TP 3 : Les ensembles

Objectifs du TP Ce TP a pour but

1. manipuler des tableaux

1 Introduction

Le but du travail de cette semaine est d'implanter des fonctions et des procédures permettant de manipuler des ensembles. On souhaite que les éléments des ensembles soit des valeurs choisie dans un type ordinal quelconque qu'on a appellé ELEMENT.

On vous fournit le fichier `squelette_ensemble.pas` qu'il faudra compléter.

On suppose qu'on dispose de deux procédures nommées respectivement lire et ecrire qui permettent respectivement d'entrer et d'afficher des éléments du type ELEMENT

Pour fixer les idées, on a procédé aux déclarations suivantes :

```
type MOIS = (janvier, fevrier, mars, avril, mai,
             juin, juillet, aout, septembre,
             octobre, novembre, decembre);
const NOM_DU_MOIS : array [MOIS] of string
                  = ('janvier', 'fevrier', 'mars', 'avril', 'mai',
                     'juin', 'juillet', 'aout', 'septembre',
                     'octobre','novembre','decembre');
type ELEMENT = MOIS;

// contrainte d'utilisation:
// l'utilisateur DOIT taper le mois en minuscule
// sans ajouter d'espace ni avant ni après,
// et ne pas faire d'erreur...
procedure lire(out x : MOIS);
var s :string;
    i :MOIS;
begin
  readln(s);
  for i := low(MOIS) to HIGH(MOIS) do
  // c'est maladroit de faire une boucle pour
  // pour faire la recherche dans un tableau
  // mais ici comme la longueur du tableau
  // est assez petite...
  begin
     if NOM_DU_MOIS[i]=s then x:=i;
  end {for};
end {lire};

procedure ecrire(const x : MOIS);
begin
  write(NOM_DU_MOIS[x]);
end {ecrire}
```

Pour représenter un ensemble E d'éléments, on décide d'utiliser un tableau t de booléens et dont les indices sont des éléments. On se fixe la règle suivante, $x \in E \Longleftrightarrow t[x]$

Par exemple pour représenter l'ensemble : $\{janvier, mars, juin\}$ on utilise le tableau t suivant :

indice	J	F	M	A	M	J	J	A	S	O	N	D
élément	V	F	V	F	F	V	F	F	F	F	F	F

REMARQUE : Conservez précieusement le travail suivant car il resservira plus tard.

2 Travail à faire

2.1 le type ENSEMBLE

Q 1 . Déclarer le type ENSEMBLE
Q 2 . Dans le programme principal, donner les instructions qui permettent d'initialiser la variable t avec la valeur du tableau donné plus haut.
Q 3 . Réaliser une procédure afficher qui écrit les éléments qui appartiennent à l'ensemble, séparés par des espaces.

Par exemple :

```
afficher(t);
```

donnera

```
janvier mars juin
```

2.2 Opération sur les ensembles

Q 4 . Réaliser les fonctions suivantes :

```
// fonction sans parametre dont le résultat est l'ensemble vide
function partie_vide : ENSEMBLE;

// fonction sans paramètre dont le résultat est l'ensemble de tous
//   les éléments du type ELEMENT.
function partie_pleine : ENSEMBLE;

// fonction dont le résultat est l'ensemble des éléments qui
//   appartiennent à a ou à b (ou les deux)
function union(const a,b : ENSEMBLE): ENSEMBLE;

// fonction dont le résultat est l'ensemble des éléments qui
//   appartiennent à a et à b
function intersection(const a,b:ENSEMBLE): ENSEMBLE;

// fonction dont le résultat est l'ensemble des éléments de
// l'ensemble a qui n'appartiennent pas à b
function difference(const a,b : ENSEMBLE): ENSEMBLE;

// fonction dont le résultat est le nombre d'éléments de
//  l'ensemble a (on ne pouvait pas l'appeler cardinal...)
function nombre(const a : ENSEMBLE):CARDINAL;

// fonction dont le résultat est l'ensemble des éléments
// de la partie_pleine qui n'appartiennent pas à a
function complementaire(const a : ENSEMBLE): ENSEMBLE;

// prédicat permettant de tester si x est un élément de a
function appartient(const x : ELEMENT; const a : ENSEMBLE):
BOOLEAN;

// fonction dont le résultat est un ensemble contenant
// uniquement l'élément x
function singleton(const x : ELEMENT) : ENSEMBLE;

// fonction dont le résultat est un ensemble contenant tous les
//   éléments de a sauf x  (il n'est pas nécessaire que x
//   soit un élément de a
function prive(const x : ELEMENT ; const a : ENSEMBLE):ENSEMBLE;

// fonction dont le résultat est un ensemble contenant tous
```

```
//  les éléments de a  et x (il n'est pas nécessaire que
//  x ne soit pas un élément de a)
function ajout(const x : ELEMENT ; const a : ENSEMBLE):ENSEMBLE;
```

Q 5 . Modifier la procédure afficher pour qu'elle écrive l'ensemble en extension, comme en mathématique, c'est à dire qui écrit la liste des éléments de l'ensemble séparés par des virgules et entouré par des accolades. (On représentera l'ensemble vide de la manière suivante : VIDE)

Q 6 . Réfléchir au moyen d'implanter une procédure lire qui permet à l'utilisateur d'entrer un ensemble de son choix.

Q 7 . Pour chacune des fonctions et procédures, réaliser des tests dans le programme principal.