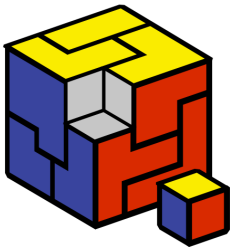
Prologin 2011



The Grid

SHALL WE PLAY A GAME?

Sujet de la finale du Concours National d'Informatique
Samedi 23 avril 2011

Hein ? Quoi ?

...

Où suis-je ?

...

Qui sont tous ces gens ? Pourquoi de jolies filles [1] viennent me déshabiller ?

...

```
Greetings, Program! I'm Castor! Your host! Provider of any and
all, entertainment, and diversions...

... at your service.
```

1. Si vous vous prénommez Betty, Diane ou Estelle, comprendre « de beaux garçons ».

Table des matières

1 Introduction

La Grille.

Ce monde de programmes, régi par des programmes pour les programmes, est dirigé d'un pointeur de fer par Clu. Chargé de concevoir un monde parfait, Clu fait régner la terreur, désactivant sans pitié [2] tout programme faisant un pas de travers [3].

Depuis qu'il connaît l'existence du monde réel, Clu cherche à tout prix à s'y infiltrer, pour y porter sa vision du monde parfait. La seule passerelle entre son monde numérique et le nôtre est le Portail, dont l'ouverture nécessite des quantités astronomiques d'énergie.

Pour arriver à ses fins, il organise donc des jeux sur la Grille pour que les programmes s'affrontent deux à deux. Cela lui permet à la fois de déterminer les éléments les mieux conçus au moyen d'un classement, et de recueillir de l'énergie.

Prologin, une organisation secrète visant à sauver la Terre d'une destruction imminente, a repéré les 100 meilleurs concepteurs [4] à travers toute la France. Pensant qu'ils allaient participer à la finale d'un concours d'informatique [5], ils ont en réalité été envoyés dans la Grille en mission d'infiltration [6].

Vous êtes donc à présent des programmes.

2. Le fameux supplice de la planche à Clu.
3. Tu fais une *segfault*, t'es mort.
4. Les 100 les moins mauvais, disons.
5. Ha ha, non mais quels naïfs !
6. Et non, cette fois, ce n'est pas en vous suicidant que vous retournerez dans le monde réel. Bien essayé.

1.1 Votre mission

```
"What am I supposed to do?"
"Survive."
```

Vous devez relier des sources d'énergie à des consommateurs, en fonction des teneurs des premières et des besoins des dernières, au moyen d'une succession de traînées contiguës extensibles laissées par des lumicycles [7].

Lors d'une partie, deux programmes s'affronteront, agissant à tour de rôle. Au bout de 150 tours, celui qui aura obtenu le plus de points de la Grille sera le vainqueur.

Attention. C'est lui ou vous [8].

1.2 La Grille

```
The Grid. A digital frontier. I tried to picture clusters of
information as they moved through the computer. What did they
look like? Ships, motorcycles? Were the circuits like freeways?
I kept dreaming of a world I thought I'd never see. And then, one
day, I got in...
```

Vous vous battrez sur la Grille, environnement numérique carré se découpant en 30 cases par 30 cases, sur lesquelles sont notamment éparpillées les précieuses sources d'énergie et les consommateurs.

7. Vous pouvez relire la phrase.

8. C'est dramatique, mais voyez le bon côté des choses : vous aurez une plus grosse part de gâteau lundi midi si votre adversaire ne revient pas dans le monde réel.

2 Les règles

`Patience, Sam Flynn. All of your questions will be answered soon.`

Un lumicycle est une moto qui laisse une traînée de lumière.

Vous êtes-vous déjà demandé pourquoi les traînées [9] des motos sont dangereuses ? C'est parce qu'elles véhiculent de l'énergie !

2.1 Les traînées de lumière

À chaque tour, vous interagirez avec des traînées de lumière sur la Grille. Vous pouvez en déplacer les extrémités, les couper, les fusionner, ou les concentrer en un point.

Deux traînées de lumière ne peuvent pas se croiser sur une case normale.

Soyez vigilants, tout ordre vous coûtera un PA (point d'action), et ces PA sont rares [10]. La Grille vous en donne un certain nombre en début de tour, mais récupérera les PA inutilisés en fin de tour.

