Algorithmique et programmation : les bases (F)
Corrigé

Résumé

Ce document décrit l'écriture dans le langage F des éléments vus en algorithmique.

Table des matières

1 Pourquoi définir notre langage algorithmique ? 3

2 Structure d'un algorithme 3
2.1 Exemple d'algorithme : calculer le périmètre d'un cercle . . . 3
2.2 Structure de l'algorithme . . . 3
2.3 Identificateurs . . . 4
2.4 Commentaires . . . 4

3 Variables 4
3.1 Qu'est ce qu'une variable ? . . . 4
3.2 Définition d'une variable . . . 4

4 Types fondamentaux 4
4.1 Les entiers . . . 4
4.2 Les réels . . . 5
4.3 Les booléens . . . 5
4.4 Les caractères . . . 5
4.5 Les chaînes de caractères . . . 5

5 Constantes 5

6 Expressions 6

7 Instructions d'entrée/sorties 6
7.1 Opération d'entrée . . . 6
7.2 Opération de sortie . . . 6

8 Affectation 7

9 Structures de contrôle **8**
9.1 Enchaînement séquentiel 8
9.2 Instructions conditionnelles 9
9.2.1 Conditionnelle **`Si ... Alors ... FinSi`** 9
9.2.2 Conditionnelle **`Si ... Alors ... Sinon ... FinSi`** 10
9.2.3 La clause **`SinonSi`** 12
9.2.4 Conditionnelle **`Selon`** 13
9.3 Instructions de répétitions 14
9.3.1 Répétition **`TantQue`** 14
9.3.2 Répétition Répéter ... JusquÀ 17
9.3.3 Répétition **`Pour`** 19
9.3.4 Quelle répétition choisir ? 20

Liste des exercices

Exercice 1 : Cube d'un réel *6*
Exercice 2 : Permuter deux caractères *7*
Exercice 3 : Cube d'un réel (avec une variable) *7*
Exercice 4 : Une valeur entière est-elle paire ? *9*
Exercice 5 : Maximum de deux valeurs réelles *11*
Exercice 6 : Signe d'un entier *12*
Exercice 7 : Réponse *13*
Exercice 8 : Somme des premiers entiers (TantQue) *14*
Exercice 9 : Saisie contrôlée d'un numéro de mois *16*
Exercice 10 : Plusieurs sommes des n premiers entiers *17*
Exercice 11 : Saisie contrôlée d'un numéro de mois *18*
Exercice 12 : Somme des premiers entiers *19*

1 Pourquoi définir notre langage algorithmique ?

2 Structure d'un algorithme

2.1 Exemple d'algorithme : calculer le périmètre d'un cercle

Un exemple d'algorithme/programme est donné ci-dessous. Il décrit comment obtenir le périmètre d'un cercle à partir de son diamètre. Cet exemple est volontairement très simple.

Listing 1 – Programme F pour calculer le périmètre d'un cercle

```
1  !**********************************************************************
2  ! Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  ! Version : 1.1
4  ! Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  ! Titre : Déterminer le perimetre d'un cercle à partir de son rayon.
6  !**********************************************************************
7
8  PROGRAM main
9      REAL,PARAMETER:: PI=3.14159
10     REAL:: rayon  ! le rayon du cercle lu au clavier
11     REAL:: perimetre ! le perimetre du cercle
12
13     ! Saisir le rayon
14     PRINT*,"Rayon = "
15     READ*,rayon
16
17     ! Calculer le perimetre
18     perimetre = 2 * PI * rayon   ! par definition
19     !{ perimetre = 2 * PI * rayon }
20
21     ! Afficher le perimetre
22     PRINT*,"Le perimetre est : ", perimetre
23 END PROGRAM main
```

2.2 Structure de l'algorithme

La structure d'un programme F est proche de celle d'un algorithme. Le fichier, qui doit avoir l'extension .f95, commence par un cartouche faisant apparaître le nom des auteurs du programme, la version ou la date de réalisation et l'objectif du programme. Ces éléments sont mis dans des commentaires et sont donc ignorés par le compilateur.

