



Cégep de Saint-Hyacinthe
Département d'informatique

Algorithme et programmation (420-1MP-HY)

Travail Pratique #2

Préparé par
Martin Lalancette
bureau B-2335

DESCRIPTION

But :	Utiliser les chaînes de caractères, les boucles et les fichiers de type TEXTE
Objectifs :	<ul style="list-style-type: none"> • Élaborer les solutions via pseudo-code • Élaborer des cas tests manuels • Réviser les notions antérieures • Appliquer les principes logiques algorithmiques avec une bonne syntaxe en suivant vigoureusement les normes de programmation • Utiliser les fonctionnalités des chaînes de caractères • Utiliser les structures répétitives (boucles) • Utiliser les notions de fichiers de type TEXTE • Programmer en C# une application de type console (.NET Core) sous Visual Studio
Durée :	10 h
Pondération :	sur 10
Remise :	À la semaine 8.
Contenu général :	<ul style="list-style-type: none"> • Remettre vos documents d'analyse et votre solution contenant les deux projets dans un document .ZIP via LÉA avant la date et heure limite.
Notes	<ul style="list-style-type: none"> • Conserver une copie de sécurité. Il est de votre responsabilité de conserver une copie de sécurité dans l'éventualité où la lecture des données serait impossible. Cette copie doit <u>être disponible sur demande.</u>

Spécifications du travail

Dans ce travail pratique, il y aura deux parties à faire **seul ou en équipe de deux.** Pour l'ensemble des deux parties, vous devez créer **une seule solution nommée TP2.sln,** et **deux projets de type console .NET Core.**

Console #1: Jeu de dés – 50 %**Règles du jeu :**

- Deux joueurs s'affrontent dans une partie de dés
 - Vous devez saisir le nom des deux adversaires et les stocker dans leur variable respective.
- Les deux joueurs brassent (l'objet Random) deux dés cinq fois de suite. C'est celui qui a la plus grosse somme des résultats qui gagne. En cas d'égalité au cinquième lancer, on brasse à nouveau les dés tant et aussi longtemps qu'il n'y a pas de gagnant.
- À chaque partie, vous devez vider l'écran.
- À chaque lancée de dés, vous devez afficher le numéro du tour, le résultat de chaque dé pour chaque joueur et le cumulatif du tour. Voir écran ici-bas.
- Le nombre de victoires est comptabilisé pour chaque joueur et est affiché dans la barre de titre. Ces compteurs sont remis à zéro si les joueurs changent.
- Après une partie, un menu apparaît pour guider les utilisateurs. **Valider les choix.**
- Le résultat du dé doit être en format « string » écrit de la façon suivante : 1/6, 2/6... 6/6. Voici un exemple d'écran à programmer :

```

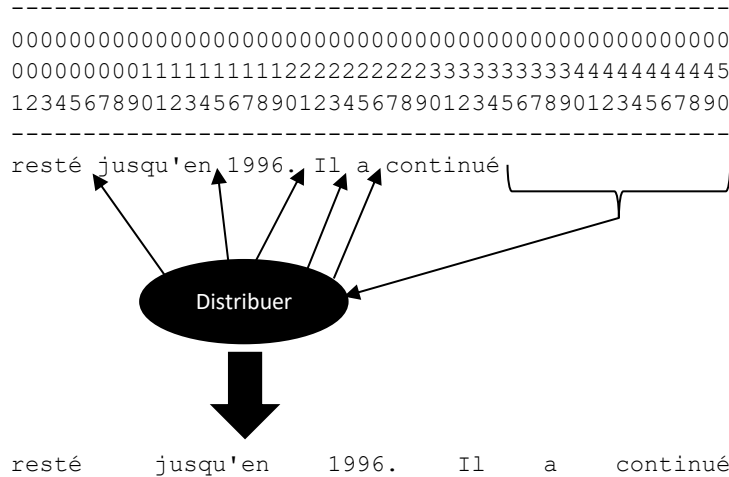
C# Jouer aux dés (Victoires) ==> Martin: 1 VS Hugo: 0
<===== JOUER AUX DÉES =====>
Nom du joueur #1: Martin
Nom du joueur #2: Hugo
----- Tour #1 -----
Résultat...: Martin --> 5/6 et 3/6 VS Hugo --> 6/6 et 4/6
Cumulatif..: Martin --> 8 et Hugo --> 10
----- Tour #2 -----
Résultat...: Martin --> 1/6 et 1/6 VS Hugo --> 1/6 et 3/6
Cumulatif..: Martin --> 10 et Hugo --> 14
----- Tour #3 -----
Résultat...: Martin --> 5/6 et 6/6 VS Hugo --> 6/6 et 2/6
Cumulatif..: Martin --> 21 et Hugo --> 22
----- Tour #4 -----
Résultat...: Martin --> 3/6 et 6/6 VS Hugo --> 4/6 et 1/6
Cumulatif..: Martin --> 30 et Hugo --> 27
----- Tour #5 -----
Résultat...: Martin --> 4/6 et 5/6 VS Hugo --> 2/6 et 2/6
Cumulatif..: Martin --> 39 et Hugo --> 31
Le gagnant est Martin avec 39 points.
Que voulez-vous faire?
  R) Rejouer avec les mêmes participants (défaut).
  N) Jouer avec des nouveaux joueurs.
  Q) Quitter le programme
Votre choix: 
  