2.2 La Grille

`In there is a new world! In there is our future! In there is our destiny!`

Sur la Grille, vous pouvez rencontrer :

- des **obstacles**  ;
- un **bonus** que vous pouvez ramasser en vous déplaçant dessus ;

- un **point de croisement**  : sur ces cases, il est possible pour au moins deux traînées de se croiser sans dommage.
- une **unité d'énergie** : soit une source d'énergie, soit un consommateur.

Sans oublier les traînées de votre adversaire, qui se comportent comme des obstacles.

2.3 Les unités d'énergie

Votre objectif est de transférer un maximum d'énergie des sources vers les consommateurs, et éventuellement d'empêcher votre adversaire de le faire.

9. Ne jouez pas sur les mots, hein, on a déjà assez peu de filles comme ça.
10. Comme dirait Ken le survivant, « Décidément les temps comme les œufs sont durs. »

2.3.1 Transfert d'énergie

Pour vous relier à une source d'énergie, il suffit qu'une des traînées que vous commandez touche l'une des 8 cases[11] autour de celle-ci. Il en va de même pour les consommateurs. Deux traînées peuvent se transmettre de l'énergie si on peut sauter de l'une à l'autre en empruntant une des 4 directions[12]. Une source reliée à au moins un consommateur par vos traînées seulement est dite *active*.

Chaque unité d'énergie est caractérisée par une valeur v. Si elle est positive, il s'agit d'une source de *teneur* v. Si elle est négative, il s'agit d'un consommateur de *besoin* $-v$. Si cette valeur est nulle, l'unité est dite *épuisée*.

À la fin du tour, le transfert issu d'une liaison entre des sources de teneur totale T (somme des teneurs) et des consommateurs de besoin total B vous rapportera $\min(T, B)$ points. Il est donc de votre intérêt de relier les sources les plus abondantes aux consommateurs les plus affamés.

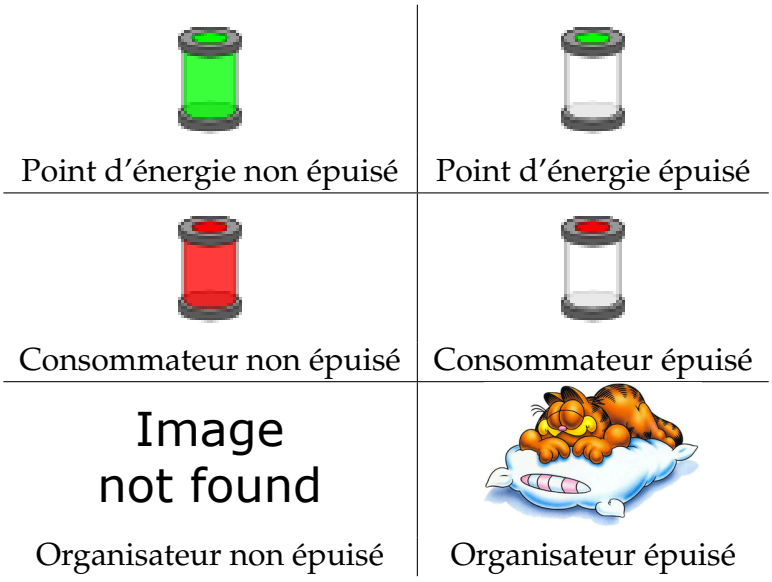


FIGURE 1 – Les différentes unités d'énergie

2.3.2 Affaiblissement et régénération des unités

La *charge* d'une traînée est le nombre d'unités directement reliées à celle-ci.

À chaque fin de tour, après que les 2 programmes ont joué, chaque source d'énergie active (resp. consommateur actif) diminue sa teneur (resp. besoin) de la somme des charges des traînées qui lui sont directement reliées[13].

11. Et non 4.
12. Et non 8.
13. Si vous n'avez pas compris, demandez à un gentil organisateur (ils se font rares, en cette saison) et priez pour qu'il ne vous dise pas le contraire.

Les sources inactives (resp. consommateurs inactifs), quant à elles, augmentent leur teneur (resp. besoin) d'un point.

