**Langage de programmation avancé**

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

**DA-IA – bloc 1**

Exercice 0 – Tableaux à deux indices

Objectifs

* Continuer les bonnes pratiques acquises lors des exercices précédents

Déclarer et utiliser des tableaux à deux indices

# Les portes des étoiles…

Le but de cet exercice est de manipuler des tableaux « double indice ».

**Commencez par faire un diagramme d'actions et faites-le valider par le professeur !**

Dans la série Stargate SG1, chaque destination/planète a une adresse unique qui est une combinaison de 7 symboles.

Le Colonel Jack O'Neill et son équipe, après de nombreuses missions, voudraient tenir un répertoire des différentes planètes visitées en mémorisant, pour chacune d'elles, son adresse.

Par simplicité, chaque symbole est représenté par un identifiant entier selon la table ci-dessous :



Il est à noter que le dernier de ces 7 symboles est celui de la planète d'origine, dans notre cas, la terre. Il n'est donc pas nécessaire de le mémoriser.

Pour mémoriser les adresses, on utilise un tableau à double indices adressesSG. Ce tableau comportera au plus 5000 adresses/lignes, chaque adresse ayant 6 symboles/colonnes.

**adressesSG**

**adresse**

*5000 cellules*

**symbole**

*6 cellules*

## Étape 1 : obtenir l’adresse d’une planète

Écrire une fonction qui permet d’obtenir les symboles d’une adresse.

┌────────────────┐↓ nbSymboles

│ adresseObtenue │

└────────────────┘↓ adresse

*Précondition* : nbSymboles (≥ 0) est le nombre de symboles nécessaires pour une adresse (dans l’exemple, il est de 6).

*Postcondition* : adresse est un tableau rassemblant les nbSymboles (≥ 0) symboles obtenus de l’utilisateur au clavier (et correspondant donc à une planète).

## Étape 2 : afficher l’adresse d’une planète

Écrire une fonction qui permet d’afficher les symboles d’une adresse.

┌────────────────┐↓ adresse, nbSymboles

│ afficheAdresse │

└────────────────┘

*Précondition* : adresse contient les nbSymboles (≥ 0) symboles constituant l’adresse d’une planète.

*Postcondition* : l’adresse de la planète est affichée à l’écran au format suivant (utilisez "%2d " pour chaque symbole).

27 07 15 13 12 30

## Étape 3 : fonction principale

Dans la fonction principale,

* déclarez le tableau adressesSG qui contiendra les adresses des planètes visitées,
* déclarez le tableau nomsSG qui contiendra les noms des planètes visitées, dans le même ordre que celui des adresses.

N’oubliez pas d’utiliser des constantes symboliques pour les valeurs constantes.

Écrivez ensuite les instructions qui permettent de garnir les deux tableaux ci-dessus, avec, respectivement, les coordonnées et le nom des planètes obtenues les unes à la suite des autres.

Après chaque obtention d’une planète, affichez son adresse en faisant appel à la fonction de l’étape 2.

Afin de savoir s'il y a une autre planète à mémoriser, posez la question "Avez-vous visité une autre planète ? (O/N) Réponse : ".

Comptez également le nombre d'adresses nbPlanetes obtenues afin d'éviter de parcourir les 5000 lignes inutilement dans la suite.

## Étape 4 : afficher les planètes

Écrire une fonction qui permet d’afficher les planètes.

┌─────────────────┐↓ adresses, noms, nbPlanetes

│ affichePlanetes │

└─────────────────┘

*Précondition* : adresses et noms sont garnis avec nbPlanetes (≥ 0) valeurs.

*Postcondition* : les adresses et les noms des planètes sont affichées à l’écran comme dans les exemples suivants :

27 07 15 13 12 30 Abydos

24 04 20 09 15 29 Argos

09 02 23 15 37 20 Chulak

11 35 22 17 06 26 Cimmeria

28 26 05 36 11 29 Terre

Faites appel à cette fonction dans le programme principal pour afficher les nbPlanetes planètes obtenues.

## Étape 5 : nombre d’occurrences de chaque symbole

Écrivez une fonction qui détermine le nombre d’occurrences de chaque symbole.

┌─────────────────────┐↓ adresses, nbPlanetes

│ histogrammeSymboles │

└─────────────────────┘↓ histogramme

*Précondition* : adresses est garni avec nbPlanetes (≥ 0) valeurs.

*Postcondition* : histogramme contient nbCodes compteurs, correspondant chacun à un symbole et indiquant le nombre d’occurrences du symbole dans le tableau adresses.

