IUT de Paris - Rives de Seine
Université Paris Cité

BUT1 – SAÈ S1.02

Comparaison d'approches algorithmiques

– Le quart de singe –

Le but du projet est de développer un logiciel permettant à un ensemble de joueurs de disputer une partie de quart de singe. L'application doit veiller au respect des règles du jeu et gérer la totalité du déroulement de la partie jusqu'à l'annonce du perdant.

1 Règles du jeu

Les joueurs annoncent à tour de rôle une lettre qui vient compléter un mot. Un joueur donnant une lettre terminant un mot existant (de plus de deux lettres) perd la manche et est gratifié d'un quart de singe. Le joueur courant devant annoncer une lettre peut préférer demander au joueur précédant à quel mot il pense. Si ce dernier est incapable de donner un mot existant et qui soit cohérent avec les lettres déjà annoncées, il perd la manche. Dans le cas contraire, c'est le joueur qui a posé la question qui perd la manche. Le premier joueur ayant récolté quatre quarts de singe perd la partie.

Admettons que, lors des tours précédents d'une manche, les lettres 'A', 'B', 'A' et 'C' ont été annoncées dans cet ordre. Si le joueur courant annonce 'A', il perd la manche car le mot "ABACA" est un mot existant (c'est une matière textile). Si par contre, il joue 'U' et que le joueur suivant lui demande à quel mot il pense, il pourra répondre "ABACULE" (c'est un petit élément cubique d'une mosaïque) et ce joueur écopera d'un quart de singe.

Les mots retenus sont ceux acceptés au Scrabble, les accents sont ignorés et les verbes peuvent être conjugués.

2 Cahier des charges

L'application que vous devez développer doit permettre à au moins deux joueurs de jouer une partie dans sa totalité. Pour qu'un joueur seul puisse s'exercer, votre application doit implémenter des joueurs robots qui joueront automatiquement. Le nombre de joueurs, leur nature (humain ou robot) et l'ordre dans lequel les joueurs prendront leur tour sont spécifiés par une chaîne de caractère passée sur la ligne de commande. Par exemple, si votre fichier exécutable est nommé `singe.exe`, alors la commande `singe HRHR` lance une partie où 4 joueurs s'affrontent. Le premier et le troisième sont des humains alors que le deuxième et le quatrième sont des robots.

Dans les affichages, les joueurs sont désignés par leur numéro d'ordre (`1`, `2`, etc) et leur nature (`H` pour les humains et `R` pour les robots). A chaque tour de jeu, le numéro du joueur courant, sa nature ainsi que les lettres déjà annoncées sont affichés ("1H (ABAC) > " par exemple). Le joueur saisit soit une lettre de l'alphabet (non accentuée et en majuscule ou en minuscule) s'il veut compléter la chaîne de lettre, soit le caractère '?' s'il veut demander au joueur précédent à quel mot qu'il pense, soit le caractère '!' s'il veut abandonner la manche (et prendre un quart de singe).

Si la lettre jouée vient finir un mot existant du dictionnaire, votre application affiche "`le mot XXX existe, le joueur X prend un quart de singe`" (où `XXX` est remplacé par le mot ainsi formé et `X` par le numéro du joueur courant). Si le caractère '`?`' a été saisi, le joueur précédent est invité à saisir le mot auquel il pense. Si les premières lettres de ce mot ne correspondent pas à celles déjà jouées, votre application affiche "`le mot XXX ne commence pas par les lettres attendues, le joueur X prend un quart de singe`" (où `XXX` est remplacé par le mot saisi). Si le mot saisi n'appartient pas au dictionnaire, votre application affiche "`le mot XXX n'existe pas, le joueur X prend un quart de singe`". Dans le cas contraire, votre application affiche "`le mot XXX existe, le joueur X prend un quart de singe`". Dans le cas où le caractère '`!`' est saisi, votre application affiche `"le joueur X abandonne la manche et prend un quart de singe"`.

