

RÉVISIONS DES CONCEPTS

Je propose parfois deux versions possibles : une avec filtrage de la saisie utilisateur et une sans.

Remarque : tous les filtrages de données récupérées depuis le clavier sont faites ici avec répéter ... jusqu'à. On peut bien entendu faire la même chose sans avec les autres boucles non bornées.

Exercice 1 :

Question 1.1

```
/* Version Q1.1 SANS FILTRAGE */
CONST MAX = 30;

Programme exo1.1
  VAR k : Entier; /* k compteur */
      ca : Entier; /* ca pour Chiffre d'Affaires */
      np : Entier; /* Nombre Participants d'une session */
début
  ca ← 0;
  pour k de 1 à MAX pas 1 faire /* Faire 30 fois */
  | Ecrire("Combien de participants ? ") ;
  | Lire(np);
  | ca ← ca + np*5 ;
  fin pour
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

```
/* Version Q1.1 avec filtrage */
CONST MAX = 30;

Programme exo1.1
  VAR k : Entier; /* k compteur */
      ca : Entier; /* ca pour Chiffre d'Affaires */
      np : Entier; /* Nombre Participants d'une session */
début
  ca ← 0;
  pour k de 1 à MAX pas 1 faire /* Faire 30 fois */
  | répéter
  | | Ecrire("Combien de participants ? ") ;
  | | Lire(np);
  | | jusqu'à (np > 0) ;
  | ca ← ca + np*5 ;
  fin pour
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

Question 1.2

Passons du pour au TQ : il faut initialiser et incrémenter à la main.

```
/* Version SANS FILTRAGE */
CONST MAX = 30;

Programme exo1.2
  VAR k, ca, np : Entier;
début
  ca ← 0;
  k ← 0 ;                               /* INITIALISATION COMPTEUR */
  tant que (k < MAX) faire
    | Ecrire("Combien de participants ? " ;
    | Lire(np);
    | ca ← ca + np*5 ;
    | k ← k + 1;                         /* INCREMENTATION COMPTEUR */
  fin tant que
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

```
/* Version avec filtrage */
CONST MAX = 30;

Programme exo1.2
  VAR k, ca, np : Entier;
début
  ca ← 0;
  k ← 0 ;                               /* INITIALISATION COMPTEUR */
  tant que (k < MAX) faire
    | répéter
    | | Ecrire("Combien de participants ? " ;
    | | Lire(np);
    | jusqu'à (np > 0) ;
    | ca ← ca + np*5 ;
    | k ← k + 1;                         /* INCREMENTATION COMPTEUR */
  fin tant que
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

Question 1.3

Dans cette correction, je réalise la boucle MAX fois en faisant varier le compteur de **0 à (MAX-1)** pour montrer que cela revient à créer un compteur de **1 à MAX**.

```
/* SANS FILTRAGE */
CONST MAX = 30;

Programme exo1.3
  VAR k, ca, np : Entier;
      rep      : Caractere      /* 'c' ou 'p' uniquement */
début
  ca ← 0;
  pour k de 0 à (MAX - 1) pas 1 faire /* toujours 30 fois */
  |   Ecrire("Type de séance : p ou c : ");
  |   Lire(rep);
  |   Si (rep = 'c') alors
  |   |   Ecrire("Combien de participants ? " ;
  |   |   Lire(np);
  |   |   ca ← ca + np*5 ;
  |   Sinon
  |   |   ca ← ca + 30;
  |   fin Si
  fin pour
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

```
/* AVEC FILTRAGE */
CONST MAX = 30;

Programme exo1.3
  VAR k, ca, np : Entier;
      rep      : Caractere      /* 'c' ou 'p' uniquement */
début
  ca ← 0;
  pour k de 0 à (MAX - 1) pas 1 faire /* toujours 30 fois */
  |   répéter
  |   |   Ecrire("Type de séance : p ou c : ");
  |   |   Lire(rep);
  |   jusqu'à (rep = 'c' OU rep = 'p') ;
  |   Si (rep = 'c') alors
  |   |   répéter
  |   |   |   Ecrire("Combien de participants ? " ;
  |   |   |   Lire(np);
  |   |   jusqu'à (np > 0) ;
  |   |   ca ← ca + np*5 ;
  |   Sinon
  |   |   ca ← ca + 30;
  |   fin Si
  fin pour
  Ecrire("Chiffre d'affaires : ", ca);
fin
```

Question 1.4

Attention, cette question demande uniquement de créer le type !