Les déclarations et instructions sont regroupées entre **PROGRAM** main et **END PROGRAM** main, d'abord les déclarations, puis les instructions. La constante PI est définie par le mot clé **PARAMETER**. Il correspond donc à une définition. Notons qu'en F les constantes sont également typées. Les instructions sont les mêmes que celles présentées dans l'algorithme même si elles ont une forme un peu différente.

2.3 Identificateurs

Un identificateur est un mot de la forme : une lettre (y compris le souligné) suivie d'un nombre quelconque de lettres et de chiffres.

Attention, il n'est pas possible d'utiliser les lettres accentuées en F.

2.4 Commentaires

Un commentaire commence par `!` et se termine à la fin de la ligne.

Pour représenter une propriété du programme, nous utiliserons `!{...}`.

3 Variables

3.1 Qu'est ce qu'une variable ?

3.2 Définition d'une variable

En F, on commence par mettre le type suivi de `::` et du nom de la variable.

```
REAL :: prix_unitaire           ! prix unitaire dun article (en euros)
INTEGER :: quantite             ! quantité darticles commandés
CHARACTER (LEN=20) :: nom       ! nom de larticle
```

Les types et leur signification seront présentés dans la suite du cours.

Il est possible de déclarer plusieurs variables du même type en les séparant par des virgules mais ceci est déconseillé sauf si le même commentaire s'applique à toutes les variables.

```
INTEGER :: a, b, c      ! trois entiers
```

4 Types fondamentaux

Les opérateurs de comparaison entre les types fondamentaux se notent : `<`, `>`, `<=`, `>=`, `==` et `/=`. Notez bien que l'opérateur de comparaison (égalité) est noté avec deux fois le caractère `=`.

4.1 Les entiers

Le type entier se note **`INTEGER`**. Cependant, un qualificatif peut préciser sa taille, mais cela n'est utilisé que pour la compatibilité avec d'autres langages.

Le reste de la division entière n'existe pas. Il peut être facilement calculé (10-10/3) et la division entière de a par b se note tous simplement `a/b`. Il faut faire attention à ne pas la confondre avec la division sur les réels.

```
10 / 3          ! 3 (le quotient de la division entière de 10 par 3)
1 / 2           ! 0 (le quotient de la division entière de 1 par 2)
abs( 5 )        ! 5 (l'entier est mis entre parenthèses (cf sousprogrammes))
```

Notons que les débordements de capacité sur les opérations entières ne provoquent aucune erreur à l'exécution... mais le résultat calculé est bien sûr faux par rapport au résultat attendu !.

4.2 Les réels

Il existe deux types réels, les réels simple précision appelés **`REAL`** et les réels double précision nécessitant un paramètre supplémentaire.

La valeur absolue se note `ABS`.

La partie entière d'un réel s'obtient par la fonction `INT` : `INT(3.14)` correspond à 3. Ceci revient à convertir en entier le réel 3.14.

4.3 Les booléens

Le type booléen est **`LOGICAL`**. Les deux valeurs se note `.TRUE.` et `.FALSE.`. Les opérateurs logiques se notent `.AND.` pour **`Et`**, `.OR.` pour **`Ou`** et `.NOT.` pour **`Non`**. Notez bien le point avant et après les opérateurs booléens et les deux constantes booléennes.

Les expressions booléennes sont évaluées en court-circuit (on parle d'évaluation partielle), c'est-à-dire que dès que le résultat d'une expression est connu, l'évaluation s'arrête. Par exemple, `.TRUE. .OR. expression` sera évaluée à `.TRUE.` sans calculer la valeur de `expression`.