```

Consignes :

1. Ajouter à la solution un projet de type console (.NET Core) nommé **JouerDés**.
2. Ajouter un fichier texte (ou Word) dans ce projet et y **rédigier le pseudo-code**.
3. Utiliser la grille de tests dans LÉA pour rédiger les cas de tests.
4. Dans le fichier **Program.cs**, coder la logique de votre pseudo-code en ajoutant des commentaires (Description, Auteur, Date, etc.) et en appliquant les normes.
5. Effectuer les cas de test **associés à la saisie des données** pour s'assurer de la bonne fonctionnalité.

Console #2 : Utilitaire servant à justifier un texte – 50 %

La compagnie STH Imprimerie inc. requiert vos services afin de concevoir un programme qui permet de **justifier** (aligner sur les marges de gauche et de droite) des textes contenus dans des fichiers de type texte. JUSTIFIER consiste à distribuer équitablement les espaces à la fin de la chaîne de caractères vers ceux se trouvant dans la chaîne de caractères. Exemple sur 50 colonnes :



Vous devez concevoir cet algorithme de distribution.

Au démarrage de l'application, vous devez faire apparaître le menu suivant :

```

===== Justifier un texte =====
A) Définir le nom du fichier à charger (Nom: Anders.txt)
B) Définir le nombre de colonnes en 50 et 120 (Dimension: 50)
C) Charger et afficher le texte justifié
D) Charger et enregistrer le texte justifié
Q) Quitter
Votre choix:

```

Après chaque opération, il y a une pause (attente d'une touche), la console se vide et le menu apparaît. Voici en détail chacune des opérations exigées :

1. **Définir le nom du fichier à charger** : Lorsque l'utilisateur choisit A, il faut demander d'entrer un nom de fichier TEXTE et valider son existence (Afficher un message d'erreur). Le nom du fichier est conservé dans une variable et affiché dans le menu. Il servira pour les choix C et D. Exemples :

