	Algorithmique et programmation séquentielle	
	ITI1	***Le jeu du «démineur »***
	hepia, HES-SO//Genève	Laboratoire Ada

Objectifs

1. Utilisation de types énumérés et tableaux, de procédures et fonctions.
2. Affichage d'abord en mode texte puis en mode graphique.
3. Récursivité
4. Indentation, commentaires et modularisation du code.

Enoncé

Il s'agit de programmer le jeu « démineur » en Ada (voir la figure ci-dessous).

Règles du jeu

Le jeu du « démineur » se joue seul.
Un nombre fixé de bombes cachées se trouvent sur une aire de jeu rectangulaire.
A chaque étape, le joueur peut soit ouvrir une case, soit placer un drapeau là où il pense que se trouve une bombe.
Lorsqu'une case est ouverte, le nombre de bombes dans le voisinage de cette case est affiché. Si ce nombre est 0, alors les 8 voisins de la case sont également ouverts. Le processus est répété pour tout voisin n'ayant aucune bombe dans ses 8 cases adjacentes, puis ceci est propagé aux voisins, etc.
Si une case contenant une bombe est ouverte, le jeu se termine sur une défaite et l'aire de jeu est affichée avec toutes les cases ouvertes.
Si le joueur réussit à ouvrir toutes les cases sans bombe, alors il gagne et la partie se termine.
Le temps écoulé doit s'afficher à la fin de la partie.
A noter qu'il ne doit pas être possible de placer plus de drapeaux que de bombes et que le joueur peut toujours décider d'enlever un drapeau d'une case s'il pense s'être trompé sur la présence d'une bombe.
Pour un exemple, voir http://demineur.hugames.fr/

Recommandation

Dans un premier temps, contentez-vous de :

- implémenter une version en mode texte avec un affichage sur la console ;
- propager la découverte des cases avec zéro bombe autour, uniquement sur les verticales, horizontales et diagonales.

Renseignements

Une partie graphique pourra être réalisée dans une 2ème version avec la librairie Sglib. Pour utiliser Sglib dans un programme, il suffit de l'inclure dans la clause de contexte :

```
with Sglib; use Sglib;
```

D'éventuelles indications de compilation vous seront fournies en séance.

Une documentation sur Sglib (fichier gnat_gvd_gtk.pdf) est accessible sur *cyberlearn* dans « Ressources ADA »

La bibliothèque est accessible dans le répertoire Sglib dans Doc. Un fichier README vous guidera à travers les étapes de compilation.

Pour les mesures de temps, utiliser la librairie Calendar qu'il suffit d'inclure dans la clause de contexte : `with Ada.Calendar; use Ada.Calendar;`

Dans le corps du programme, avec des variables start et temps de type Integer, vous pouvez afficher le temps écoulé en secondes avec :

```
start := Integer(Seconds(clock));
... -- instructions à chronométrer
temps := Integer(Seconds(clock)) - start;
```

La fonction clock de la bibliothèque Calendar retourne un objet de type Time. La fonction Seconds qui prend un objet de type Time en paramètre retourne un objet Duration. Celui-ci peut être convertit en Natural.

Vous êtes fortement encouragé à aller au libre service pour obtenir des conseils et de l'aide pour avancer votre TP.

Rendu

Le listing du code sur papier doit être rendu **au plus tard le 12 novembre 2017**. **Veuillez imprimer deux pages par feuille.** Le code doit être bien **indenté, commenté et modularisé** avec des fonctions et procédures. Un seul fichier source à votre nom de la version en **mode texte** devra être déposé **au plus tard le 12 novembre 2017 sur cyberlearn**, dans ***hepia_Algorithmique et programmation séquentielle***, répertoire **`Demineur.`**

Plus précisément, l'étudiant **Jean-Albert Sampaio Rodriguez** déposera un **seul** fichier nommé **jean_albert_sampaio_rodriguez.adb**

Attention ni espaces, ni accents, ni majuscules dans les noms !

Un fichier de base nommé **demineur_skel.adb** vous est fourni dans lequel est indiqué les **formats des entrées et sorties**. Celui-ci ne doit être modifié qu'aux endroits indiqués "à compléter" ou "à modifier". Vous devez impérativement respecter les indications données dans ce fichier. Aucune autre saisie au clavier ou impression à l'écran ne doit être faite dans la version que vous rendrez.

Le programme de Jean-Albert Sampaio Rodriguez doit pouvoir être lancé en ligne de commande avec la syntaxe :

```
<nom_executable> <dimensions_aire_jeu> <liste_positions_bombes>
```

Par exemple : `./jean_albert_sampaio_rodriguez 5 5 3 4 1 1 2 5`

ce qui créera une aire de jeu de taille 5×5 avec 3 bombes aux positions (3,4), (1,1) et (2,5).

Si le programme est juste lancé avec : `./jean_albert_sampaio_rodriguez 5 5`

alors une aire de jeu de taille 5×5 sera créée avec les bombes placées aléatoirement, leur nombre étant saisi par l'utilisateur.

Sous peine de sanction, vous devez respecter toutes les consignes.

Ce travail pratique est noté. L'évaluation sera faite sur la base de l'exécution de votre programme, du listing et d'une interrogation orale.