**Langage de programmation avancé**

**HAUTE ÉCOLE DE NAMUR-LIÈGE-LUXEMBOURG**

**DA-IA – bloc 1**

Visite de donjon

Objectifs

* Continuer les bonnes pratiques acquises lors des exercices précédents

Déclarer et utiliser des énumérations

Déclarer et utiliser des structures

Manipuler des opérations bit à bit (bitwise operations)

# Visite de donjon

Le but de cet exercice est de représenter en mémoire et d’afficher à l’écran des « donjons » constitués de salles formant potentiellement un labyrinthe et comportant des trésors et des créatures hostiles.

A black background with white lines and letters

Description automatically generated

## Étape 1 : contenu d’une salle

Commencez par définir une énumération simple permettant d’indiquer le contenu d’une salle et donnez-lui le synonyme Contenu. On suppose qu’une salle ne peut contenir qu’un seul contenu à la fois. Cette énumération doit reprendre les valeurs suivantes :

* CONTENU\_RIEN, pour une salle vide ;
* CONTENU\_TRESOR, pour une salle comportant un trésor ;
* CONTENU\_KOBOLD, pour une salle gardée par un kobold ;
* CONTENU\_GOBELIN, pour une salle gardée par un gobelin.

|  |  |
| --- | --- |
|  |  |
| *Un kobold (illustration par Damien Mammoliti, (c) Paizo)* | *Un gobelin (illustration par Kiki Moch Rizky, (c) Paizo)* |

Écrivez une fonction symboleContenu qui reçoit une valeur de cette énumération et renvoie un caractère représentant le contenu en question (à savoir une lettre parmi R, T, K et G).

Par exemple, l’appel symboleContenu(CONTENU\_KOBOLD) doit renvoyer le caractère K.

## Étape 2 : connexions d’une salle

A blue star with white text

Description automatically generatedOn suppose que chaque salle a une forme carrée et possède des murs parfaitement alignés avec les axes nord-sud et ouest-est. Chaque salle peut donc posséder au plus 4 sorties (appelées connexions) : une vers le nord, une vers l’est, une vers le sud et une vers l’ouest.

Contrairement à ce qui se passe pour les contenus, une salle peut avoir plusieurs sorties. Dans l’énumération citant les connexions possibles, il faut donc indiquer des valeurs qui pourront être combinées et dont chaque combinaison pourra être déchiffrée sans ambiguïté.

Par exemple, les associations de valeurs suivantes sont problématiques, car une valeur de 3 pourrait indiquer soit une unique connexion vers le sud soit deux connexions (une vers le nord et une vers l’est).

|  |  |
| --- | --- |
| **Connexion** | **Valeur** |
| CONNEXION\_NORD | 1 |
| CONNEXION\_EST | 2 |
| CONNEXION\_SUD | 3 |
| CONNEXION\_OUEST | 4 |

Pour pouvoir utiliser facilement les opérateurs bit-à-bit qui fonctionnent au niveau de la représentation binaire des entiers, il vaut donc mieux utiliser des valeurs qui correspondent à des puissances de 2. Avec les valeurs suivantes, toute combinaison numérique peut être décomposée sans ambiguïté possible : par exemple, une valeur de 7 ne peut correspondre qu’à la combinaison 1 + 2 + 4.

|  |  |
| --- | --- |
| **Connexion** | **Valeur** |
| CONNEXION\_NORD | 1 |
| CONNEXION\_EST | 2 |
| CONNEXION\_SUD | 4 |
| CONNEXION\_OUEST | 8 |
| CONNEXION\_AUCUNE | 0 |

Définissez une énumération pour les connexions et donnez-lui le synonyme Connexions (ici, au pluriel vu qu’une valeur représente *toutes* les connexions permettant de sortir d’une salle).

Pour bien vous rendre compte de comment ça fonctionne, affichez à l’écran les valeurs numériques des variables suivantes et assurez-vous de bien comprendre le fonctionnement et l’utilité de l’opérateur bit-à-bit |.

Connexions connexions1 = CONNEXION\_OUEST;

Connexions connexions2 = CONNEXION\_EST | CONNEXION\_SUD;

Connexions connexions3 = CONNEXION\_NORD | CONNEXION\_SUD | CONNEXION\_OUEST;

Connexions connexions4 = CONNEXION\_AUCUNE;

## Étape 3 : analyser les connexions d’une salle

Affichez à l’écran les valeurs des expressions suivantes (sous forme numérique par exemple) et assurez-vous de bien comprendre le fonctionnement et l’utilité de l’opérateur bit-à-bit &.

connexions1 & CONNEXION\_OUEST

connexions1 & CONNEXION\_EST

connexions2 & CONNEXION\_EST

Ensuite, utilisez cet opérateur pour écrire une fonction afficheConnexions qui reçoit une valeur de type Connexions et affiche en clair la liste des connexions correspondantes ; par exemple, « Aucune connexion » ou « Nord Sud Ouest » dans le cas des deux derniers exemples donnés à la fin de l’étape précédente.

