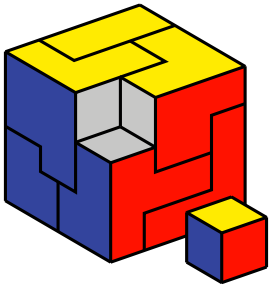
Prologin 2019



Exploitation minière
P.R.O.L.O.G.I.N.

Mine standard de granite standard

Sujet de la finale du Concours National d'Informatique
Vendredi 31 mai 2019

Table des matières


PROLOGIN'19

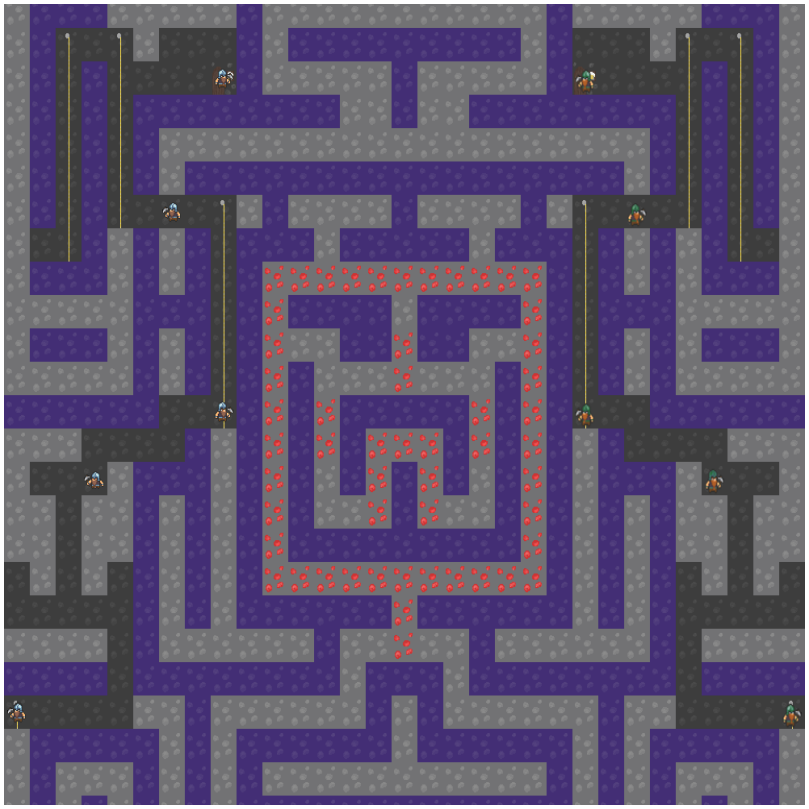
1 Le début des ennuis

Félicitations ! Votre dernière promotion au titre de nain responsable de la mine au P.R.O.L.O.G.I.N. (Profits Rapides Organisés Logiquement, Obtenus Grâce à des Individus Nanesques) est bien méritée. Vous voici donc chargé de superviser l'extraction de minerais rares d'une mine (standard). Malheureusement, la direction ne vous a pas accordé tous les moyens nécessaires [1] ...

Passons donc en revue vos nouvelles conditions d'exercice.

1.1 La mine (standard)

C'est une mine. Parfois il y a des tunnels déjà creusés, d'autres fois non.



1.2 Le minerai

Le granite [2] *est une roche magmatique plutonique à structure grenue, c'est-à-dire entièrement cristallisée, formée par le refroidissement lent et en profondeur d'un*

1. Restriction budgétaire, crise économique, augmentation des taxes...
2. Il ne faut pas confondre « granite » et « granit », le premier désignant une roche spécifique, tandis que le second est un terme commercial utilisé dans l'industrie extractive, indépendamment de sa lithologie

magma issu de la fusion partielle de la croûte continentale. C'est une roche acide composée essentiellement de quartz et de mica[3] .

La région est riche en granite (standard), mais il existe des veines de minerai de plus forte valeur. Vous consulterez avec attention le rapport de votre géologue en chef qui reprend les différents minerais présents sur site.