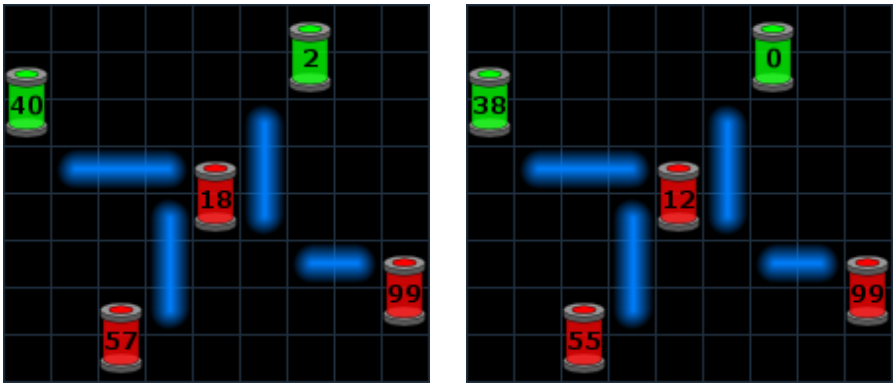


FIGURE 2 – Un exemple de transfert, avant puis après la fin du tour

Ici, 4 unités d'énergie sont reliées (le consommateur de besoin 99 n'appartient pas au réseau car les traînées ne peuvent véhiculer de l'énergie en diagonale), le nombre de points obtenu est : $\min(40+2, 18+57) = 42$[14]. Le consommateur de besoin 99 n'est pas relié à une source, donc il n'est pas affaibli à la fin du tour.

2.4 L'interaction avec la Grille

```
This is Blue Leader to Blue Bikes. Run these guys into your jet
walls.
```

2.4.1 Déplacement d'une traînée

Une traînée de lumière a une certaine *intensité*, qui représente le nombre maximum de cases sur lesquelles elle peut s'étendre.

Vous pouvez déplacer une traînée à partir d'une de ses deux extrémités en direction des 4 points cardinaux.

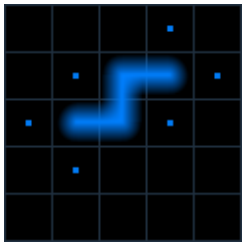


FIGURE 3 – Différents déplacements possibles

14. Vous voyez, ce n'était même pas la peine de vous poser la question.

Lorsqu'une traînée n'est pas complètement déployée, le déplacement d'une de ses extrémités laisse l'autre extrémité immobile.

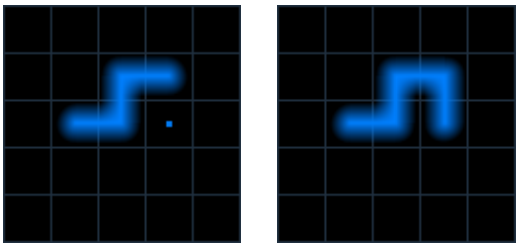


FIGURE 4 – Déplacement d'une traînée de longueur 4 d'intensité 5

Lorsqu'une traînée est complètement déployée, les deux extrémités se déplacent.

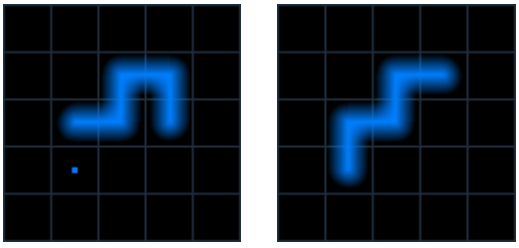


FIGURE 5 – Déplacement d'une traînée de longueur 5 d'intensité 5

2.4.2 Autres actions

Vous pouvez **couper** une traînée entre deux positions adjacentes et répartir l'intensité comme vous le souhaitez entre les deux traînées résultantes, que vous pouvez contrôler indépendamment.

Inversement, il vous est possible de **fusionner** deux traînées contiguës [15]. La traînée résultante aura pour intensité la somme de leurs intensités.

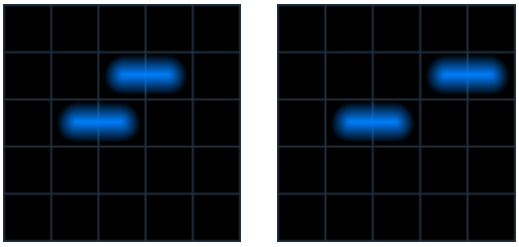


FIGURE 6 – Deux traînées contiguës à gauche, non contiguës à droite.

Enfin, vous pouvez **concentrer** toute une traînée en un seul point, sans altérer son intensité.

15. « contiguës », hein.

3 Les bonus

`Now for some real user power.`

Les bonus qui peuvent apparaître sur la carte sont les suivants.