Testez votre fonction sur les quatre exemples donnés dans l’étape 2.

## Étape 4 : représenter les salles

Une salle du donjon est décrite par deux éléments : ses connexions et son contenu (chacun de ces éléments correspondant à l’une des deux énumérations que vous avez définies).

Créez une structure pour les salles et donnez-lui le synonyme Salle.

Écrivez une fonction décritSalle qui reçoit une salle et affiche le caractère correspond à son contenu ainsi que la liste de ses connexions.

Vous pouvez utiliser les exemples suivants pour tester votre fonction.

Salle salle1 = { CONNEXION\_EST | CONNEXION\_OUEST, CONTENU\_TRESOR };

Salle salle2 = { CONNEXION\_NORD, CONTENU\_GOBELIN };

Salle salle3 = { CONNEXION\_NORD | CONNEXION\_EST | CONNEXION\_SUD, CONTENU\_RIEN };

## Étape 5 : dangers et trésors

Définissez les fonctions estDanger et estTresor qui reçoivent un contenu et indiquent respectivement s’il s’agit d’un danger (c’est-à-dire d’une créature à affronter) et si la salle contient un trésor.

Définissez une fonction afficheContenu qui reçoit un contenu et affiche à l’écran le caractère qui le représente. S’il s’agit d’un danger, le caractère est écrit en rouge. S’il s’agit d’un trésor, il est écrit en jaune. Dans les autres cas, il est écrit dans la couleur normale.

Pour modifier la couleur d’écriture, utilisez les fonctions ci-dessous, que vous ajouterez à votre programme.

void changeCouleurEnJaune(void) { printf("\033[1;33m"); }

void changeCouleurEnRouge(void) { printf("\033[1;31m"); }

void changeCouleurEnBleu(void) { printf("\033[1;34m"); }

void changeCouleurEnNormal(void) { printf("\033[1;0m"); }

Les trois premières fonctions permettent de changer la couleur d’affichage de tous les *futurs* printf. Vous devez donc les appeler avant d’effectuer l’affichage à colorer. Après l’affichage en question, n’oubliez pas d’appeler la quatrième fonction qui permet de revenir à la couleur standard pour la suite.

Testez votre fonction sur les trois exemples de salle donnés dans l’étape précédente.

## Étape 6 : donjons

A black background with white lines and red letters

Description automatically generatedDans un premier temps, on suppose qu’un donjon est un agencement de salles (carrées) formant une structure carrée. Un donjon est décrit par deux éléments :

* sa taille, un entier qui correspond au nombre de salles en largeur et en hauteur ;
* ses salles, un tableau de salles à deux indices (les indices fonctionnant comme un système de coordonnées).

Définissez une structure pour représenter les donjons et donnez-lui le synonyme Donjon.

Voici un exemple de déclaration de donjon que votre code doit pouvoir supporter et correspondant à la représentation ci-contre.

Donjon donjonTest = {

3,

{

{

{ CONNEXION\_EST | CONNEXION\_SUD, CONTENU\_RIEN },

{ CONNEXION\_OUEST | CONNEXION\_EST, CONTENU\_KOBOLD },

{ CONNEXION\_OUEST, CONTENU\_TRESOR }

},

{

{ CONNEXION\_NORD | CONNEXION\_SUD, CONTENU\_RIEN },

{ CONNEXION\_EST, CONTENU\_TRESOR },

{ CONNEXION\_OUEST | CONNEXION\_SUD, CONTENU\_GOBELIN }

},

{

{ CONNEXION\_EST | CONNEXION\_NORD, CONTENU\_KOBOLD },

{ CONNEXION\_OUEST | CONNEXION\_EST, CONTENU\_RIEN },

{ CONNEXION\_OUEST | CONNEXION\_NORD, CONTENU\_TRESOR }

},

}

};

Pour vérifier que la structure correspond à cet exemple, ajouter cette déclaration à la fonction principale et passez à l’étape suivante.

## Étape 7 : afficher une salle

Dans la représentation du donjon, chaque salle sera affichée sous la forme d’un carré de 5 caractères de large et de 3 caractères de haut (oui, 5 × 3 ne forme pas un carré mais, visuellement, c’est ce qui s’en rapproche le plus).

Le format de base pour représenter une salle sera le suivant :

/---\

| |

\---/

Dans la première et la troisième ligne, les coins de la salle sont représentés par des barres obliques. Les trois tirets --- à la première ligne correspond au mur nord ; ceux de la troisième ligne, au mur sud. La première barre verticale | de la 2e ligne représente le mur ouest et la seconde barre, le mur est.

En fonction des connexions de la salle, certains de ces murs devraient être remplacés par des espaces. Ainsi, dans le cas d’une salle comportant des connexions vers le nord et vers l’est, l’affichage produit doit être le suivant (où les « murs » sont remplacés par des espaces) :

/ \

|

\---/

Finalement, au milieu de la salle, l’affichage doit indiquer le caractère correspondant au contenu de la salle. Pour éviter de surcharger inutilement la représentation avec des « R » indiqués dans chacune des salles vides, modifiez la fonction symboleContenu pour qu’elle renvoie un espace dans ce cas-là.

