420-555 Programmation appliquée aux objets connectés

TP1 – 10%

Interpréteur Brainf*ck

Introduction

Brainf*ck est un langage ésotérique représentant étroitement une machine Turing. Le langage opère sur un « ruban » de mémoire où chaque cellule est un octet de données. Seulement quelques opérations primitives sont spécifiées par le langage.

Syntaxe du langage

<	Déplace le ruban vers la gauche
>	Déplace le ruban vers la droite
+	Incrémente la valeur contenue à l'emplacement actuel du ruban
_	Décrémente la valeur contenue à l'emplacement actuel du ruban
[	Début de boucle – Si la cellule courante contient une valeur de 0, on saute directement à la fin de boucle correspondante
]	Fin de boucle – Saute directement au début de boucle correspondant
.	Imprime la cellule mémoire
,	Lit une valeur en mémoire

Spécifications

Vous devrez implémenter un interpréteur pour le langage Brainf*ck supportant l'intégrale de la syntaxe. Votre programme doit prendre en arguments un nom de fichier contenant le code a exécuté ainsi que, optionnellement, une taille maximale pour le ruban.

Pour lire le fichier de code Brainf*ck, vous pouvez utiliser la classe « **`ifstream`** » pouvant être inclus à l'aide du fichier en-tête « **`fstream`** »

Écrivez les classes suivantes :

1. Interpreteur
    a. Cette classe est la classe principale du programme. C'est elle qui va contenir les autres classes que vous allez écrire et se charger d'ultimement interpréter le code Brainf*ck.
2. Programme
    a. Cette classe va se charger de lire le code du programme. C'est aussi cette classe qui va se charger de conserver l'index de la commande a exécuté.
    b. Supportez les opérations suivantes :
        i. char lireInstruction()
            - Retourne l'instruction à exécuter
        ii. void instructionSuivante()
            - Déplace le curseur d'instruction vers l'instruction suivante
        iii. void finDeBoucle()
            - Déplace le curseur d'instruction vers la fin de boucle
        ii. bool finDeProgramme()
            - Retourne true si la fin du programme a été atteinte
3. Ruban
    a. Cette classe va gérer la mémoire accordée au programme Brainf*ck
    b. Supportez les opérations suivantes :
        i. unsigned char lire()
            - Retourne la valeur actuelle
        ii. void incremente()
            - Incrémente la valeur actuelle
        iii. void decremente()
            - Décrémente la valeur actuelle
        iv. void deplacerGauche()

- Déplace le ruban vers la gauche. Si l'on est pour dépasser le début du ruban, on saute à la fin.

v. void deplacerDroite()

- Déplace le ruban vers la droite. Si l'on est pour dépasser la fin du ruban, on saute au début.

4. TableauMemoire<T>
    a. Classe utilisée par la classe Ruban
    b. Classe templaté
    c. Gère un tableau dont la taille est définie à l'exécution.
    d. Assurez-vous de bien libérer la mémoire.
    e. Supportez les opérations suivantes :
        i. const T& lire(size_t index)
        ii. void ecrire(size_t index, const T& valeur);
        iii. size_t taille();

Critères d'évaluation

Un programme ne compilant pas se verra accordé automatiquement une note de 0.

Fonctionnalité du code 35%

- Le code répond aux spécifications demandées
- Le programme fonctionne bien et fait ce qui est demandé

Qualité du code 40%

- Bonne gestion de la mémoire, pas de fuite de mémoire.
- Évite les comportements non-définis (e.g. écriture passé les limites d'un buffer)
- Utilisation appropriée des pointeurs et des références
- Utilisation des bons opérateurs de suppression
- Utilisation juste des fonctionnalités du langage.

Esthétique du code 25%

- Noms de classes, de méthodes, de variables
- Clarté du code
- Commentaire