1.3 Les nains (standard)

Les nains sont des êtres humanoïdes de petite taille avec une barbe, deux bras, deux jambes, un casque et généralement une ou plusieurs choppes de bière dans les mains.

La direction vous a affecté la meilleure catégorie de personnel pour l'exploitation, vous êtes donc à la tête d'une équipe de nains. Le manuel relatif à l'exploitation minière, annexé en fin de document, vous donnera de plus amples détails. Sachez toutefois que les nains ont certaines... particularités, comme des *points d'action*, *de déplacement* et *de vie*.

Pour des questions de restrictions budgétaires, le nombre de nains (standards) est limité.

1.4 De la concurrence pour plus de productivité

La société a décidé de vous mettre en concurrence avec un autre manager de nains, afin d'exploiter au mieux le gisement. Sachez que celui d'entre vous qui fera le plus de profit remportera une très belle prime !

P.R.O.L.O.G.I.N. n'est pas responsable des blessures que subissent ses employés. Elle nie toute implication dans les récentes rumeurs sur les bagarres entre équipes concurrentes, les contusions de pioches relevées n'étant que le produit de malheureux accidents[4].

2 Rapport géologique

2.1 Le granite (standard)

Le granite est une roche magmatique plutonique à structure grenue, c'est-à-dire entièrement cristallisée, formée par le refroi...

Bref, comme vous le savez déjà, le granite (standard) est présent en abondance sur le site. Cependant, il a une valeur marchande de très exactement 0 pièce d'or[5], et **requiert un seul coup de pioche pour être cassé**.

3. ...do
4. Mineur est un métier dangereux, même à plus de 18 ans
5. Avec 20% de taxes, on peut mieux faire en termes de rentabilité.

2.2 Le minerai

En plus du granite (standard), il a été identifié 6 types de minerais différents. Le marché pouvant être particulièrement volatile, leur valeur en pièces d'or varie, de même que le nombre de coups de pioche nécessaires pour les extraire. Les valeurs dépendent de la mine en question, référez-vous au géologue en chef et aux courtiers pour de plus amples informations.

Voici en attendant une liste de ces minerais, avec un échantillon pour les distinguer plus facilement et des approximations de leurs caractéristiques :

Minerai	Échantillon	Rendement
Granite		0
Obsidienne		–
Charbon		1 à 4
Fer		5 à 9
Or		10 à 14
Diamants		15 à 19
Émeraudes		20 à ∞
Rubis		$\mathbb{Z}$

2.3 Divers

Le terrain comporte de nombreuses cavités naturelles, formant parfois des trous profonds. C'est très utile pour descendre vos nains plus rapidement ... dans les deux sens du terme ! Plus ils chuteront de haut, plus ils se blesseront à l'arrivée [6]. Vous trouverez plus de détails sur les risques de chute dans le manuel d'exploitation.

Encore un point de détail : de l'obsidienne a été trouvée en sous-sol. Il est inutile de s'en préoccuper, les nains ne peuvent pas la miner [7] avec le matériel qu'ils ont a disposition.

3 Manuel relatif à l'exploitation minière

Version 13, édition 37, opus 42

6. Nous rappelons que P.R.O.L.O.G.I.N. n'est responsable en aucune manière des blessures reçues par ses employés, et que toute la charge en incombe sur le manager (vous, donc).
7. Et non pas laminer

3.1 Reconnaissance du terrain

Le standard de la compagnie est d'exploiter les gisements depuis la surface, en creusant verticalement. L'exploitation à flanc de montagne n'est pas pratiquée, sauf exception validée par la direction.

Le terrain est donc globalement plat, et l'exploitation se fait globalement verticalement.

Conformément aux articles 16 et suivants de la Convention Collective Applicable, une taverne est mise à disposition de chaque équipe, et déterminera leur point de regroupement principal. Les mesures de sécurité prévoient que les mineurs sans affectation doivent y être présents en permanence. Par dérogation au code du travail, la consommation de bière pendant les horaires de travail est autorisée, en vertu de ses propriétés médicales exceptionnelles : **chaque mineur se présentant à la taverne récupère instantanément la totalité de ses points de vie** [8].

La taverne abrite également une délégation du département trading, qui collecte le minerai extrait.