- **point de croisement**  : vous permet de placer un point de croisement sur n'importe quelle case de la Grille ;
- **régénération**  : restaure la teneur maximale d'une source ou le besoin maximal d'un consommateur ;
- **allongement**  : augmente l'intensité d'une traînée ;
- **PA+**  : augmente votre nombre de points d'action.

Vous pouvez les utiliser quand vous le souhaitez [16].

4 Conclusion

`End of Line, man.`

À l'issue d'un tournoi sans merci de 36 heures, un classement sera établi. Clu, élitiste, vouera une confiance aveugle aux 10 meilleurs programmes. Or, être puissant ne veut pas dire qu'on est de confiance [17]. Ainsi, l'Équipe des Dix rejoindra le monde réel, en compagnie de Clu. Leur tâche consistera ensuite à le noyer [18] dans la zone de décontamination [19]. Enfin, ils passeront un entretien avec le Conseil des Quatre pour expliquer les stratégies qu'ils auront utilisées. Quant aux 90 autres, ils seront condamnés à perdre au jeu éternellement [20].

En bref, le sort de la Terre est entre vos mains.

« Je trouve toujours toute cette compétition dégueulasse »
— Valérie Lemercier

16. Avant la fin des 150 tours, c'est mieux.
17. Prenez exemple sur le gouvernement français.
18. Clu ne sait pas nager, il est un peu rouillé.
19. Communément appelée *piscine de mousse*.
20. Je sais. C'est terrible.

5 API

5.1 Constantes

Constante : TAILLE_TERRAIN
Valeur : 30
Description : Taille du terrain

Constante : FIN_PARTIE
Valeur : 150
Description : Nombre de tours par partie

Constante : MAX_PA
Valeur : 3
Description : Nombre de points d'action par tour

Constante : INTENSITE_TRAINEE
Valeur : 120
Description : Taille des traînées de moto

Constante : MAX_ALLONGEMENT
Valeur : 5
Description : Longueur maximale de l'allongement

Constante : AJOUT_PA
Valeur : 5
Description : Nombre de points d'action à rajouter avec bonus

5.2 Énumérations

- **erreur**

Description : Énumération représentant une erreur renvoyée par une des fonctions d'action

Valeurs :

OK :	aucune erreur n'est survenue
ID_INVALIDE :	identifiant invalide
POSITION_INVALIDE :	la position spécifiée est invalide
PLUS_DE_PA :	vous n'avez pas assez de points d'action
BONUS_INVALIDE :	vous n'avez pas ce bonus
PAS_A_TOI :	l'unité n'est pas a vous
INTENSITE_INVALIDE :	cette intensité est invalide

- **type_case**

Description : Énumération représentant les différents types de case

Valeurs :

VIDE :	rien n'est présent sur la case
OBSTACLE :	cette case est inaccessible
POINT_CROISEMENT :	point de croisement de traînées
UNITE :	unité d'énergie

- **type_bonus**

Description : Énumération représentant les différents types de bonii

Valeurs :

PAS_BONUS :	ceci n'est pas un bonus :-)
BONUS_CROISEMENT :	bonus permettant de croiser deux traînées de moto sur une case
PLUS_LONG :	bonus permettant d'agrandir une traînée de moto
PLUS_PA :	bonus permettant d'avoir plus de points d'action
BONUS_REGENERATION :	bonus permettant de regenerer une unité d'énergie

5.3 Structures

- **position**

```
struct position {
    int x;
    int y;
};
```

Description : Représente une position sur le terrain du jeu

Champs :

x :	coordonnée en X
y :	coordonnée en Y

- **unite_energie**

```
struct unite_energie {
    int id;
    position pos;
    int valeur;
    int valeur_max;
};
```

Description : Caracteristiques d'une unité d'énergie

Champs :

id :	identifiant de l'unité d'énergie
pos :	position de l'unité d'énergie
valeur :	coefficient représentant les points d'énergie que l'unité va vous apporter
valeur_max :	coefficient représentant la capacité de l'unité lorsqu'elle est chargée au maximum

- **trainee_moto**

```
struct trainee_moto {
    int id;
    position array emplacement;
    int team;
    int intensite;
};
```

Description : Représente une traînée de moto sur le terrain

Champs :

id :	identifiant de la traînee
emplacement :	position de chaque composant de la traînée de moto
team :	identifiant de l'équipe qui possède cette traînée de moto
intensite :	taille maximale de la traînée

5.4 Fonctions d'information

Les fonctions d'information permettent à chaque champion de prendre connaissance de l'état courant de la partie. Sans prendre en compte les performances, un champion peut les appeler autant de fois qu'il le désire sans aucune répercussion. En d'autres termes, elles ne modifient pas l'état du jeu.