## Étape 6 : histogramme des symboles

Dans la fonction principale, déclarez un tableau nbOccurrencesSymboles permettant de mémoriser le nombre d’occurrences de chacun des symboles.

Après l’affichage des planètes (étape 4), affichez, pour chaque symbole utilisé au moins une fois, son nombre d’occurrences.

Pour ne pas encombrer la fonction principale, placez cette partie du code dans une fonction annexe.

## Étape 7 : conversion de format

L’objectif est de remplir les tableaux adresses et noms sur bases de données reçues dans un format différent où les noms et les adresses sont représentés par des chaînes de caractères rassemblées dans un seul tableau.

Les données sont organisées comme suit (le code intégral se trouve en fin de document) :

const int LG\_DATA = 75;

char data[][40] = {

"ABYDOS", "aGOfLd",

"Abythres", "GFIJKL",

"Abythres2", "GBLIEF",

"Ailurus Fulgens", "eNlBmc",  
 …  
};

Dans ce format, chaque planète est décrite d’abord par son nom, puis par une seconde chaîne de caractères correspondant à son adresse où chaque symbole est représenté par une lettre plutôt qu’un nombre.

Les majuscules de A à Z correspondent aux symboles 1 à 26, les minuscules aux symboles suivants (jusqu’à 39 symboles maximum).

Par exemple, l’adresse d’Abydos, « aGOfLd » donne 27 7 15 32 12 30.

Écrivez une fonction qui se charge de garnir un tableau d’adresses et un tableau de noms à partir de ces données.

Commencez par recopier le code qui se situe à la fin de ce document au début de votre fonction.

Tenez compte des remarques suivantes.

* En plus de garnir les deux tableaux passés en arguments, la fonction renvoie un entier qui correspond au nombre de planètes encodées.
* Les noms des planètes doivent être normalisés (une majuscule en début et le reste en minuscules, les autres caractères restent inchangés).
* Organisez votre code de manière modulaire : créez des fonctions auxiliaires quand c’est pertinent et faites en sorte que, si on décide d’utiliser d’autres données, il suffise de remplacer les lignes copiés/collées depuis la fin du document sans toucher au reste du code.

N’hésitez pas à mettre en commentaires la boucle d’obtention de votre fonction principale (le do…while) pour la remplacer par un appel à cette nouvelle fonction.

## Étape 8 : planètes utilisant un symbole donné

L’objectif est d’écrire une fonction qui retrouve toutes les planètes utilisant un symbole donné. Elle doit renvoyer un tableau qui contient les indices des planètes en question, ainsi que la taille de ce tableau.

La signature du module est la suivante.

┌────────────────────┐↓ adresses, nbPlanetes, symbole

│ occurrencesSymbole │

└────────────────────┘↓ indicesOccurrences, nbOccurrences

En C, cela peut se faire en passant le tableau par adresse, comme d’habitude, et en renvoyant le nombre de cellules garnies… Le prototype de la fonction est le suivant :

int occurrencesSymbole (... , int indicesOccurrences[]);

↓ ↓

Nombre de cellules garnies Tableau des indices des planètes

Écrivez cette fonction et faites-y appel dans la fonction principale de façon à pouvoir afficher les noms de planètes contenant le symbole obtenu de l’utilisateur au clavier.

Voici un exemple :

Symbole a rechercher : 29

Le symbole 29 apparait 9 fois.

Planete #003 : 31 14 38 02 39 29 Ailurus fulgens

Planete #009 : 29 03 06 09 12 16 Castiana

Planete #015 : 32 09 21 30 18 29 Cyathea

Planete #019 : 28 26 05 36 11 29 Earth

Planete #031 : 02 04 23 30 29 37 Jon's staging gate

Planete #033 : 29 08 18 22 04 25 Juna

Planete #054 : 35 03 31 29 05 17 Praclarush taonas

Planete #060 : 29 18 19 20 21 22 Sahal

Planete #067 : 39 19 17 05 29 21 The.continuum

Pensez à faire une fonction permettant d’afficher les noms et adresses des planètes dont les indices sont récoltés dans le tableau. Elle est réutilisée ci-dessous !

## Étape 9 : recherche dans les noms

L’objectif est d’écrire une fonction qui permet de citer toutes les planètes dont le nom contient une chaîne de caractères donnée (à la manière d’une recherche de mot clé).

Le résultat de la fonction est exprimé de la même façon qu’à l’étape précédente.

Une fois la fonction écrite, ajoutez à la fonction principale des instructions permettant de

* demander une chaine de caractères à rechercher,
* appeler la fonction avec les paramètres adéquats, et
* afficher la liste des résultats ou « Aucun » si aucune planète ne correspond au critère de recherche.