3.2 Cordages (standard)

Le matériel des nains inclut des cordes (standard) illimitées. Pour les utiliser il convient de disposer d'une poulie en tête de cordage, cela permettra les actions sur la corde.

Ces cordes permettent aux nains de se déplacer au-dessus du vide plus rapidement qu'en s'agrippant aux parois.

Une fois la poulie posée, **la corde descend jusqu'au sol**. Votre équipe de nains s'assurera de prévoir suffisamment de jeu afin que la corde puisse toujours se dérouler au fur et à mesure que de nouveaux blocs de granite (standard) sont creusés en dessous de celle-ci. En revanche, **poser une corde est une entreprise complexe, qui nécessite la totalité des points d'action de toute l'équipe combinée**.

De plus les cordes peuvent être actionnées par un autre nain qui n'est pas sur la corde. En consommant des points d'actions **un nain peut tirer la corde dans un sens ou dans un autre** ce qui déplacera de une case dans un sens ou dans l'autre tous les nains qui sont sur la corde et agrippé. Ce déplacement ne peut pas déplacer un nain hors de la corde et comme d'habitude, un nain bloquera un nain adverse.

8. Ce type de réaction n'a été observée dans nos laboratoires que chez les nains standard adultes. N'essayez pas chez vous, même si vous mesurez moins d'un mètre 50.

3.3 Déplacements

Le personnel minier peut se déplacer dans deux dimensions : verticalement et transversalement, en marchant au sol, en s'agrippant, en chutant ou emporté par une corde.

Un déplacement n'est possible que vers une position libre [9], et nécessite des *points de déplacement*. Une case n'est pas considérée comme libre lorsqu'elle contient du granite, de l'obsidienne, ou un nain (standard) ennemi. De plus, nous rappelons aux nains que les effets de la gravité existent [10], nous vous recommandons donc de lire attentivement la section dédiée.

Il est également à noter que la promiscuité ne pose pas de problème aux mineurs [11], ils peuvent donc se trouver sur une même position. Il est néanmoins nécessaire de se coordonner un minimum avant : ainsi seuls **les mineurs d'une même équipe se tolèrent entre eux**.

Enfin, de par leur activité physique intense, les mineurs ont les épaules larges. Ils peuvent sans problème supporter le poids d'un autre mineur. En application du paragraphe précédent, **un mineur ne gardera jamais un mineur qui est dans son équipe sur ses épaules** car ils peuvent se coordonner afin de se tenir sur la même case.

3.4 Cas particulier des déplacements verticaux

Dans le cas particulier des déplacements verticaux, il est rappelé que la gravité existe, et qu'elle attire inexorablement les corps vers le bas. Les collisions avec le sol entraînent des dégâts exponentiels avec la hauteur de chute, selon la formule ci-dessous.

$$\text{Dégâts} = \begin{cases} 0 & \text{pour } h < 4 \\ 2^{h-4} & \text{pour } h >= 4 \end{cases}$$

avec h la hauteur de chute. Si un nain meurt de chute, les dieux vont immédiatement s'occuper de faire disparaitre son butin [12].

Cependant il est possible d'échapper à ça : **un nain agrippé, libéré de l'emprise de la gravité, peut se déplacer dans toutes les directions sans craindre de chuter**, ce qui lui demandera plus de *points d'action* que de se déplacer au sol. Il y a alors plusieurs manières de se déplacer à distinguer pour un nain agrippé :

9. C'est évident mais ça va toujours mieux en le disant.
10. Nos scientifiques ont rapporté que les nains qui en oubliaient l'existence affichaient un rendement plus faible, voir nul.
11. C'est un critère de recrutement.
12. Il est inscrit dans les registres sacrés : "On subit tellement avec les bêtises des nains qu'on a bien été forcés d'y mettre une taxe".

— déplacement standard (relativement lent)
— déplacement dans une direction occupée par une corde (plus rapide)
— rester agrippé sur une case occupée par une corde actionnée par un autre nain (ce qui peut faire gagner beaucoup de temps si vous avez confiance en l'opérateur de cette corde)

S'agripper à la paroi ne contraint aucune autre action que le déplacement du nain.