```
CONST MAXSEANCES = 1560;  
TYPE TAB_COACH = tableau[MAXSEANCES] de Caractere;
```

Question 1.5

```
/* SANS FILTRAGE */  
CONST MAX = 30;  
CONST MAXSEANCES = 1560;  
TYPE TAB_COACH = tableau[MAXSEANCES] de Caractere;  
  
Programme exo1.5  
  VAR i      : Entier;  
      t      : Tab;  
      rep    : Caractere  
début  
  pour i de 0 à (MAX -1) pas 1 faire  
    Ecrire("Séance numéro ", i+1); /* pour le lecteur humain */  
    Ecrire("Type de séance : p ou c : ");  
    Lire(t[i]);  
  fin pour  
fin
```

```
/* AVEC FILTRAGE */  
CONST MAX = 30;  
CONST MAXSEANCES = 1560;  
TYPE TAB_COACH = tableau[MAXSEANCES] de Caractere;  
  
Programme exo1.5  
  VAR i      : Entier;  
      t      : Tab;  
      rep    : Caractere  
début  
  pour i de 0 à (MAX -1) pas 1 faire  
    Ecrire("Séance numéro ", i+1); /* pour le lecteur humain */  
    répéter  
      Ecrire("Type de séance : p ou c : ");  
      Lire(t[i]);  
    jusqu'à (t[i] = 'c' OU t[i] = 'p');  
  fin pour  
fin
```

Exercice 2 :

Question 2.1

```
Programme exo2.1
    VAR k, v : Entier;          /* k compteur, v valeur */
début
    pour k de 1 à 10 pas 1 faire
        v ← aléa_entier(1, 6) ;
        Ecrire(v) ;
    fin pour
fin
```

Question 2.2

```
Programme exo2.2
    VAR v      : Entier;  /* v pour valeur */
        rep   : Caractère ;
début
    rep ← 'o'; /* Valeur initiale permettant de commencer */
    Tant que (rep = 'o') faire
        v ← aléa_entier(1, 6) ;
        Ecrire(v) ;
        Ecrire("Continuer (o/n) : ");
        Lire(rep);
    fin Tant que
fin
```

Question 2.3

```
Programme exo2.3
    VAR nb, v : Entier;  /* v pour valeur */
début
    nb ← 0 ;
    Répéter
        v ← alea_entier(1, 6) ;
        Ecrire(v) ;
        nb ← nb + 1 ;
    Jusqu'à (v = 6);
    Ecrire("Nombre de lancés pour obtenir 6 : ", nb) ;
fin
```

Question 2.4

```
Programme exo2.4
    VAR nb, v, somme : Entier;
début

    nb ← 0 ;
    somme ← 0 ;

    Faire
        v ← alea_entier(1, 6) ;
        Ecrire(v) ;
        somme ← somme + v ;
        nb ← nb + 1 ;
    tant que (nb ≠ 10 ET v ≠ 6);
```

```
Ecrire("Nombre de lancés : ", nb) ;  
Si (v = 6) alors  
    Ecrire("Un 6 a été tiré.") ;  
fin si  
fin
```

REMARQUE 2.4 :

On s'arrête si on a fait **10 lancés OU** qu'un **six est sortie**.

Donc on continue tant qu'on a **pas 10 lancés ET** que le nombre **six n'est pas sorti**.

Exercice 3 :

Question 3.1

