TRAVAIL PRATIQUE 1

MISE EN SITUATION

Vous devez créer un programme *C++* permettant de calculer le résultat d'opérations arithmétiques de base sur des nombres entiers positifs de taille illimitée.

DÉROULEMENT DE L'EXÉCUTION

Au lancement, le programme demande à l'utilisateur une opération arithmétique :

```
Entrez une opération arithmétique:
```

L'opération doit être composée de :

1. Un nombre entier positif
2. Un opérateur arithmétique parmi :
   a. **+** (addition)
   b. **-** (soustraction)
   c. *****, **x** ou **X** (multiplication)
   d. **/** (division)
   e. **%** (modulo)
   f. **^** (exposant)
3. Un nombre entier positif
4. L'opérateur **=**

Le programme affiche alors le résultat de l'opération et fait une pause pour permettre de le lire.

Par exemple :

```
Entrez une opération arithmétique: 2 + 5 =
7
Appuyez sur une touche pour continuer...
```

Lorsque l'utilisateur appuie sur une touche, l'écran est effacé et le programme demande une autre opération. Le programme boucle ainsi tant que des opérations valides sont données.

Lorsqu'une opération invalide est donnée (un opérateur invalide, un nombre invalide, vide ou négatif), le programme affiche un message d'erreur et se termine.

```
Entrez une opération arithmétique: 2 # 5 =
Opération invalide - Fin du programme
```

Le programme se protège contre les divisions par zéro, mais ceci n'est pas considéré comme une opération invalide.

```
Entrez une opération arithmétique: 12 / 0 =
Impossible de diviser par zéro!
Appuyez sur une touche pour continuer...
```

Lors de l'affichage du résultat, les groupes de 3 chiffres sont séparés par une espace. De plus, les zéros inutiles sont ignorés et ne sont pas réaffichés dans le résultat.

```
Entrez une opération arithmétique: 0001234567 + 1 =
1 234 568
```

Il n'y a pas de limite à la taille des nombres donnés, et le programme calcule correctement le résultat, peu importe la taille des deux nombres, pour toutes les opérations.

```
Entrez une opération arithmétique:
96 573 865 276 290 835 744 308 175 038 185 773 235 474 815 +
     5 664 265 502 542 164 938 216 598 743 265 984 372 659 =
96 579 529 541 793 377 909 246 391 636 929 039 219 847 474
```

Exemple d'exécution

Joint à cet énoncé, vous trouverez le fichier exécutable **ExempleTP1.exe**. Utilisez-le comme référence. Votre programme doit se comporter exactement comme celui-ci. Portez une attention particulière aux formats d'affichage. Les entrées et sorties **doivent être identiques**.

Algorithmes

Lecture de l'équation

Pour lire un nombre :

```
Lire un caractère
TANT QUE le caractère représente un chiffre
   Convertir le caractère en chiffre
   Ajouter le chiffre à un vecteur
Inverser le vecteur
Rogner les zéros inutiles
```

Le dernier caractère lu est l'opérateur. Pour le premier nombre, il doit faire partie des opérateurs supportés. Pour le deuxième nombre, il doit être « = ».

Si l'opérateur est invalide ou aucun chiffre n'a été lu, l'opération est invalide.

NOTEZ BIEN : La conversion du caractère en chiffre **doit** utiliser un vecteur de conversion. Les opérations arithmétiques avec des caractères ne sont pas permises, puisque jamais vues en classe.

Addition

```
Ajouter des 0 au plus petit nombre pour atteindre la taille du plus grand
POUR TOUS les chiffres
   Additionner les chiffres des 2 nombres, ainsi que la retenue
   Ajouter les unités de l'addition à un vecteur contenant le résultat
   Conserver les dizaines de l'addition comme retenue
SI il y a une retenue
   L'ajouter au résultat
```

Soustraction

```
Ajouter des 0 au plus petit nombre pour atteindre la taille du plus grand
SI le nombre2 est plus petit que le nombre1
   POUR TOUS les chiffres
      Soustraire du chiffre du nombre1 le chiffre du nombre2 et la retenue
      SI la différence est négative
         Ajouter 10 à la différence
         Conserver 1 comme retenue
      Ajouter la différence à un vecteur contenant le résultat
   Rogner les zéros inutiles
SINON
   Faire la même chose que le cas précédent, mais en soustrayant le
    nombre1 du nombre2
   Inverser le signe du dernier chiffre du résultat
```

Multiplication

```
POUR CHAQUE chiffre2 du nombre2
   Ajouter des zéros selon la puissance de 10 du chiffre2 à un vecteur
    contenant le résultat intermédiaire
   POUR CHAQUE chiffre2 du nombre1
      Multiplier les 2 chiffres, et additionner la retenue
      Ajouter les unités de la multiplication au résultat intermédiaire
      Conserver les dizaines de la multiplication comme retenue
   SI il y a une retenue
      L'ajouter au résultat intermédiaire
   Additionner le résultat intermédiaire au résultat
```

Division / Modulo

```
SI le nombre2 est 0
   Retourner une erreur
Initialiser un compteur à 0
TANT QUE le nombre2 est plus petit ou égal au nombre1
   Soustraire le nombre2 du nombre1
   Ajouter 1 au compteur
```

Pour la division, le résultat est le **`compteur`**. Pour le modulo, le résultat est le **`nombre1`**.