Vous ne pouvez pas pousser les nains dans les trous (i.e. les putsch sont interdits).

3.5 Traitement du minerai

Les minerais précieux doivent être extraits par du personnel qualifié. L'extraction consomme des points d'action, et peut nécessiter plusieurs coups de pioche en fonction du minerai extrait.

Comme les nains n'ont pas de problèmes à se tenir côte à côte, il est possible de paralléliser l'extraction avec plusieurs mineurs, de manière à accélérer celle-ci.

Une fois extrait, le minerai est stocké temporairement par le personnel. Afin d'être pris en compte par le département trading (et être mis en sécurité...), **le minerai doit être rapporté à la taverne**.

Notez que comme **les nains ne peuvent porter qu'une quantité limitée de minerai** [13], ils devront régulièrement faire un retour à la taverne pour se décharger. Si un nain extrait du minerai qu'il ne peut pas porter, il détruit immédiatement ce qu'il y a de trop pour éviter que ça ne tombe entre de mauvaises mains [14].

3.6 Cas particulier des rencontres malveillantes

Il est rappelé à l'ensemble du personnel que la pioche (standard) qui est donnée à chaque nain, peut également être considérée comme une arme de quatrième catégorie... Son usage à cette fin est explicitement ignoré par la direction, à la discrétion de chacun. Dans ce cas, la pioche se manie de manière identique à son utilisation standard. À noter que **si plusieurs nains sont sur la même position et que l'un d'entre eux est attaqué, ils sont tous blessés** [15] !

Un soin particulier sera accordé aux *points de vie* des intervenants. Lorsque

13. Tous leurs points de compétences dédiés ayant été investis dans le transport de bière.
14. Ami ou ennemi, un nain n'est jamais trop prudent.
15. Striiike !

ce compteur arrive à zéro, un point de destin [16] est automatiquement utilisé. Le mineur ainsi ressuscité regagne la mine dans la taverne du chantier à son prochain tour, mais ayant perdu tout son butin sur le coup ...

Le nain qui a donné le coup de pioche s'empresse alors de récupérer le butin des nains éliminés, dans la limite de la *capacité* qu'il a à le transporter, comme expliqué dans la section précédente. Il détruira donc tout ce qu'il n'arrivera pas à emporter avec lui.

Un nain qui a l'audace de s'égarer sur la case occupée par la taverne du joueur adverse sera mal accueilli, et verra ses points de vie réduire à 0 [17], toutes les richesses qu'il transportait seront ramassées par l'équipe ennemie.

3.7 Fin de la période d'essai

À la fin de 100 tours de jeu, le joueur qui a ramené le plus de pièces d'or à sa taverne gagne. Les pièces d'or encore portées dans l'inventaire de nains dispersés dans la map ne sont pas comptabilisées.

16. La direction assure qu'un nain aura toujours un point de destin disponible pour lui dans cette situation, au risque d'un prélèvement de salaire.
17. On ne déconne pas avec la bière...

4 Tournois

4.1 Tournois intermédiaires

Afin de vous aider à perfectionner vos algorithmes, des tournois intermédiaires vous seront proposés. Ces matchs n'ont absolument aucune influence sur le classement final, mais sont néanmoins à prendre au sérieux, car ils vous permettront de vous situer par rapport aux autres joueurs, de connaître vos ennemis, vos points forts et vos faiblesses, et vous donneront des pistes pour vous améliorer pendant la finale. L'annonce des tournois intermédiaires se fera dans l'heure qui précède.

Pour chaque tournois, nous prendrons le dernier champion que chaque candidat aura envoyé sur le site de soumission pour le faire participer au tournoi, et nous vous donnerons les résultats ainsi que votre progression dès que les tournois se seront terminés, avec un récapitulatif de votre progression globale. Les tournois seront exécutés sur des cartes officielles de notre choix, qui seront potentiellement amenées à changer au fur et à mesure.

4.2 Rendu final

Le tournoi final aura lieu **samedi à 23 h 42**.

Le rendu final est le seul rendu qui comptera pour le classement. Les mêmes règles s'appliquent : le dernier champion soumis à l'heure du début du tournoi sera le champion utilisé pour le tournoi final.