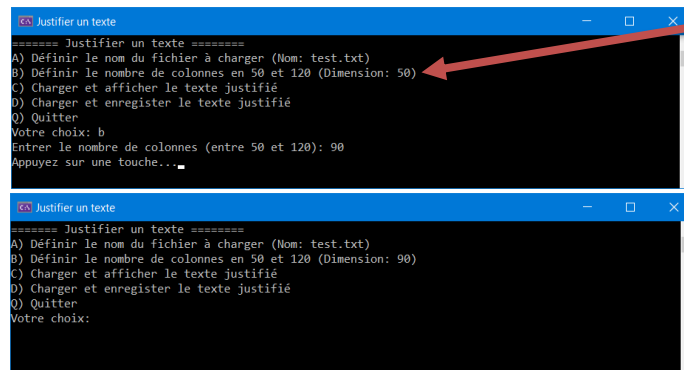
```

===== Justifier un texte =====
A) Définir le nom du fichier à charger (Nom: Anders.txt)
B) Définir le nombre de colonnes en 50 et 120 (Dimension: 50)
C) Charger et afficher le texte justifié
D) Charger et enregistrer le texte justifié
Q) Quitter
Votre choix: a
Entrez le nom du fichier à charger: test.txt
Appuyez sur une touche...

===== Justifier un texte =====
A) Définir le nom du fichier à charger (Nom: test.txt)
B) Définir le nombre de colonnes en 50 et 120 (Dimension: 50)
C) Charger et afficher le texte justifié
D) Charger et enregistrer le texte justifié
Q) Quitter
Votre choix:

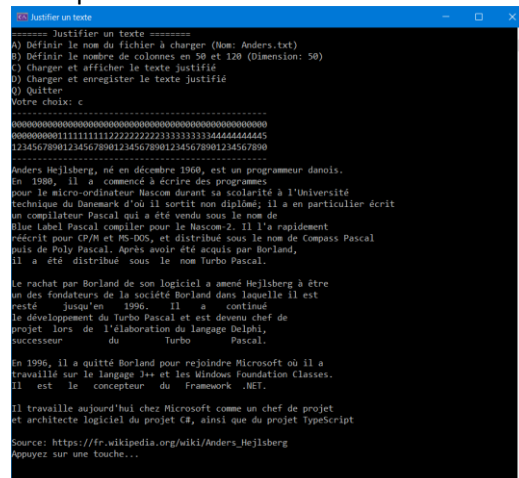
```

2. **Définir le nombre de colonnes** : Permet à l'utilisateur de définir le nombre de colonnes à utiliser pour justifier le texte. Ce nombre doit se situer entre 50 et 120 (à valider). Ce nombre est conservé dans une variable et affiché dans le menu. Il servira pour les choix C et D. Exemples :

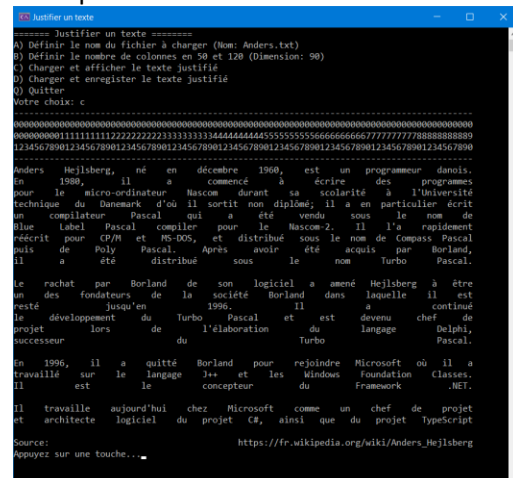


3. **Charger et afficher le texte justifié** : Cette opération consiste à ouvrir le fichier spécifié au choix A en mode lecture et lire le contenu ligne par ligne. Pour chaque ligne, justifier les caractères et l'afficher à l'écran selon le nombre de colonnes spécifié au choix B.

Exemple à 50 colonnes :



Exemple à 90 colonnes :



Les lignes de texte qui sont plus longues que la limite spécifiée seront affichées telles quelles (donc pas justifiées ni tronquées).

Les lignes de textes plus courtes que la limite doivent être justifiées.

Exemple à 120 colonnes :



4. **Charger et enregistrer le texte justifié** : Cette opération consiste à reproduire ce qui se passe lors du choix C, mais d'écrire le résultat justifié dans un fichier TEXTE. Il faut demander le nom du fichier avant. Exemple :

```

===== Justifier un texte =====
A) Définir le nom du fichier à charger (Nom: Anders.txt)
B) Définir le nombre de colonnes en 50 et 120 (Dimension: 80)
C) Charger et afficher le texte justifié
D) Charger et enregistrer le texte justifié
E) Quitter
Votre choix: d
Entrer le nom du fichier de sortie: test.txt
Appuyez sur une touche...
  