4.4 Les caractères

Le type caractère se note CHARACTER(LEN=1). Ce n'est qu'un cas particulier des chaînes de caractères du paragraphe suivant. Les constantes caractères se notent en algorithmique en utilisant uniquement le guillemet. Les fonction `Chr` et `Ord` se note respectivement `CHAR(i)` et `ICHAR(c)`

```
c = "A"         ! la valeur de c est "A"
i = ICHAR(c)    ! la valeur de i est 65, code ASCII de "A"
```

4.5 Les chaînes de caractères

La longueur réservée est imposée à la déclaration.

```
CHARACTER(LEN=10) :: chaine
```

C'est une borne maximale : on peut ensuite écrire `chaine = "bonjour"` qui ne contient que six caractères.

5 Constantes

En F, les constantes sont définies en utilisant **`PARAMETER`** à la suite du type :

```
REAL, PARAMETER :: PI = 3.1415  ! Valeur de PI
INTEGER, PARAMETER :: MAJORITE = 18     ! Âge correspondant à la majorité
REAL, PARAMETER ::  TVA = 19.6  ! Taux de TVA en vigueur au 15/09/2000 (en %)
REAL, PARAMETER ::  CAPACITE = 120
                        ! Nombre maximum d'étudiants dans une promotion
CHARACTER, PARAMETER(LEN=30) :: INTITULE = "Algorithmique et programmation"
                        ! par exemple
```

6 Expressions

7 Instructions d'entrée/sorties

7.1 Opération d'entrée

```
READ*, var,var1
```

Notons que l'on pourrait indiquer, à la place de l'étoile, le type de variable, mais le compilateur s'arrange tout seul pourvu que les différentes variables à lire sont séparées par des espaces ou un tabulateur.

7.2 Opération de sortie

On écrit

```
PRINT*,var,var1
```

pour écrire deux variables sur la même ligne.

L'instruction **PRINT*** permet de sauter une ligne.

Exercice 1 : Cube d'un réel
Écrire un programme qui affiche le cube d'un nombre réel saisi au clavier.
Solution :

```
1  R0 : Afficher le cube d'un nombre réel
2
3  Tests :
4      0 -> 0
5      1 -> 1
6      2 -> 8
7      -2 -> -8
8      1.1 -> 1,331
9
10 R1 : Raffinage De « Afficher le cube d'un nombre réel »
11   | Saisir un nombre réel       x: out Réel
12   | Afficher le cube de x       x: in Réel
13
14 R2 : Raffinage De « Afficher le cube de x »
15   | Écrire(x * x * x)
```

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.2
!* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
!* Objectif : afficher le cube d'un nombre reel
!*********************************************************************

PROGRAM main

        REAL:: x   ! un nombre saisi par l'utilisateur
        ! Saisir un nombre reel
        PRINT*,"Nombre = "
        READ*,x
        !Afficher le cube de x
        PRINT*,"Son cube est : ", x * x * x
END PROGRAM main
```

8 Affectation

L'affectation se note avec un signe =.

Attention : il ne faut pas confondre = (affectation) et == (opérateur de comparaison : égalité).

Exercice 2 : Permuter deux caractères

Écrire un programme qui permute la valeur de deux variables `c1` et `c2` de type caractère.

Solution : Le principe est d'utiliser une variable intermédiaire (tout comme on utilise un récipient intermédiaire si l'on veut échanger le contenu de deux bouteilles).

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.2
!* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
!* Objectif : permuter deux caracteres
!*********************************************************************

PROGRAM main
        CHARACTER(LEN=1):: c1, c2  ! les deux caracteres a permuter
        CHARACTER(LEN=1):: tmp     ! notre intermediaire
        ! initialiser c1 et c2
        c1 = "A"
        c2 = "Z"
        ! permuter c1 et c2
        tmp = c1
        c1 = c2
        c2 = tmp
        ! afficher pour verifier
        PRINT*,"c1 = ",c1," et c2 = ", c2
END PROGRAM main
```

Exercice 3 : Cube d'un réel (avec une variable)

Reprenons l'exercice 1.