Exposant

```
Initialiser le résultat à 1
POUR toutes les valeurs jusqu'au nombre2
   Multiplier le résultat par nombre1
```

Affichage du résultat

```
POUR CHAQUE chiffre du résultat, EN SENS INVERSE
   Affiche le chiffre
   SI la position est un multiple de 3
      Affiche une espace
```

Fichiers fournis

Joint à cet énoncé, vous trouverez les fichiers **UtilitairesVecteur.h** et **UtilitairesVecteur.cpp**. Ils contiennent des fonctions qui vous seront utiles dans le développement de votre projet. Ils vous sont offerts gracieusement. Copiez-les et incluez-les dans votre projet.

Exigences

Fichiers

Joint à cet énoncé, vous trouverez le fichier **TP1.cpp**. Il contient la fonction `main` du projet, ainsi qu'une autre fonction. Vous **DEVEZ** copier et inclure ce fichier dans votre projet. Vous **NE DEVEZ PAS** le modifier.

Pour la correction, le fichier **original** fourni sera utilisé. Si votre programme nécessite des modifications à ce fichier, alors il ne compilera pas lors de la correction!

Vous devez créer et inclure dans votre projet les fichiers **Calculatrice.h** et **LectureEtAffichage.h**, et y déclarer les fonctions nécessaires à la compilation du fichier **TP1.cpp** fourni. Vous devez aussi créer et inclure dans votre projet les fichiers **Calculatrice.cpp** et **LectureEtAffichage.cpp** et y définir les fonctions. Vous pouvez créer d'autres fichiers si vous voulez.

Fonctions

Le but de ce travail est de mettre en pratique l'utilisation des fonctions. Votre programme devra donc limiter autant que possible la duplication de code et opter pour la définition de fonctions qui permettront la réutilisation du code. Il devra aussi découper en plus petites fonctions les tâches longues et complexes.

Commentaires et Normes de programmation

Votre programme doit respecter les normes de programmation et les normes de commentaires décrites dans le document **Normes de programmation.pdf**, disponible dans le dossier partagé du cours.

Plan de tests

Joint à cet énoncé, vous trouverez le fichier **Plan de tests.txt**. Il contient des équations à effectuer avec votre programme ainsi que le résultat attendu pour chacune. Utilisez-le pour tester votre programme.

Notez que vous pouvez copier les équations du fichier et les coller dans la console pour simplifier les tests.

Pondération

Ce travail compte pour **15** % de la note finale

- Fonctionnement correct du programme 40 %
  - Un programme qui ne compile pas entraine automatiquement 0 pour ce critère
- Choix approprié des types de données élémentaires 10 %
- Découpage efficace du code informatique 15 %
- Choix et organisation logiques des instructions 10 %
- Notation claire et pertinente de commentaires en nombre approprié dans le code informatique 20 %
- Respect systématique des normes de programmation 5 %

Pénalités

Instruction inconnue : Pénalité de **10** % par notion ou instruction non vue en classe.

Français : Jusqu'à **10** % de pénalité (commentaires, noms de variables/constantes/fonctions)

Retard : Pénalité de **10** % par jour de retard (incluant les fins de semaine)

Plagiat : Ceci est un travail individuel. Application de la Politique institutionnelle pour *toutes les personnes* impliquées.

Consignes

- Sur le site *GitHub*, créez un nouveau dépôt **privé** nommé « **TP1ProgStructuree** ».
  - Dans l'onglet « **Settings** », dans la catégorie « **Collaborators** », cliquez le bouton vert « **Add people** ».
  - Dans la fenêtre qui s'ouvre, entrez « **jbeaudet-cstj** », puis cliquez le bouton « **Add jbeaudet-cstj to this repository** ».
- Clonez ce nouveau dépôt sur votre ordinateur.
- Dans *Visual Studio*, créez un nouveau projet vide.
  - Nommez-le projet « **ProjetTP1** ».
  - Choisissez votre dépôt « **TP1ProgStructuree** » comme emplacement.
  - Nommez-la solution « **SolutionTP1** ».
  - Ajoutez les fichiers distribués et créez les autres fichiers requis.

Ne laissez pas votre travail dans les laboratoires

Les disques des ordinateurs des laboratoires sont accessibles à tous. Si vous y laissez votre travail et que quelqu'un d'autre y accède, vous pourriez vous retrouver dans une situation de plagiat.

Assurez-vous de sauvegarder votre travail sur *GitHub*, puis supprimez tout du disque local.

REMISE

Retrouvez (dans l'historique de *GitHub* ou à l'aide de la commande « **`git log`** ») l'identifiant du commit correspondant à votre remise. Copiez-le.

Ouvrez l'application « **`Bloc-Notes`** » (*notepad*) et collez l'identifiant copié. Sauvegardez le fichier.

Vous devez remettre le fichier contenant l'identifiant dans LÉA, dans la rubrique Travaux, pour le TP 1.

Vous devez remettre votre travail avant **`08:00`** le **`jeudi 14 novembre 2024`**. La date de remise dans LÉA, et non la date du commit, sera prise en compte pour déterminer les pénalités de retard.