```

```

test.txt - Bloc-notes
-----
0000000000111111112222222233333333444444445555555566666666777777778888888899999999
1234567890123456789012345678901234567890123456789012345678901234567890
Anders Hejlsberg, né en décembre 1960, est un programmeur danois.
En 1980, il a commencé à écrire des programmes
pour le micro-ordinateur Nascom durant sa scolarité à l'Université
technique du Danemark d'où il sortit non diplômé; il a en particulier écrit
un compilateur Pascal qui a été vendu sous le nom de
Blue Label Pascal compiler pour le Nascom-2. Il l'a rapidement
réécrit pour CP/M et MS-DOS, et distribué sous le nom de Compass Pascal
puis de Poly Pascal. Après avoir été acquis par Borland,
il a été distribué sous le nom Turbo Pascal.

Le rachat par Borland de son logiciel a amené Hejlsberg à être
un des fondateurs de la société Borland dans laquelle il est
resté jusqu'en 1996. Il a continué
le développement du Turbo Pascal et est devenu chef de
projet lors de l'élaboration du langage Delphi,
successeur du Turbo Pascal.

En 1996, il a quitté Borland pour rejoindre Microsoft où il a
travaillé sur le langage J++ et les Windows Foundation Classes.
Il est le concepteur du Framework .NET.

Il travaille aujourd'hui chez Microsoft comme un chef de projet
et architecte logiciel du projet C#, ainsi que du projet TypeScript

Source: https://fr.wikipedia.org/wiki/Anders_Hejlsberg
  
```

5. **Quitter** : Permet de quitter la console.

Consignes :

1. Ajouter à la solution un projet de type console (.NET Core) nommé **JustifierTexte**.
2. Ajouter un fichier texte (ou Word) dans ce projet et y **rédigier le pseudo-code**.
3. Utiliser la grille de tests dans LÉA pour rédiger les cas de tests.
4. Dans le fichier **Program.cs**, coder la logique de votre pseudo-code en ajoutant des commentaires (Description, Auteur, Date, etc.) et en appliquant les normes.
5. **Concernant le nombre de colonnes inférieur (50) et supérieur (120), définir des constantes et les utiliser.**
6. Effectuer **5 cas de test** pour s'assurer de la bonne fonctionnalité.

Barème d'évaluation

Console #1 : Pseudo-code	/1.0
Console #1 : Élaboration des cas de tests (GrilleDeTests.xlsx).....	/1.5
Console #1 : Programmation : Respect des normes et des fonctionnalités demandées.....	/2.5
Console #2 : Pseudo-code	/1.0
Console #2 : Élaboration des cas de tests (GrilleDeTests.xlsx).....	/1.5
Console #2 : Programmation : Respect des normes et des fonctionnalités demandées.....	/2.5
Note totale*	/10.0

* Tout travail plagié en partie ou en totalité se verra attribuer une note totale de 0 %.

PDEA #1: Lors d'activités d'évaluation sommative en classe ou hors classe (programmes (incluant les commentaires), documentation, rapport de laboratoire, rapport de stage, examen), une pénalité **maximale de 10 %** peut être retranchée de la note finale de ladite évaluation (le barème étant **de 0,5%/erreur**).

PDEA #4 : Toute évaluation sommative remise après la date d'échéance fixée se voit attribuer la note zéro pour les étudiants **de 2^e à 6^e session**. Afin de faciliter l'accueil et l'intégration des nouveaux étudiants de **1^{re} session**, cette règle s'appliquera de la façon suivante : 30 % de pénalité pour une 1^{re} offense (à condition que la remise soit faite dans les 24 heures suivant la date de remise officielle), 100% de pénalité pour les offenses subséquentes.