- **mon_equipe**

```
int mon_equipe()
```

Description : Retourne le numéro de votre équipe

• **scores**

```
int array scores()
```

Description : Retourne les scores de chaque équipe

• **nombre_equipes**

```
int nombre_equipes()
```

Description : Retourne le nombre d'équipes sur le terrain

• **tour_actuel**

```
int tour_actuel()
```

Description : Retourne le numéro du tour actuel

• **unites_energie**

```
unite_energie array unites_energie()
```

Description : Retourne la liste des unités d'énergie

• **trainees_moto**

```
trainee_moto array trainees_moto()
```

Description : Retourne la liste des traînées de moto

• **regarder_type_case**

```
type_case regarder_type_case(position pos)
```

Description : Retourne le type d'une case
Parametres : *pos* : position de la case

• **regarder_type_bonus**

```
type_bonus regarder_type_bonus(position pos)
```

Description : Retourne le type de bonus d'une case
Parametres : *pos* : position de la case

• **regarder_bonus**

```
type_bonus array regarder_bonus(int equipe)
```

Description : Retourne la liste des bonus d'une équipe
Parametres : *equipe* : identifiant de l'équipe visée

• **regarder_trainee_case**

```
int array regarder_trainee_case(position pos)
```

Description : Retourne la liste des id des traînées présentes sur une case
Parametres : *pos* : position de la case

• **case_traversable**

```
bool case_traversable(position pos)
```

Description : Retourne si une case peut être traversée par une traînée de plus
Parametres : *pos* : position de la case

• **gain_tour_suivant**

```
int gain_tour_suivant()
```

Description : Renvoie les points que vous allez gagner a la fin du tour

• **chemin**

```
position array chemin(position p1, position p2)
```

Description : Renvoie le chemin le plus court entre deux points (fonction lente)
Parametres : *p1* : position de départ
p2 : position d'arrivée

5.5 Fonctions d'action

À l'inverse des fonctions d'information, les fonctions d'action modifient l'état du jeu. Chaque action coûte à un champion un Point d'Action (PA). Un champion est par conséquent limité dans le nombre d'actions qu'il peut effectuer par tour par sa réserve de PA. La constante MAX_PA indique combien

de PA un champion a au début de chaque tour. Le bonus `PLUS_PA` permet à un champion d'obtenir plus de PA, mais ceux-ci ne sont valables que *pendant le tour courant* : ils sont perdus ensuite.

Chaque fonction d'action renvoit un code d'erreur, indiquant si l'action s'est bien passée (constante `OK`), ou si l'action est invalide, précisant pourquoi (voir les autres constantes d'erreur)."

Pour rendre l'écriture d'un champion plus facile, la fonction spéciale `annuler` a été introduite. Elle permet d'annuler la précédente action effectuée pendant ce tour.

- **deplacer**

```
erreur deplacer(int id, position de, position vers)
```

Description : Déplace une moto
Parametres : *id* : identifiant de la moto à déplacer
de : position de l'extrémité que l'on déplace
vers : nouvelle position pour cette extrémité

- **couper_trainee_moto**

```
erreur couper_trainee_moto(int id, position p1, position p2, int intens
```

Description : Coupe une traînée de moto en deux nouvelles traînées. « p1 » et « p2 » doivent être deux positions adjacentes occupées par une même traînée de moto.
Parametres : *id* : identifiant de la traînée de moto à couper
p1 : nouvelle extrémité de la première traînée de moto
p2 : nouvelle extrémité de la deuxième traînée de moto
intensite_p1 : croissance restante de la moitié de la traînée de moto conte

- **annuler**

```
bool annuler()
```

Description : Annule l'action précédente. Renvoie true si une action a été annulée, false sinon.