```
/* SANS FILTRAGE */
Programme Sobriete
  VAR nb : Entier;
      elect, gaz, eau : Réel;
debut
  /* Récupération des données */

  Ecrire("Nombre de personnes ? "); Lire(nb);
  Ecrire("Conso électricité ? ") ; Lire(elec);
  Ecrire("Conso gaz ? ") ; Lire(gaz);
  Ecrire("Conso eau ? ") ; Lire(eau);

  /* Gestion et affichage */

  Si (nb ≥ 4) alors
    | Si ( ((gaz+elec) < 15000) ET (eau < 150) ) alors
    | | Ecrire("Bravo");
    | Sinon
    | | Ecrire("Vous pouvez progresser.");
    | fin si
  Sinon
    | Si ( ((gaz+elec) < 10000) ET (eau < 100) ) alors
    | | Ecrire("Bravo");
    | Sinon
    | | Ecrire("Vous pouvez progresser.");
    | fin Si
  fin Si
fin
```

```

/* AVEC FILTRAGE */
Programme Sobriete
  VAR nb : Entier;
      elect, gaz, eau : Réel;
debut
  /* Récupération des données */

  répéter
  | Ecrire("Nombre de personnes ? "); Lire(nb);
  jusqu'à (nb > 0);

  répéter
  | Ecrire("Conso électricité ? ") ; Lire(elec);
  jusqu'à (elec > 0);

  répéter
  | Ecrire("Conso gaz ? ") ; Lire(gaz);
  jusqu'à (gaz > 0);

  répéter
  | Ecrire("Conso eau ? ") ; Lire(eau);
  jusqu'à (eau > 0);

  /* Gestion et affichage */

  Si (nb ≥ 4) alors
  | Si ( ((gaz+elec) < 15000) ET (eau < 150) ) alors
  | | Ecrire("Bravo");
  | Sinon
  | | Ecrire("Vous pouvez progresser.");
  | fin si
  Sinon
  | Si ( ((gaz+elec) < 10000) ET (eau < 100) ) alors
  | | Ecrire("Bravo");
  | Sinon
  | | Ecrire("Vous pouvez progresser.");
  | fin Si
  fin Si
fin

```

Question 3.2

```

CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

```

Question 3.3

```
/* SANS FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme exo3.3
  VAR i, N : Entier ;
      te   : TAB_CONSO ;
début
  Ecrire("Combien de mois à stocker ? ");
  Lire(N);

  pour i de 0 à (N-1) pas 1 faire
  | Ecrire("Fournir la conso élect du mois ", N+1) ;
  | Lire(te[i]);
  fin pour
fin
```

```
/* AVEC FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme exo3.3
  VAR i, N : Entier ;
      te   : TAB_CONSO ;
début
  répéter
  | Ecrire("Combien de mois à stocker ? ");
  | Lire(N);
  jusqu'à (N > 0 ET N ≤ MAX);

  pour i de 0 à (N-1) pas 1 faire
  | Ecrire("Fournir la conso élect du mois ", N+1) ;
  | Lire(te[i]);
  fin pour
fin
```

Le décalage de 1 sur le numéro du mois évite à l'utilisateur d'être perturbé par le fait qu'il existe un mois 0 : on stocke sur les indices 0 à (MAX-1) mais on affichera les mois 1 à MAX à l'écran.

Question 3.4

```
/* SANS FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme exo3.4
  VAR i, N : Entier ;
      te   : TAB_CONSO ;
début
  Ecrire("Combien de mois à stocker ? ");
  Lire(N);

  i ← 0 ;                               /* INITIALISATION */
  tant que (i < N) faire
  | Ecrire("Fournir la conso élect du mois ", N+1) ;
  | Lire(te[i]);
  | i ← i + 1 ;      /* INCREMENTATION */
  fin pour
fin
```

