UNIVERSITE LIBRE DE BRUXELLES
Faculté des Sciences Appliquées

29 mars 2010

INFO-H-100 : Introduction à la programmation
Projet 2 : *Boggle*

On vous demande de réaliser un jeu de « *Boggle* »[1] simplifié en `C++`.

La section suivante est basée sur Wikipedia.

Déroulement du jeu simplifié

Le jeu de « *Boggle* » en version simplifiée est un jeu à un joueur. Le jeu commence par le mélange d'un plateau (carré 4 fois 4) de 16 dés à 6 faces, généralement en le secouant. Chaque dé possède une lettre différente sur chacune de ses faces. Les dés sont rangés sur le plateau, et seule leur face supérieure est visible.

Après cette opération, le joueur cherche le mot le plus long pouvant être formé à partir de lettres adjacentes du plateau. Par « adjacentes », il est sous-entendu horizontalement ou verticalement. Les mots doivent être de 3 lettres au minimum, ne peuvent pas utiliser plusieurs fois le même dé, et doivent exister dans un dictionnaire fourni (voir plus bas).

Les points sont attribués suivant la taille des mots trouvés comme suit : si le mot à une taille de 3 ou 4 le joueur reçoit 1 point, pour des mots de 5, 6 ou 7 le joueur reçoit respectivement 2, 3 ou 5 points, finalement pour les mots de longueur 8 et plus le joueur reçoit 11 points.

Programme à réaliser

On vous demande de réaliser à l'aide uniquement de la matière vue au cours et aux séances d'exercices un jeu de « *Boggle » simplifié à un seul joueur* en C++. Votre programme devra se dérouler comme suit :

Tout d'abord, votre programme chargera en mémoire le dictionnaire ainsi que les configurations des 16 dés disponibles sous forme de fichiers sur la page web des projets[2]. Ces fichiers devront obligatoirement se trouver dans le même répertoire que l'exécutable de votre projet. Le format de ces fichiers est détaillé ci-dessous.

Ensuite, le jeu commence. Il est composé de 5 tours, chaque tour se déroule comme suit.

- Le plateau est mélangé et présenté à l'utilisateur.
- Le candidat propose un mot.
- Le système vérifie alors les conditions suivantes :
  - Le mot existe dans le dictionnaire fourni. La recherche dans le dictionnaire sera faite de manière efficace. Les diacritiques sont ignorés.
  - Le mot est composé uniquement des lettres adjacentes (horizontalement : à gauche ou à droite ou verticalement : en haut ou en bas) à la disposition du joueur. Chaque dé ne peut être utilisé qu'une seule fois. Pour simplifier cet algorithme, vous pouvez considérer que pour chaque lettre, il n'y a pas deux lettres adjacentes identiques.
- Si ces conditions sont vérifiées, le candidat gagne les points correspondant à la taille du mot proposé (comme expliqué dans l'introduction ci-dessus). Sinon, il perd ce nombre de points.

---

[1] `http://fr.wikipedia.org/wiki/Boggle`
[2] `http://cs.ulb.ac.be/public/teaching/infoh100/projets`

Exemple

Voici un exemple de plateau qui contient le mot « bonjour ».

E	Z	F	**R**
H	K	T	**U**
A	**B**	**J**	**O**
U	**O**	**N**	V

Structure des fichiers

Le fichier `dico.txt` contient une liste de mots triés alphabétiquement. Les mots sont séparés par un retour à la ligne.

Dans le fichier `des.txt` chaque ligne est composée des lettres composant un des dés. Extrait du fichier `des.txt` :

```
LENUYG
ELUPST
ZDVNEA
SDTNOE
```

Apports personnels

Les apports personnels seront valorisés à hauteur de 5 points sur 20. En d'autres mots, un projet parfait sans apport personnel aura une valeur de 15 points sur 20. Ces apports personnels peuvent être par exemple des règles ignorées dans la version simplifiée proposée ci-dessus.

Voici quelques exemples d'apports personnels inspirés notamment de la version complète du jeu décrite sur Wikipedia :

- Implémenter l'algorithme complet de recherche d'un mot dans le plateau, c'est à dire supprimer l'hypothèse que pour chaque lettre, il n'y a jamais deux lettres adjacentes identiques. Utilisez pour cela la technique du *backtracking*. (difficile).
- Gestion du temps : enregistrer le temps lorsque le plateau est présenté et enregistrer le temps lorsque l'utilisateur donne sa réponse, le mot est accepté si moins de trois minutes se sont écoulées (facile) ;
- Plusieurs mots : le joueur a 3 minutes pour proposer le plus de mots possibles, le score est la somme des scores de tous les mots (difficulté moyenne) ;
- Gestion de plusieurs joueurs (difficulté moyenne) ;
- Les cases diagonales sont aussi adjacentes (facile).
- Enregistrer une partie sur fichier et pouvoir la rejouer (difficile) ;
- Algorithme de recherche de la meilleure solution (très difficile) ;
- Affichage amélioré grâce à l'Art ASCII[3] (difficulté moyenne) ;
- Enregistrement des meilleurs scores (facile) ;
- Fichier de configuration (temps, difficulté, nombre de lettres d'un mot, . . .) (difficulté moyenne).