- **enrouler**

```
erreur enrouler(int id, position p)
```

Description : Enroule la traînée de moto en un point
Parametres : *id* : identifiant de la traînée de moto à enrouler
p : point sur lequel enrouler la moto

• **regenerer_unite_energie**

```
erreur regenerer_unite_energie(int id)
```

Description : Régénère une unité d'énergie à son maximal
Parametres : *id* : identifiant de l'unité d'énergie à regénérer

• **allonger_pa**

```
erreur allonger_pa()
```

Description : Allonge le tour en rajoutant des points d'action

• **etendre_trainee_moto**

```
erreur etendre_trainee_moto(int id, int longueur)
```

Description : Allonge une traînée de moto. L'allongement se fera aux prochains déplacements. La longueur du prolongement doit être comprise entre 0 et MAX_ALLONGEMENT (inclus).
Parametres : *id* : identifiant de la traînée de moto à allonger
longueur : longueur du prolongement

• **poser_point_croisement**

```
erreur poser_point_croisement(position point)
```

Description : Pose un point de croisement sur une case du terrain. La case doit ne pas déjà être un point de croisement.
Parametres : *point* : position de la case sur laquelle poser le point de croisement

• **fusionner**

```
erreur fusionner(int id1, position pos1, int id2, position pos2)
```

Description : Fusionne deux traînées de moto. Les deux doivent appartenir à la même équipe, mais doivent être deux traînées distinctes. « pos1 » et « pos2 » doivent être adjacentes et occupées respectivement par « id1 » et « id2 ».
Parametres : *id1* : identifiant de la première traînée
pos1 : extrémité à fusionner de la première traînée
id2 : identifiant de la seconde traînée
pos2 : extrémité à fusionner de la seconde traînée

6 Notes sur l'utilisation de l'API

6.1 C

- Les booléens sont représentés par des entiers. `false` est représenté par la valeur 0, tous les autres entiers sont équivalents à `true` ;
- Les fonctions prenant des tableaux en paramètres prennent à la place de ce tableau deux paramètres : `type* tableau`, le tableau, et `size_t size`, la taille du tableau ;
- Les fonctions retournant des tableaux retournent à la place le type `void`, et prennent deux paramètres supplémentaires : `type** tableau` ou sera placé le tableau résultant (dont l'allocation n'est pas à votre charge), et `size_t* size` ou sera placée la taille du tableau résultant. La libération du tableau est laissée au soin du candidat ;
- Tout le reste est comme indiqué dans le sujet.

6.2 C++

- Les tableaux sont représentés par des `std::vector<type>` ;
- Le reste est identique au sujet.

6.3 C#

- Les fonctions à utiliser sont des méthodes statiques de la classe Api. Ainsi, pour utiliser la fonction `Foo`, il faut faire `Api.Foo` ;
- Les noms des fonctions, structures et énumérations sont en `CamelCase`. Ainsi, une fonction nommée `foo_bar` dans le sujet s'appellera `FooBar` en C#.

6.4 Caml

- L'API est fournie par le fichier `api.ml`, qui est `open` par défaut par le fichier à compléter ;
- Les énumérations sont représentées par des types sommes avec des constructeurs sans paramètres. Seule la première lettre des noms des constructeurs est en majuscule ;
- Les structures sont représentées par des records, sauf pour la structure `position` qui est représentée par un couple `int * int` ;
- Les tableaux sont représentés par des `array` Caml classiques.

6.5 Java

- Les fonctions à utiliser sont des méthodes statiques de la classe

`Interface`. Ainsi, pour utiliser la fonction `foo`, il faut faire `Interface.foo` ;
- Les structures sont représentées par des classes dont tous les attributs sont publics.

6.6 Pascal

- Tout ou presque est dans l'API, rien de spécial.

6.7 PHP

- Les constantes sont définies via des `define` et doivent donc être utilisées sans les précéder d'un signe dollar ;
- Les énumérations sont définies comme des séries de constantes. Se référer à la puce au dessus ;
- Les structures sont gérées sous forme de tableaux associatifs. Ainsi, une structure contenant un champ `x` et un champ `y` sera crée comme ceci : `array('x' => 42, 'y' => 1337)`.

6.8 Python

- L'API est fournie par le module `api`, dont tout le contenu est importé par défaut par le code à compléter ;
- Les constantes des énumérations sont représentées par des entiers ;
- Les structures sont représentés par des `namedtuple` Python, dont les champs peuvent être accèdés via la notation pointée habituelle, et qui peuvent être créés comme ceci : `foo(bar=42, x=3)`.