Voici un exemple :

Texte a rechercher : do

Planete #000 : 27 07 15 32 12 30 Abydos

Planete #020 : 28 38 35 09 15 03 Edora

Planete #028 : 33 09 31 05 23 03 Hundoozelfra

## Étape 10 : recherche dans les adresses

L’objectif est d’écrire une dernière fonction de recherche qui permet de citer les planètes dont l’adresse reprend chacun des symboles d’une liste donnée.

Le critère de recherche est exprimé à travers deux arguments : un tableau de symboles et le nombre de symboles qu’il contient.

Le résultat, quant à lui, est formalisé de la même façon qu’à l’étape précédent.

Voici un exemple :

Symboles recherches : 15 35

Planete #020 : 28 38 35 09 15 03 Edora

Planete #042 : 38 28 15 35 03 19 Nid off-world base

Planete #045 : 38 09 28 15 35 03 P34-353j

Planete #048 : 03 28 09 35 24 15 P3x-562

Planete #056 : 20 02 35 08 26 15 Px1-767

## Code de l’étape 7

const int LG\_DATA = 75;

char data[][40] = {

"ABYDOS", "aGOfLd",

"Abythres", "GFIJKL",

"Abythres2", "GBLIEF",

"Ailurus Fulgens", "eNlBmc",

"APOPHIS'S BASE", "TRKlJf",

"Benzaar Primary", "dHSKeB",

"Boanet", "DdRHQa",

"Cassidy", "CdLebR",

"Cassiopeia", "UNfECZ",

"CASTIANA", "cCFILP",

"Centauri Prime", "LSDNYa",

"CHULAK", "IBWOkT",

"CLAVA THESSARA INFINITAS", "NUPKdG",

"CLAVA THESSARA INFINITAS2", "ZTifWD",

"CoRoT 7b", "CORGTX",

"Cyathea", "fIUdRc",

"DESTINY", "FQUeiXEN",

"DESTROYERS", "CfPHJL",

"Dochia", "DRWkFQ",

"EARTH", "bZEjKc",

"EDORA", "bliIOC",

"Estros", "GHTglUB",

"EURONDA", "daIGRP",

"eXos", "dgaKLl",

"FINAL DESTINATION", "IRaOUj",

"Fras Alpha Gate", "IVCNkF",

"Gaia designated P3X-774", "ISZQVE",

"Gallifrey", "EKakZb",

"Hundoozelfra", "gIeEWC",

"Jon Moyes", "LRZGhK",

"Jon's Gate #2", "DEFGVU",

"Jon's Staging Gate", "BDWdck",

"Jon's Test Gate", "mEFGHI",

"JUNA", "cHRVDY",

"KALLANA", "FPHCZY",

"KHEB", "ZiFHWN",

"Kristian Tysse", "GETafU",

"K'TAU", "RBdLZg",

"LANTIA (ATLANTIS)", "SUBPOHT",

"MARTIN LLOYD'S HOMEWORLD", "XLfGKh",

"Nab-Yloo", "JWYGFQ",

"New Horizon", "IFPLCH",

"NID OFF-WORLD BASE", "lbOiCS",

"OTHALA", "KaWPgCI",

"P2X-555", "bHPgMe",

"P34-353J", "lIbOiC",

"P3W-451 (black hole)", "SHDkZP",

"P3X-118", "IZhkQU",

"P3X-562", "CbIiXO",

"P9C-372", "YHRhDV",

"Patitan", "LBFHZU",

"PB5-926", "LjWRGa",

"PBH-403", "LJaDBX",

"Phoenix Planet (Phoenix Site)", "HeZgFL",

"PRACLARUSH TAONAS", "iCecEQ",

"Proxima Centauri B", "BQVadm",

"PX1-767", "TBiHZO",

"PX1-482 JonathanS Gate", "jLgCBY",

"Roverius", "LPfbQm",

"Rubito Prime", "WJIBUZ",

"SAHAL", "cRSTUV",

"SANGREAL PLANET", "dShIgR",

"Sepa's Sanctum", "LEIWHm",

"Serenity", "LCTJSV",

"TARTARUS", "gbWZPe",

"Terra Nova", "eEHTYC",

"Tesseract", "ULSCWg",

"The.Continuum", "mSQEcU",

"TOLLAN", "FgakKR",

"TOLLANA", "DhHVRY",

"Tython", "GmPTfZ",

"VAGON BREI", "CHBLSd",

"Vasut's gate", "eWBdGP",

"Vasut's second gate ", "fLFCRB",

"Velma", "KJEFCV"

};