[3] `http://fr.wikipedia.org/wiki/Art_ASCII`

Rapport

Un rapport dactylographié vous est demandé. Celui-ci devra comprendre :

- Une page de garde indiquant clairement vos noms, numéros de matricules et votre série.
- Une **brève** introduction au problème.
- Une documentation de vos fonctions, c'est à dire pour chaque fonction : son prototype, une description succincte de la tâche qu'elle remplit, ses paramètres et leur domaine (le type et les valeurs possibles), le type de la valeur de retour et son domaine, les erreurs possibles et un exemple d'utilisation. Par exemple :

  `int daysInMonth(int month, int year)` : Calcule le nombre de jours du mois (`month`) de l'année (`year`) fourni en paramètres.
  - Paramètre `int month` : entier de 1 à 12 représentant le mois.
  - Paramètre `int year` : entier de 0 à 9999 représentant l'année au format AAAA.
  - Valeur de retour entière indiquant le nombre de jours du mois.
  - Erreurs : renvoie `-1` si les paramètres ne respectent pas les domaines.
  - Exemple : `daysInMonth(04,2008)` renvoie l'entier `30`.

- Une explication claire de vos choix algorithmiques pour :
  1. le mélange du plateau ;
  2. la vérification que le mot est composé de lettres adjacentes et n'utilise pas deux fois le même dé ;
  3. la recherche dans le dictionnaire.
- Une (ou plusieurs) capture d'écran.
- Le code source en police de caractère à empattement fixe (par exemple `courier`).

Consignes

Le projet se fera obligatoirement par groupes de deux étudiants de la même série.

Le rapport au format papier (uniquement des feuilles et des agraffes, **pas de plastique** ni de reliure) devra être rendu au bureau UB4.131. Le code source de votre application sera envoyé par mail à l'adresse ulb.code+infoh100@gmail.com. Ce mail devra avoir comme sujet "INFO-H-100 - Projet 2 - Nom1, Prénom1 et Nom2, Prénom2". Tout cela devra être rendu pour le **3 mai à 12h30**. Les groupes remettant leur projet après cette limite seront pénalisés.

Vous devrez défendre votre projet devant les assistants à un horaire fixé ultérieurement.

Votre code devra compiler sans avertissement à l'aide de la commande `g++ -Wall` installée dans les salles machines. Les seules librairies autorisées sont `iostream`, `ctime`, `cstdlib` et `fstream`.

L'évaluation de ce projet prendra en compte les points suivants :

- La résolution du problème énoncé.
- Le respect scrupuleux de l'énoncé.
- Le respect scrupuleux des conventions et des règles de bonne pratique publiés sur le site web des projets.
- La qualité du rapport : intérêt, présentation, soin, concision, grammaire et orthographe.
- Votre défense en salle machine devant les assistants.
- L'efficacité de vos algorithmes et structures de données, en restant bien sûr dans le cadre de la matière vue aux cours et aux séances d'exercices.

Les élèves assistants sont disponibles tous les midis dans la salle Socrate afin de répondre à vos questions.

Bon travail !

Lecture et écriture sur fichier

La librairie `fstream` est une librairie standard permettant de lire et écrire dans un fichier. Cette librairie offre deux nouveaux types de données : `ifstream` pour la lecture de fichier et `ofstream` pour l'écriture. Leur fonctionnement est similaire à celui de `cin` et `cout`.

Voici un exemple de lecture d'un fichier :

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
  char chaine[255];
  ifstream entree("test.txt");     //ouvre le fichier test.txt en lecture
                                   //ce fichier se trouve dans le meme dossier
                                   //que le programme
  entree >> chaine;
  while (!entree.eof())            //tant que le fichier n'est pas terminé
  {

    cout << chaine << endl;
    entree >> chaine;              //copie une portion dans le vecteur chaine
  }                                //(chaque portion est séparée par un espace,
                                   // une tabulation ou un retour à la ligne)

  entree.close();                  //ferme le fichier

  return 0;
}
```

Voici un exemple d'écriture d'un fichier :

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
  int valeur;
  ofstream sortie("truc.txt");     //ouvre le fichier truc.txt en écriture

  cin>>valeur;
  while (valeur != -1)
  {
    sortie << valeur << ',' ;      //écrit dans le fichier les valeurs lues
    cin >> valeur;                 //au clavier
  }

  sortie.close();                  //ferme le fichier

  return 0;
}
```