3.1 Utiliser une variable intermédiaire pour le résoudre.
Solution : On reprend le même R0 et les mêmes tests. En fait, seule la manière de résoudre le problème change.

```
R1 : Raffinage De « Afficher le cube d'un nombre réel »
  | Saisir un nombre réel       x: out Réel
  | Calculer le cube de x       x: in Réel ; cube: out Réel
  | Afficher le cube

R2 : Raffinage De « Afficher le cube de x »
  | cube <- x * x * x
```

```
!*****************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.1
!* Revision : Xuan Meyer <XuanMi.Meyer@ensiacet.fr>
!* Objectif : afficher le cube d'un nombre reel
!*****************************************************************

PROGRAM main

        REAL:: x   ! un nombre saisi par l'utilisateur
        REAL:: cube   ! le cube de x
        ! Saisir un nombre reel
        PRINT*,"Nombre = "
        READ*,x
        ! Calculer le cube de x
        cube = x * x * x
        !Afficher le cube de x
        PRINT*,"Son cube est : ", cube
END PROGRAM main
```

3.2 Quel est l'intérêt d'utiliser une telle variable ?
Solution : L'intérêt d'utiliser une variable intermédiaire est d'améliorer la lisibilité du programme car elle permet de mettre un nom sur une donnée manipulée. Ici on nomme `cube` la donnée `x * x * x`.

De plus, ceci nous a permis, au niveau du raffinage, de découpler le calcul du cube de son affichage. Il est toujours souhaitable de séparer calcul des opérations d'entrées/sorties car l'interface avec l'utilisateur est la partie d'une application qui a le plus de risque d'évoluer.

3.3 Exécuter à la main l'algorithme ainsi écrit.
Solution : À faire soi-même !

9 Structures de contrôle

9.1 Enchaînement séquentiel

La séquence s'exprime comme en algorithmique.

9.2 Instructions conditionnelles

9.2.1 Conditionnelle `Si ... Alors ... FinSi`

```
IF (condition) une seule instruction
```

ou

```
IF (condition) THEN
    instruction
    ...
    instruction
ENDIF
```

Cette deuxième forme est largement préférable car dans la première on ne peut mettre qu'une seule instruction contrôlée par le **`IF`** alors que dans la seconde, on peut en mettre autant qu'on veut.

Exercice 4 : Une valeur entière est-elle paire ?
Écrire un algorithme qui lit une valeur entière au clavier et affiche « paire » si elle est paire.
Solution :

```
1  R0 : Afficher « paire » si une valeur entière saisie au clavier est paire
2
3  tests :
4      2 -> paire
5      5 -> -----
6      0 -> paire
7
8  R1 : Raffinage De « Afficher ... »
9    | Saisir la valeur entière n
10   | Afficher le verdict de parité
11
12 R2 : Raffinage De « Afficher le verdict de parité »
13   | Si n est paire Alors
14   |   | Écrire("paire")
15   | FinSi
16
17 R3 : Raffinage De « n est paire »
18   | Résultat <- n Mod 2 = 0
```

Dans le raffinage précédent un point est à noter. Il s'agit du raffinage R2 qui décompose « Afficher le verdict de parité ». Nous n'avons pas directement mis la formule « `n Mod 2 = 0` ». L'intérêt est que la formulation « n est paire » est plus facile à comprendre. Avec la formule, il faut d'abord comprendre la formule, puis en déduire sa signification. « n est paire » nous indique ce qui nous intéresse comme information (facile à lire et comprendre) et son raffinage (R3) explique comment on détermine si n est paire. Le lecteur peut alors vérifier la formule en sachant ce qu'elle est sensée représenter.

Raffiner est quelque chose de compliquer car on a souvent tendance à descendre trop vite dans les détails de la solution sans s'arrêter sur les étapes intermédiaires du raffinage alors que ce sont elles qui permettent d'expliquer et de donner du sens à la solution.

