. Spécification formelle des membres de la classe Representation

Constructeurs/destructeurs

Representation (); // initialise à zéro les membres privés

Representation (int taille); // initialise la taille du vecteur à taille

Representation (int taille, const char & data); // initialise la taille du vecteur à taille
// et la chaîne à data

Representation (const Representation & vec); // constructeur de copie

~Representation(); // destructeur

affectation

operator = // opérateur d'affectation

indexation

char & operator [] // retourne le k ème élément
const char & operator [] const // retourne le k ème élément

Membres privés

int maTaille; // nombre d'éléments dans mon tableau
char * maListe; // tableau utilisé pour stocker les données

N.B.

Vous êtes libres d'ajouter d'autres méthodes privées pour implanter les classes GrdEnt et Representation.

6. Récapitulation

```
class GrdEnt {  
  
    // constructeurs  
    // entrees/sorties  
    // arithmetiques  
    // comparaison  
    // membres prives dont:  
    Representation Digits  
    // +des choses que vous jugez utiles ou nécessaires  
}  
  
classe Representation {  
    // constructeurs  
    // destructeur  
    // affectation  
    // indexation  
    // membres prives  
}
```

7. Organisation des fichiers

vous devez structurer votre programme sur 4 fichiers:

Pour la classe **Representation**:

rep.h
rep.cpp

Pour la classe **GrdEnt**:

grdent.h
grdent.cpp

8. Tester votre programme

8.1. Compilation

Il ne vous est pas demandé d'écrire un programme client (i.e. d'écrire une fonction main). Pour tester des fragments de vos programmes, vous pouvez vous servir de notre application client. Pour le faire:

compilez votre programmes comme suit:

```
g++ -Wall -o test rep.cpp grdent.cpp partage.o calculette.o
```

puis:

```
./test
```

partage.o & calculette.o sont deux fichiers objets qui codent notre application client. La fonction "main" se trouve dans le fichier "calculette.o".

8.2. Description de l'application client

L'application client est une petite interface qui permet de réaliser les 3 opérations: addition (+), soustraction (-) et multiplication (*).

Après exécution du programme test, vous allez recevoir ce qui suit:

```
-----  
test des grands entiers!
```

```
A vaut:  
7437380164651435143865874852299929767
```

```
B vaut:  
2
```

```
La somme de A+B est egale a:  
7437380164651435143865874852299929769
```

```
A vous de jouer .....
```

```
entrer une valeur: 89548458704578045
```

```
----> 89548458704578045
```

entrer l'un des operateurs: + - * (ou la lettre: q pour quitter) +

entrer une valeur: 745745278524785427575755

----> 745745368073244132153800

entrer l'un des operateurs: + - * (ou la lettre: q pour quitter) q

Nous avons voulu tester votre programme avant de vous donner la possibilité de jouer avec! Pour cela, nous avons initialisé A & B avec deux valeurs du type GrdEnt. Et nous avons affiché le résultat. Par la suite, la main vous est donnée pour utiliser l'application client.

Dans notre exemple, nous avons donne au départ la valeur de:

89548458704578045

puis le signe + pour signifier l'addition que nous voulons faire,
la valeur d'un autre grand entier

745745368073244132153800

et le résultat qui s'affiche:

745745368073244132153800

puis la lettre 'q' pour quitter l'interface.

8.3. Fichiers à votre disposition

Le répertoire:

/u/dift1166/Sujet/tp2

est organisé comme suit:

répertoire **prg** contient les fichiers partage.o, calculette.o et partage.h

partage.o, calculette.o ont été compilés sous phobos/deimos

répertoire **enonce** contient l'énoncé sous plusieurs formats.

répertoire **redhat** contient les versions RedHat 5.2 & 6.0

9. Remise

Vous devez remettre **4 fichiers**:

Pour la classe **Representation**:

rep.h

rep.cpp

Pour la classe **GrdEnt**:

grdent.h

grdent.cpp

Vous devez faire deux types de remise :

- Papier (à glisser sous la porte de mon bureau, Pavillon André-Aisenstadt, local 2249)

bien identifier sur le listing le cours et vos noms.

- Électronique : dans le répertoire où est votre programme, tapez:

remise ift1166 tp2 rep.h rep.cpp grdent.h grdent.cpp

10. Date limite

Vendredi 12 Novembre 1999, MINUIT.

11. Barème

Le tp2 est noté sur 15 points.

Les points seront répartis comme suit :

Exactitude : 7 points

un programme avec des avertissements "warnings" mais qui fait des choses non prévues dans la spécification 0/7

un programme qui ne compile pas 0/7

un programme avec des avertissements ("warnings") et qui donne les résultats escomptés 5.5

Modularité : 6 points

structure de votre programme, décomposition entre .h et .cpp, décomposition en fonctions, la manière avec laquelle vous allez coder votre programme etc.

Style : 2 point

le code doit être lisible, bien indenté et commenté.