Lors du tournoi final, plusieurs cartes seront ajoutées. Celles-ci resteront inconnues de tous les joueurs jusqu'à la fin du concours, afin de mesurer l'adaptabilité de vos algorithmes à des situations inconnues.

Pour le rendu final, nous vous demandons de rajouter des commentaires qui résument le fonctionnement des différents blocs logiques de votre code, ainsi qu'un **commentaire global en haut de votre fichier principal** qui détaille votre stratégie ainsi que les différents algorithmes que vous avez employés pour l'implémenter.

5 Considérations techniques

Vous disposez d'une seconde (temps réel !) à chaque fois qu'une de vos fonctions est appelée pour rendre la main. Passé ce délai, votre programme est tué, le match continue sans vous et vos fonctions ne sont plus appelées. Il n'est pas possible de revenir en jeu tout simplement parce qu'il n'y a aucun moyen de rétablir l'état des environnements des langages après une interruption. Les limites de mémoire sont faites avec des `cgroups`, ce qui fait que l'allocation

échouera si vous essayez de dépasser la limite qui vous est accordée. Cette limite compte aussi la taille de la pile.

D'autres limitations sont appliquées :

— le système de fichiers est entièrement en lecture seule ;
— seuls `/usr`, `/var` et `/tmp` sont montés ;
— vous n'avez pas le droit d'utiliser des processus en parallèle ;
— la mémoire est limitée à 500 Mio ;
— la taille totale de votre output ne doit pas dépasser 256 Kio (elle sera tronquée à partir de cette limite) ;
— le temps d'exécution total du processus est limité à 300 secondes de temps réel ;
— chaque appel de fonction est limité à une seconde de temps réel plus 500 millisecondes de marge pour prendre en compte le surcoût de sérialisation/désérialisation des valeurs depuis et vers les langages cibles.

6 API

Constante : TAILLE_MINE
Valeur : 31
Description : Taille de la mine (longueur et largeur).

Constante : NB_TOURS
Valeur : 100
Description : Nombre de tours à jouer avant la fin de la partie.

Constante : NB_POINTS_DEPLACEMENT
Valeur : 5
Description : Nombre de points de déplacement par tour par nain (standard).

Constante : NB_POINTS_ACTION
Valeur : 6
Description : Nombre de points d'action par tour par nain (standard).

Constante : VIE_NAIN
Valeur : 10
Description : Nombre de points de vie d'un nain (standard).

Constante : NB_JOUEURS
Valeur : 2
Description : Nombre de joueurs.

Constante : NB_NAINS
Valeur : 6
Description : Nombre de nains (standard) par joueur.

Constante : DEGAT_PIOCHE
Valeur : 3
Description : Dégât infligé par un coup de pioche à un nain (standard).

Constante : BUTIN_MAX
Valeur : 25
Description : Valeur cumulée maximale des minerais qu'un nain (standard) peut emporter avec lui.

Constante : COUT_DEPLACEMENT
Valeur : 1
Description : Nombre de points de déplacement pour qu'un nain (standard) se déplace d'une case.

Constante : COUT_ESCALADER
Valeur : 2
Description : Nombre de points de déplacement pour qu'un nain (standard) se déplace d'une case lorsqu'il est agrippé.

Constante : COUT_ESCALADER_CORDE
Valeur : 1
Description : Nombre de points de déplacement pour qu'un nain (standard) se déplace vers une case occupée par une corde lorsqu'il est agrippé.

Constante : COUT_MINER
Valeur : 6
Description : Nombre de points d'action pour qu'un nain (standard) mine un bloc.

Constante : COUT_TIRER
Valeur : 1
Description : Nombre de points d'action pour qu'un nain (standard) tire sur une corde.

Constante : COUT_LACHER
Valeur : 0
Description : Nombre de points d'action pour qu'un nain (standard) lâche la paroi.

Constante : COUT_AGRIPPER
Valeur : 0
Description : Nombre de points d'action pour qu'un nain (standard) s'agrippe à la paroi.

