Les types utilisateurs : exercices résolus en C

Objectifs

- Connaître les types utilisateurs : tableaux, enregistrements et type énumérés ;
- Connaître les algorithmes fondamentaux sur les tableaux ;
- Continuer à appliquer les principes de la semaine 1.

Exercice 1 : Occurrences des chiffres d'un entier

Écrire un programme qui compte le nombre d'occurrences des 10 chiffres dans un entier naturel donné.

Par exemple, l'entier 4214 a une occurrence du chiffre 1, une de 2 et deux de 4.

Exercice 2 : Modéliser un robot de type 1

L'objectif de cet exercice est de modéliser un robot de type 1. Un tel robot se déplace dans un environnement qui peut être modélisé par un quadrillage dont chaque case correspond à une position possible du robot.

Un robot est donc caractérisé par sa position (son abscisse, x, et son ordonnée, y) et sa direction (nord, sud, est ou ouest). La figure 1 décrit un robot à la position $x = 4$ et $y = 2$, sa direction est « ouest ».

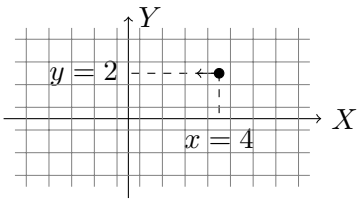

FIGURE 1 – Le robot dans un environnement illimité

Les robots de type 1 ont des possibilités réduites qui se limitent à pivoter de 90^o vers la droite et à avancer d'une case suivant sa direction courante.

Pour simplifier, on considère que le robot évolue dans un environnement infini et sans obstacle.

2.1 Définir les types nécessaires pour modéliser un robot de type 1.

2.2 L'environnement dans lequel évolue le robot est en fait fini et comporte des obstacles. On suppose qu'un obstacle occupe une case du quadrillage. La question est donc de savoir si la case est libre ou contient un obstacle.

Définir un type pour modéliser l'environnement.

2.3 Écrire un programme qui fait avancer un robot toujours tout droit jusqu'à ce qu'il rencontre un obstacle ou qu'il arrive à la limite de son environnement. On supposera que le robot est initialement sur une position valide avec pour direction « est ».

Exercice 3 : Lecture d'un tableau
Écrire un programme qui initialise un tableau d'entiers en lisant des valeurs au clavier. On considèrera que les valeurs sont saisies les unes après les autres et que la saisie s'arrête sur une valeur négative. La valeur négative ne doit pas être conservée dans le tableau.

Exercice 4 : Tri par insertion séquentielle
Soit $A[1..N]$ un vecteur de N entiers relatifs quelconques, l'objectif est de trier le vecteur A. Le vecteur A est trié si $A[1] \leq A[2] \leq \cdots \leq A[N]$.

Le tri utilisé est le tri par insertion séquentielle. C'est un tri en $(N-1)$ étapes. L'étape i consiste à placer le $(i+1)^{\text{e}}$ élément du vecteur à sa place dans le sous-vecteur $A(1..i+1)$ sachant que $A(1..i)$ a été trié par les étapes précédentes. Dans le cas du tri par insertion séquentielle, la recherche de la position d'insertion, se fait séquentiellement en comparant successivement l'élément à insérer aux i premiers éléments du vecteur.
Exemple : Voici les différentes valeurs du vecteur 8 2 9 5 1 7 après chaque étape (la partie encadrée correspond à la partie du vecteur déjà traitée et donc triée) :

vecteur initial	:	$\boxed{8}\ \underline{2}\ 9\ 5\ 1\ 7$
après l'étape 1	:	$\boxed{2\ 8}\ \underline{9}\ 5\ 1\ 7$
après l'étape 2	:	$\boxed{2\ 8\ 9}\ \underline{5}\ 1\ 7$
après l'étape 3	:	$\boxed{2\ 5\ 8\ 9}\ \underline{1}\ 7$
après l'étape 4	:	$\boxed{1\ 2\ 5\ 8\ 9}\ \underline{7}$
après l'étape 5	:	$\boxed{1\ 2\ 5\ 7\ 8\ 9}$

4.1 Écrire un programme qui lit une série de N ($N > 0$, lu au clavier) nombres relatifs, les range dans un vecteur A, les classe par ordre croissant en utilisant le tri par insertion séquentielle et, enfin, affiche la série ordonnée.

4.2 En conservant le principe du tri par insertion, expliquer comment améliorer l'efficacité de cet algorithme.