```
/* AVEC FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme exo3.4
  VAR i, N : Entier ;
      te   : TAB_CONSO ;
début
  répéter
  | Ecrire("Combien de mois à stocker ? ");
  | Lire(N);
  jusqu'à (N > 0 ET N ≤ MAX);

  i ← 0 ;                               /* INITIALISATION */
  tant que (i < N) faire
  | Ecrire("Fournir la conso élect du mois ", N+1) ;
  | Lire(te[i]);
  | i ← i + 1 ;      /* INCREMENTATION */
  fin pour
fin
```

Question 3.5 SOMME

Compléter donc il s'agit de la suite du programme précédent : ceci est à placer entre le fin pour et le fin en fin de question 4. Néanmoins, il faudrait déclarer la nouvelle variable M dans l'en-tête.

```
/* SANS FILTRAGE */
CONST MAX = 36 ;
TYPE TAB_CONSO = Tableau[MAX] de Réel ;

Programme exo3.5
  VAR i, N, M : Entier ;
      te      : TAB_CONSO ;
début
  ...
  fin pour

  Ecrire("Somme sur combien de mois ? ") ;
  Lire(M);

  somme ← 0;          /* IMPORTANT */
  pour i de 0 à (M-1) pas 1 faire
  | somme ← somme + te[i] ;
  fin pour
  Ecrire("Conso sur les ", M, "premiers mois : ", somme);
fin
```

```
/* AVEC FILTRAGE */
CONST MAX = 36 ;
TYPE TAB_CONSO = Tableau[MAX] de Réel ;

Programme exo3.5
  VAR i, N, M : Entier ;
      te      : TAB_CONSO ;
début
  ...
  fin pour

  répéter
  | Ecrire("Somme sur combien de mois ? ") ;
  | Lire(M);
  jusqu'à (M > 0 ET M ≤ N);

  somme ← 0;          /* IMPORTANT */
  pour i de 0 à (M-1) pas 1 faire
  | somme ← somme + te[i] ;
  fin pour
  Ecrire("Conso sur les ", M, "premiers mois : ", somme);
fin
```

Question 3.6 ENUMERATION

On signale dans l'énoncé : une année complète. On veut donc stocker 12 valeurs.

```
/* SANS FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme NbMoisSobres
  VAR i, sobre : Entier ; /* le compteur se nomme sobre */
      te      : TAB_CONSO ;
      seuil   : Réel ; /* Réel pour être compatible avec te */
début
  sobre ← 0 ;

  Ecrire("Quel seuil désirez-vous ? ");
  Lire(seuil);

  pour i de 0 à 11 pas 1 faire
    | Ecrire("Fournir une conso de gaz : ") ;
    | Lire(te[i]);
    | si (te[i] < seuil) alors
    |   | sobre ← sobre + 1;
    |   fin si
  fin pour

  Ecrire("Nombre de mois sobres en gaz : ", nb);
fin
```

```
/* AVEC FILTRAGE */
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme NbMoisSobres
  VAR i, sobre : Entier ; /* le compteur se nomme sobre */
      te      : TAB_CONSO ;
      seuil   : Réel ; /* Réel pour être compatible avec te */
début
  sobre ← 0 ;

  répéter
    | Ecrire("Quel seuil désirez-vous ? ");
    | Lire(seuil);
  jusqu'à (seuil > 0.0) ;

  pour i de 0 à 11 pas 1 faire
    | répéter
    |   | Ecrire("Fournir une conso de gaz : ") ;
    |   | Lire(te[i]);
    |   jusqu'à (te[i] > 0.0);
    |   si (te[i] < seuil) alors
    |     | sobre ← sobre + 1;
    |     fin si
  fin pour
  Ecrire("Nombre de mois sobres en gaz : ", nb);
fin
```

Question 3.7 POSITION MINIMUM

Dans cette question, on ne filtrera pas les valeurs avec un répéter...jusqu'à, la compétence a été évaluée de nombreuses fois sur les questions précédentes.

La difficulté ici est de ne pas confondre `imin` qui est la position où se trouve la plus petite valeur et `t[imin]` qui est la valeur minimale.