- **case_type**

Description : Types de cases
Valeurs :

LIBRE :	Case libre, qui peut abriter une corde et des nains (standard)
GRANITE :	Granite (standard), qui peut cacher du minerai
OBSIDIENNE :	Obsidienne
ERREUR_CASE :	Erreur

- **direction**

Description : Direction
Valeurs :

HAUT :	Direction : haut
BAS :	Direction : bas
GAUCHE :	Direction : gauche
DROITE :	Direction : droite
ERREUR_DIRECTION :	Erreur

- **erreur**

Description : Erreurs possibles
Valeurs :

OK :	L'action s'est effectuée avec succès.
PA_INSUFFISANTS :	Votre nain (standard) ne possède pas assez de points d'action pour réaliser cette action.
PM_INSUFFISANTS :	Votre nain (standard) ne possède pas assez de points de déplacement pour réaliser ce déplacement.
HORS_LIMITES :	L'action est en dehors des limites de la mine.
DIRECTION_INVALIDE :	La direction spécifiée n'existe pas, ou vous n'êtes pas autorisé à cibler cette direction pour cette action.
ID_NAIN_INVALIDE :	Le nain (standard) spécifié n'existe pas.
OBSTACLE_MUR :	La position spécifiée est un mur.
OBSTACLE_NAIN :	La position spécifiée est un nain (standard) adverse.
OBSTACLE_CORDE :	Il y a déjà une corde dans la direction spécifiée.
PAS_DE_CIBLE :	Il n'y a pas de nain (standard) ni de granite (standard) sur la position spécifiée.
NAIN_MORT :	Le nain (standard) spécifié est mort.
PAS_ACCROCHE :	Le nain (standard) n'est pas accroché.
DEJA_ACCROCHE :	Le nain (standard) est déjà accroché.
PAS_DE_CORDE :	Il n'y a pas de corde dans la direction spécifiée.
DRAPEAU_INVALIDE :	Le drapeau spécifié n'existe pas.

- **action_type**

Description : Types d'actions
Valeurs :

ACTION_DEPLACER :	Action "deplacer"
ACTION_LACHER :	Action "lacher"
ACTION_MINER :	Action "miner"
ACTION_POSER_CORDE :	Action "poser_corde"
ACTION_TIRER :	Action "tirer"
ACTION_AGRIPPER :	Action "agripper"

- **debug_drapeau**

Description : Types de drapeaux de débug
Valeurs :

AUCUN_DRAPEAU :	Aucun drapeau, enlève le drapeau présent
DRAPEAU_BLEU :	Drapeau bleu
DRAPEAU_VERT :	Drapeau vert
DRAPEAU_ROUGE :	Drapeau rouge

- **position**

```
struct position {
    int ligne;
    int colonne;
};
```

Description : Position dans la mine, donnée par deux coordonnées.
Champs :
- *ligne* : Coordonnée : ligne
- *colonne* : Coordonnée : colonne

• **minerai**

```
struct minerai {
    int resistance;
    int rendement;
};
```

Description : Minerai à récolter
Champs :
- *resistance* : Nombre de coups de pioches encore nécessaires avant que le bloc de minerais ne casse
- *rendement* : Valeur marchande du bloc de minerai

• **nain**

```
struct nain {
    position pos;
    int vie;
    int pa;
    int pm;
    bool accroche;
    int butin;
};
```

Description : Nain (standard)
Champs :
- *pos* : Position actuelle du nain (standard)
- *vie* : Point(s) de vie restant du nain (standard)
- *pa* : Point(s) d'action restant du nain (standard)
- *pm* : Point(s) de déplacement restant du nain (standard)
- *accroche* : Le nain (standard) est accroché à la paroi ou à une corde
- *butin* : Valeur marchande totale que le nain (standard) possède

• **action_hist**

```
struct action_hist {
```

```
    action_type atype;
    int id_nain;
    direction dir;
    direction sens;
};
```

Description : Action de déplacement représentée dans l'historique.
Champs :
- *atype* : Type de l'action
- *id_nain* : Numéro du nain (standard) concerné par l'action
- *dir* : Direction visée par le nain (standard) durant le déplacement
- *sens* : Sens de l'action, utilisé uniquement pour préciser si l'on doit tirer une corde vers le bas ou vers le haut. Direction doit cibler la droite ou la gauche.