L’objectif de cette étape est de définir 4 fonctions pour réaliser cet affichage :

* afficheLigne1, afficheLigne2 et afficheLigne3, recevant chacune une salle et produisant l’affichage de la ligne indiquée ;
* une fonction afficheSalle qui se charge d’afficher l’entièreté de la salle.

## Étape 8 : afficher un donjon

En réutilisant judicieusement les fonctions précédentes, et en en définissant éventuellement de nouvelles, écrivez une fonction afficheDonjon qui réalise l’affichage de l’entièreté du donjon (au format présenté dans les captures montrées plus haut dans ce document).

## Étape 9 : créer un donjon

L’objectif de cette étape et des suivantes est de définir des fonctions qui permettent d’encoder plus facilement un donjon :

* une fonction donjonVideCarre qui reçoit la taille (nombre de salles sur un côté) et renvoie un donjon complet de la taille en question et constitué de salles vides et sans connexion ;
* une fonction donjonAvecConnexions qui reçoit un donjon déjà construit, une connexion (par exemple CONNEXION\_EST) et un tableau reprenant des numéros de salles ainsi que sa taille et ajoute à ces salles la connexion indiquer (par exemple ajouter une connexion vers l’est aux salles citées) ;
* une fonction donjonAvecContenu qui fonctionne de manière similaire à la précédente mais reçoit un contenu, qui sera placé dans les salles citées.

Chacune de ces trois fonctions renvoit comme résultat un nouveau donjon ou le donjon modifié.

Dans le cas des deux dernières fonctions, les salles sont numérotées « ligne par ligne » en commençant par 0. Ainsi, dans un donjon de 3 salles × 3 salles, les salles de la 2e ligne sont numérotées 3, 4 et 5, et la dernière salle dans le coin sud-est du donjon porte le numéro 8.

Votre code doit alors vous permettre d’exécuter la fonction suivante, qui définit un donjon 5 × 5 correspondant à la capture d’écran présentée au début de ce document.

Donjon monDonjon(void) {

int sallesAvecSortieEst[] = { 0, 1, 2, 5, 8, 11, 13, 15, 16, 17, 20, 22, 23 };

int sallesAvecSortieOuest[] = { 1, 2, 3, 6, 9, 12, 14, 16, 17, 18, 21, 23, 24 };

int sallesAvecSortieNord[] = { 5, 7, 8, 9, 10, 12, 13, 15, 16, 19, 21, 22, 24 };

int sallesAvecSortieSud[] = { 0, 2, 3, 4, 5, 7, 8, 10, 11, 14, 16, 17, 19 };

int sallesAvecTresor[] = { 4, 6, 18, 20 };

int sallesAvecKobold[] = { 5, 9, 17, 21 };

int sallesAvecGobelin[] = { 2, 15, 24 };

Donjon donjon = donjonVideCarre(5);

donjon = donjonAvecConnexions(donjon, CONNEXION\_EST, sallesAvecSortieEst, 13);

donjon = donjonAvecConnexions(donjon, CONNEXION\_OUEST, sallesAvecSortieOuest, 13);

donjon = donjonAvecConnexions(donjon, CONNEXION\_SUD, sallesAvecSortieSud, 13);

donjon = donjonAvecConnexions(donjon, CONNEXION\_NORD, sallesAvecSortieNord, 13);

donjon = donjonAvecContenu(donjon, CONTENU\_TRESOR, sallesAvecTresor, 4);

donjon = donjonAvecContenu(donjon, CONTENU\_KOBOLD, sallesAvecKobold, 4);

donjon = donjonAvecContenu(donjon, CONTENU\_GOBELIN, sallesAvecGobelin, 3);

return donjon;

}

## [Dépassement] Étape 10 : extensions possibles

* A black background with white lines and letters

  Description automatically generatedAjouter deux nouveaux types de connexions, vers le haut et vers le bas. Dans la représentation des salles/donjons, ces passages pourraient être indiqués par des caractères « v » et « accent circonflexe » placés sur la 2e ligne, avant/après le symbole du contenu de la salle.
* Ajouter deux nouveaux types de contenu : CONTENU\_OBJET\_MAGIQUE (représenté par O et classifié comme un trésor) et CONTENU\_DRAGON (représenté par D et classifié comme un danger).
* Modifier la définition d’un donjon pour permettre de réaliser des donjons « rectangulaires » (avec des nombres différents de salles en largeur et en hauteur). Ajouter une fonction donjonVide similaire à donjonVideCarre mais permettant d’indiquer la largeur et la hauteur du donjon désiré.
* Retoucher les fonctions donjonAvecConnexions et donjonAvecContenu en remplaçant les arguments correspondant à la liste des numéros de salle et au nombre de numéros cités par une structure ListeNuméros regroupant ces deux informations. Utiliser cette structure pour exprimer le résultat d’une fonction permettant d’obtenir les numéros des salles qui contiennent un trésor.