Vous disposez d'un dictionnaire de la langue française. Il est composé de 369085 mots. Seuls les mots de plus de deux lettres participent au jeu. Votre application devra faire l'hypothèse que le fichier texte correspondant se trouve dans le répertoire courant où est lancé l'application (surtout ne pas mettre de chemin absolu pour désigner le fichier).

Les robots jouent automatiquement. Les lettres jouées par ce type de joueurs ne sont pas saisies mais affichées. Bien entendu, ils ont accès au dictionnaire et vous devez les programmer de sorte qu'ils jouent le mieux possible.

À la fin de chaque manche, le nombre de quarts de singe de chaque joueur est affiché ("`1H : 0.25; 2R : 0; 3H : 0.75; 4R : 0`" par exemple). Si aucun joueur n'a reçu quatre quarts de singe, le joueur ayant perdu la dernière manche débute la suivante. Dans le cas contraire, l'application affiche "`La partie est finie`".

Un extrait d'une trace d'exécution est donné en annexe.

3 Qui, quoi et quand?

Votre projet doit être fait en binôme. Les groupes de 3 ne seront pas acceptés. Évitez de faire votre projet tout seul (soit vous êtes très fort et des personnes ont besoin de votre aide, soit vous avez des difficultés et il faut vous faire aider).

Vous pouvez employer les structures de données vues en cours ou développez les vôtres. Par contre l'usage des chaînes de caractère du C++ (le type `string`) ainsi que les conteneurs de la bibliothèque standard (`vector`, `list`, `map`, `stack`, etc) est strictement interdit. En cas de doute, contactez moi.

Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. Votre dossier doit être un unique document `pdf` dont la composition est la suivante :

- Une page de garde indiquant le nom et **le groupe** des membres du binôme, l'objet du dossier.
- Une table des matières de l'ensemble du dossier.
- Une brève introduction du projet.
- Le graphe de dépendance des fichiers sources de vos applications. Tous les composants (qu'ils soient réutilisés ou développés) de vos applications devront figurer sur le graphe (*cf.* Cours 4).
- Le code source des tests unitaires que vous aurez écrits (précisez quels tests passent et lesquels échouent).
- Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).
- En annexe, le code complet de vos sources (triez les fichiers selon un ordre logique).

Nous vous rappelons que le critère principal de notation est la structuration de votre code. Votre rapport doit mettre en avant la qualité de celle-ci.

Tous les éléments (constantes, types et fonctions) déclarés au sein d'un fichier `.h` doivent être documentés.

Vous devez déposer une archive **zip** contenant votre rapport ainsi que l'ensemble des fichiers sources de vos applications, le **mardi 3 janvier 2023**.

Annexe

```
1H, () > A
2R, (A) > B
3H, (AB) > A
4R, (ABA) > C
1H, (ABAC) > ?
4R, saisir le mot > ABACUS
le mot ABACUS existe, 1H prend un quart de singe
1H : 0.25; 2R : 0; 3H : 0; 4R : 0
1H, () > A
2R, (A) > B
3H, (AB) > A
4R, (ABA) > C
1H, (ABAC) > A
le mot ABACA existe, 1H prend un quart de singe
1H : 0.5; 2R : 0; 3H : 0; 4R : 0
1H, () > A
2R, (A) > B
3H, (AB) > A
4R, (ABA) > ?
3H, saisir le mot > ABACADABRA
le mot ABACADABRA n'existe pas, 3H prend un quart de singe
1H : 0.5; 2R : 0; 3H : 0.25; 4R : 0
3H, () > ...
...
le mot XYZ n'existe pas, 3H prend un quart de singe
1H : 0.5; 2R : 0; 3H : 1; 4R : 0
La partie est finie
```

Figure 1: Exemple de session – en rouge, les données saisies par les joueurs humains, en bleu, les messages affichés par le programme.