• **chemin**

```
direction array chemin(position pos1, position pos2)
```

Description : Renvoie un chemin entre deux positions de la mine sous la forme d'une suite de directions à emprunter. Ce chemin minimise le nombre de blocs de granite nécessaire à casser. Si la position est invalide ou qu'il n'existe pas de tel chemin, le chemin renvoyé est vide.
Parametres :
- *pos1* : Position de départ
- *pos2* : Position d'arrivée

• **deplacer**

```
erreur deplacer(int id_nain, direction dir)
```

Description : Déplace le nain (standard) "id_nain" d'une case dans la direction choisie.
Parametres :
- *id_nain* : Numéro du nain (standard)
- *dir* : Direction visée

• **lacher**

```
erreur lacher(int id_nain)
```

Description : Le nain (standard) "id_nain" lâche la paroi.
Parametres :
- *id_nain* : Numéro du nain (standard)

• **agripper**

```
erreur agripper(int id_nain)
```

Description : Le nain (standard) “id_nain“ s’agrippe à la paroi.
Parametres : *id_nain* : Numéro du nain (standard)

• **miner**

```
erreur miner(int id_nain, direction dir)
```

Description : Le nain (standard) “id_nain“ mine le bloc ou le nain (standard) dans la direction indiquée.
Parametres : *id_nain* : Numéro du nain (standard)
dir : Direction visée

• **tirer**

```
erreur tirer(int id_nain, direction dir_corde, direction sens)
```

Description : Le nain (standard) “id_nain“ tire la corde dans le sens donné (HAUT ou BAS).
Parametres : *id_nain* : Numéro du nain (standard)
dir_corde : Direction dans laquelle se trouve la corde
sens : Sens dans lequel le nain tire sur la corde

• **poser_corde**

```
erreur poser_corde(int id_nain, direction dir)
```

Description : Le nain (standard) “id_nain“ pose une corde dans la direction indiquée.
Parametres : *id_nain* : Numéro du nain (standard)
dir : Direction visée

• **debug_afficher_drapeau**

```
erreur debug_afficher_drapeau(position pos, debug_drapeau drapeau)
```

Description : Affiche le drapeau spécifié sur la case indiquée.
Parametres : *pos* : Case choisie
drapeau : Drapeau à afficher sur la case

• **type_case**

```
case_type type_case(position pos)
```

Description : Renvoie le type d'une case donnée.
Parametres : *pos* : Case choisie

• **liste_cordes**

```
position array liste_cordes()
```

Description : Renvoie la liste de toutes les positions occupées par une corde dans la mine.

• **corde_sur_case**

```
bool corde_sur_case(position pos)
```

Description : Indique si une corde se trouve sur une case donnée.
Parametres : *pos* : Case choisie

• **nain_sur_case**

```
int nain_sur_case(position pos)
```

Description : Renvoie le numéro du joueur à qui appartienent les nains (standard) sur la case indiquée. Renvoie -1 s'il n'y a pas de nain (standard) ou si la position est invalide.
Parametres : *pos* : Case choisie

• **info_nain**

```
nain info_nain(int id_joueur, int id_nain)
```

Description : Renvoie la description du nain (standard) désigné par le numéro "id_nain" appartenant au joueur "id_joueur". Si le nain (standard) n'est pas présent sur la carte, tous les membres de la structure "nain" renvoyée sont initialisés à -1 (et le champ "accroche" à 'false').
Parametres : *id_joueur* : Numéro du joueur
id_nain : Numéro du nain (standard)

• **liste_minerais**

```
position array liste_minerais()
```

Description : Renvoie la liste de tous les minerais dans la mine.

• **info_minerai**

```
minerai info_minerai(position pos)
```

Description : Renvoie la description d'un minerai en fonction d'une position donnée. Si le minerai n'est pas présent sur la carte, ou si la position est invalide, tous les membres de la structure "minerai" renvoyée sont initialisés à -1.