```
CONST MAX = 36;
TYPE TAB_CONSO = Tableau[MAX] de Réel;

Programme MoisLePlusSobre
  VAR i, N, imin : Entier ;
      te          : TAB_CONSO ;
début
  Ecrire("Combien de mois à stocker ? ") ;
  Lire(N);

  /* Première lecture */
  Ecrire("Fournir une conso d'eau : ") ;
  Lire(te[0]);
  imin ← 0;          /* min en 0 ? */

  /* Les autres valeurs */
  pour i de 1 à (N-1) pas 1 faire
  |   Ecrire("Fournir une conso d'eau : ") ;
  |   Lire(te[i]);
  |   si (te[i] < te[imin]) alors
  | |   imin ← i;
  |   fin si
  fin pour

  Ecrire("Conso la plus faible :", te[imin]);
  Ecrire("Mois où cette conso a été mesurée : ", imin) ;
fin
```

On remarquera que la dernière ligne devrait plutôt être ceci pour que l'humain qui la lit ne soit pas perturbé par le décalage de 1 par rapport à ce qu'il a rentré.

```
Ecrire("Mois où cette conso a été mesurée : ", imin+1) ;
```

Exercice 4 :

Question 4.1

Programme 4.1

```
VAR n, m : Entier;
début
  répéter
    | Ecrire("Quel entier voulez-vous ? ");
    | Lire(n);
  jusqu'à (n > 0);

  Pour m de 2 à (n-1) pas 1 faire
    | si (n MOD m = 0) alors
    | | Ecrire(m, " est un diviseur de ", n) ;
    | fin si
  fin pour
fin
```

Question 4.2 : Affichage des diviseurs

Programme 4.2

```
VAR n, m, m2 : Entier;
    m_max : Réel;
début
  répéter
    Ecrire("Quel entier voulez-vous ? ");
    Lire(n);
  jusqu'à (n > 0);

  m_max ← racine(n); /* recherche de la dernière valeur à tester */
  m ← 2 ; /* INITIALISATION à 2 */
  Tant que (m ≤ m_max) faire
    | si (n MOD m = 0) alors
    | | Ecrire(m, " est un diviseur de ", n) ;
    | | m2 ← n DIV m ;
    | | si (m2 ≠ m) alors
    | | | Ecrire(m2, " est un diviseur de ", n) ;
    | | fin si
    | fin si
    | m ← m + 1; /* INCREMENTATION */
  fin tant que
fin
```

Explication : pour optimiser un peu, on peut se rendre compte que lorsqu'on découvre un diviseur inférieur à $\sqrt{n}$, on découvre également nécessairement un diviseur supérieur à $\sqrt{n}$.

Ainsi avec 100, lorsqu'on découvre que :

- 2 est un diviseur, on découvre également que 50 est un diviseur.
- 3 non.
- 4 est un diviseur, on découvre également que 25 est un diviseur.
- 5 est un diviseur, on découvre également que 20 est un diviseur.
- 6-7-8-9 non.
- 10 est un diviseur, on découvre également que 10 est un diviseur.
- Inutile de poursuivre au delà de 10 : ils ont déjà été découverts.

On suppose disposer d'une fonction `racine(n)` et que la comparaison entre un entier et un réel est possible.
Pour vers Tant Que : il faut rajouter l'INITIALISATION et l'INCREMENTATION.

Question 4.3 : Prédicat d'un nombre premier

Il faut changer l'algorithme : il faut tenter de diviser le nombre par les nombres inférieurs jusqu'à sa racine et s'arrêter si on trouve un diviseur.

Si on sort sans avoir trouvé de diviseur, c'est qu'il est premier.

Le booléen nommée `est_premier` est placé initialement à Vrai : on suppose que notre nombre est premier. Ensuite, si on détecte un nombre qui divise `n`, on le placera à Faux. Ce booléen nous permettra de sortir de la boucle dès qu'on détecte que le nombre n'est pas premier.

