Constructeurs de types

Objectifs

— Comprendre et savoir définir les types utilisateurs

— Manipuler les tableaux

— Écrire une méthode de tri simple... mais pas très efficace.

— Raffiner... Toujours !

Exercice 1 : 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 ».

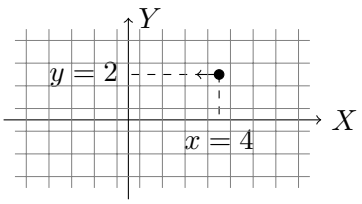

FIGURE 1 – Le robot dans un environnement illimité

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 sans obstacle.

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

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. Il s'agit donc de savoir si la case est libre ou contient un obstacle. Définir un type pour modéliser l'environnement.

3. Écrire un sous-programme qui, étant donnés un robot et son environnement, fait avancer ce robot toujours tout droit jusqu'à ce qu'il rencontre un obstacle ou qu'il arrive à la limite de son environnement. On supposera le robot initialement sur une position valide en direction de l'est.

Exercice 2 : Afficher un tableau
Écrire un sous-programme qui affiche à l'écran un tableau d'entiers. Le tableau sera délimité par des crochets et les éléments seront séparés par une virgule. Voici quelques exemples d'affichage :

```
[]
[1, 2, 3]
[4]
[10, 2, 4, 7]
```

Exercice 3 : Tri par insertion séquentielle
L'objectif est de trier le vecteur$A$ de N entiers relatifs quelconques. Le vecteur A est trié si $A[i] \leq A[i+1]$.

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 $i+1$ premiers éléments sachant que sous-vecteur des i premiers éléments est 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	:	[8] 2 9 5 1 7
après l'étape 1	:	[2 8] 9 5 1 7
après l'étape 2	:	[2 8 9] 5 1 7
après l'étape 3	:	[2 5 8 9] 1 7
après l'étape 4	:	[1 2 5 8 9] 7
après l'étape 5	:	[1 2 5 7 8 9]

1. Écrire un sous-programme pour trier dans l'ordre croissant les éléments d'un vecteur. On utilisera le principe du tri par insertion séquentielle.
2. En conservant le principe du tri par insertion, expliquer comment améliorer l'efficacité de cet algorithme.

Exercice 4 : Saisir un tableau
Écrire un programme qui initialise un tableau d'entiers en lisant des valeurs au clavier. On considèrera trois modes de lecture :

1. lecture de la taille effective du tableau, puis des valeurs ;
2. lecture des valeurs les unes après les autres et arrêt sur une valeur particulière (une valeur négative par exemple) ;
3. lecture de couples (indice, valeur) avec arrêt lorsque l'indice donné est -1.

Quels sont les avantages et les inconvénients de ces différentes approches ?
Remarque : on veillera à ce que la capacité du tableau ne soit pas dépassée.