Parametres : *pos* : Case choisie

• **cout_de_deplacement**

```
int cout_de_deplacement(int id_nain, direction dir)
```

Description : Renvoie le nombre de points de déplacement que coûterai le déplacement d'un nain (standard) dans une direction donnée. Renvoie -1 si le déplacement n'est pas possible.

Parametres : *id_nain* : Numéro du nain (standard)
dir : Direction visée

• **position_taverne**

```
position position_taverne(int id_joueur)
```

Description : Renvoie la position de la taverne appartenant au joueur "id_joueur". Si le joueur n'existe pas, renvoie la position (-1, -1).

Parametres : *id_joueur* : Numéro du joueur

• **historique**

```
action_hist array historique()
```

Description : Renvoie la liste des actions effectuées par l'adversaire durant son tour, dans l'ordre chronologique. Les actions de débug n'apparaissent pas dans cette liste.

• **score**

```
int score(int id_joueur)
```

Description : Renvoie le score du joueur “id_joueur“. Renvoie -1 si le joueur est invalide.
Parametres : *id_joueur* : Numéro du joueur

• **moi**

```
int moi()
```

Description : Renvoie votre numéro de joueur.

• **adversaire**

```
int adversaire()
```

Description : Renvoie le numéro de joueur de votre adversaire.

• **annuler**

```
bool annuler()
```

Description : Annule la dernière action. Renvoie faux quand il n’y a pas d’action à annuler ce tour ci.

• **tour_actuel**

```
int tour_actuel()
```

Description : Retourne le numéro du tour actuel.

7 Notes sur l'utilisation de l'API

7.1 C

— Les booléens sont représentés par le type `bool`, défini par le standard du C99, et que l'on retrouve dans le header `stdbool.h`;
— Les fonctions prenant des tableaux en paramètres et retournant des tableaux utilisent à la place de ces tableaux une structure `type_array`, où `type` est le type des données dans le tableau. Ces structures contiennent deux éléments : les données, `type* datas`, et la taille, `size_t length`. Dans tous les cas, la libération des données est laissée au soin du candidat;
— Tout le reste est comme indiqué dans le sujet.

7.2 C++

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

7.3 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#.

7.4 Haskell

— L'API est fournie par le module `Api`.
— Les énumérations sont représentées par des types sommes, les structures par des records. Seule la première lettre des noms de types et de constructeurs est en majuscule. Le nom du constructeur d'une structure est son nom de type.
— La commande `make doc` permet de générer la documentation dans le fichier `doc/index.html` pour votre code ainsi que pour l'API.
— Pour pouvoir conserver des valeurs entre différents appels à vos fonctions à compléter, il faut utiliser des variables mutables :

```
import Data.IORef
import System.IO.Unsafe (unsafePerformIO)

-- La pragma NOINLINE est importante !
-- MonType ne doit pas etre polymorphe !
{-# NOINLINE maVariable #-}
```

```
maVariable :: IORef MonType
maVariable = unsafePerformIO (newIORef maValeurInitiale)

fonctionACompleter :: IO ()
fonctionACompleter = do
  maValeur <- readIORef maVariable
  ...
  writeIORef maVariable maValeur'
```

7.5 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.

7.6 OCaml

— 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.

7.7 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 créée comme ceci : `array('x' => 42, 'y' => 1337)`.

7.8 Python

— L'API est fournie par le module `api`, dont tout le contenu est importé par défaut par le code à compléter ;

— Les énumérations sont représentées par des `IntEnum` Python, qui peuvent-être utilisés comme ceci : `nom_enum.CHAMP`.;
— Les structures sont représentées par des `namedtuple` Python, dont on peut accéder aux champs via la notation pointée habituelle, et qui peuvent être créés comme ceci : `foo(bar=42, x=3)`, sauf pour la structure `position` qui est représentée par un couple `(x, y)`.

7.9 Rust

— L'API est fournie par le module `api`, dont tout le contenu est importé par défaut par le code à compléter.;
— Les noms des structures et énumérations sont en `CamelCase`. Ainsi, une structure nommée `foo_bar` dans le sujet s'appellera `FooBar` en Rust.
— Les tableaux sont représentés par des `Vec<T>`.