```
Programme 4.3
VAR n, m          : Entier;
    m_max         : Réel;
    est_premier    : Booléen;
début
    est_premier ← Vrai; /* On le suppose puis on va tester */

    répéter
    | Ecrire("Quel entier supérieur à 1 voulez-vous ? ");
    | Lire(n);
    jusqu'à (n > 1);

    m_max ← racine(n);

    m ← 2 ;          /* INITIALISATION */
    Tant que (m ≤ m_max ET est_premier) faire
    | si (n MOD m = 0) alors
    | | est_premier ← Faux ;
    | fin si
    | m ← m + 1;      /* INCREMENTATION */
    fin tant que

    Si (est_premier) alors
    | Ecrire(n, " est un nombre premier.");
    Sinon
    | Ecrire(n, " n'est pas premier.");
    Fin si
fin
```

On sort de la boucle Tant que soit parce qu'on détecte que le nombre n'est pas premier, soit parce qu'on a atteint la dernière valeur sans détecter de diviseur. Le contenu du booléen `est_premier` nous permet donc de conclure.

REMARQUE : si (condition) active le bloc du alors si la condition est Vrai. Puisque notre variable `est_premier` est justement un booléen qui vaut Vrai ou Faux, on écrira donc :

```
Si (est_premier) alors
```

plutôt que ceci (même si cela fonctionne également) :

```
Si (est_premier = Vrai) alors
```

En effet, (est_premier = Vrai) est justement Vrai uniquement si est_premier est Vrai.

Question 4.4 : Crible d'Ératosthène

```
CONST MAX = 101;    /* 101 pour avoir des indices de 0 à 100 */
TYPE T = Tableau[MAX] d'Entier ;

Programme Crible_Ératosthène
  VAR i, multiple : Entier ;
      t : T ;
début
  /* Initialisation du tableau t[i] = i */
  pour i de 0 à (MAX-1) faire
    | t[i] ← i;
  fin pour

  /* 1 n'est pas premier : on y place 0 */
  t[1] ← 0;

  /* pour chaque entier supérieur à 1 */
  pour i de 2 à (MAX-1) faire
    | si (t[i] ≠ 0) alors /* si i encore signalé comme premier */
    | | pour multiple de (2 × i) à MAX pas i faire
    | | | t[multiple] ← 0 ;
    | | fin pour
    | fin si
  fin pour

fin
```

L'idée est donc de "supprimer" progressivement les entiers non premiers à savoir ceux qui sont multiples d'un autre.

Ainsi, on part de 2 puis on supprime tous les multiples de 2 de notre recherche.

Ensuite, on passe à 3 (qui n'est pas multiple de 2) donc toujours noté comme premier. On supprime alors les multiples de 3 (6-9-12...)

On passe à 4 mais sa case contient 0. Case suivante.

On passe à la case 5 qui contient encore 5, il est donc détecté comme premier. On supprime ses multiples (10-15-20....)

Ci-dessous, le même programme mais avec des TANT QUE à la place des POUR (non demandé sur la fiche) :

```
CONST MAX = 101;
TYPE T = Tableau[MAX] d'Entier ;

Programme Crible_Ératosthène
  VAR i, multiple : Entier ;
      t : T ;
début
  /* Initialisation du tableau t[i] = i */
  pour i de 0 à (MAX-1) faire
```

```

| t[i] ← i;
fin pour

/* 1 n'est pas premier : on y place 0 */
t[1] ← 0;

/* pour chaque entier supérieur à 1 */
i ← 2 ;
tant que (i < MAX) faire
|
| si (t[i] ≠ 0) alors /* si i encore signalé comme premier */
| |
| | multiple ← i × 2 ;
| | tant que (multiple < MAX) faire
| | | t[multiple] ← 0 ;
| | | multiple ← multiple + i ;
| | fin tant que
| fin si
| i ← i + 1 ; /* on passe au suivant */
fin tant que

fin

```