Dans cet exercice, vous vous êtes peut-être posé la question : « mais comment sait-on que n est paire ». Si vous avez trouvé la solution vous avez peut-êre donnée directement la formule alors que le point clé est la question. Il faut la conserver dans l'expression de votre algorithme ou programme, donc en faire une étape du raffinage.

Si vous arrivez sur une étape que vous avez du mal à décrire, ce sera toujours une indication d'une étape qui doit apparaître dans le raffinage. Cependant, même pour quelque chose de simple, que vous savez faire directement, il faut être capable de donner les étapes intermédiaires qui conduisent vers et expliquent la solution proposée. Ceci fait partie de l'activité de construction d'un programme ou algorithme.

Remarque : Il est généralement conseillé d'éviter de mélanger traitement et entrées/sorties. C'est pourtant ce qui a été fait ci-dessus. On aurait pu écrire le premier niveau de raffinage différemment en faisant.

```
1  R1 : Raffinage De « Afficher ... »
2    | Saisir la valeur entière            n: out Entier
3    | Déterminer la parité de n           n: in ; paire: out Booléen
4    | Afficher le verdict de parité       paire: in Booléen
5
6  R2 : Raffinage De « Déterminer la parité de n »
7    | parité <- (n Mod 2) = 0
8
9  R2 : Raffinage De « Afficher le verdict de parité »
10   | Si paire Alors
11   |  | Écrire("paire")
12   | FinSi
```

On constate ici que la variable intermédiaire « paire » permet d'avoir un programme plus lisible car on a donné un nom à la quantité `(n Mod 2) = 0`.

```
1  !**********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.1
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Afficher « paire » si une valeur entière est paire.
6  !**********************************************************************
7
8  PROGRAM main
9
10         INTEGER:: n   ! valeur saisie au clavier
11         ! Saisir la valeur entiere n
12         PRINT*,"Valeur = "
13         READ*,n
14         ! Afficher le verdict de parite
15         IF (n/2*2 == n) THEN !{ n est paire }
16               PRINT*,"paire"
17         ENDIF
18 END PROGRAM main
```

9.2.2 Conditionnelle `Si ... Alors ... Sinon ... FinSi`

```
IF (condition) THEN
    instruction
    ...
ELSE
    instruction
    ...
ENDIF
```

Exercice 5 : Maximum de deux valeurs réelles

Étant données deux valeurs réelles lues au clavier, afficher à l'écran la plus grande des deux.

Solution :

```
R0 : Afficher le plus grand de deux réels saisis au clavier

tests :
    1 et 2 -> 2
    2 et 1 -> 1
    3 et 3 -> 3

R1 : Raffinage De « Afficher le plus grand de deux réels ... »
  | Saisir les deux réels       x1, x2 : out Réel
  | Déterminer le maximum       x1, x2 : in ; max : out Réel
  | Afficher le maximum

R2 : Raffinage De « Déterminer le maximum »
  | Si x1 > x2 Alors
  |  | max <- x1
  | Sinon
  |  | max <- x2
  | FinSi
```

```
!**********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.1
!* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
!* Objectif : Determiner le max de deux valeurs reelles
!**********************************************************************

PROGRAM main

        REAL:: x1, x2 ! les deux reels saisis au clavier
        REAL:: max    !le plus grand de x1 et x2
        ! Saisir les deux reels
        PRINT*,"Deux reels :"
        READ*,x1, x2
        ! Déterminer le maximum
        IF (x1 > x2) THEN
            max = x1
        ELSE
                max = x2
        ENDIF
        ! Afficher le maximum
```

```
22          PRINT*,"max(", x1,",", x2,")␣=", max
23  END PROGRAM main
```

9.2.3 La clause `SinonSi`

```
IF (condition_1) THEN
    instruction_{1,1}
    ...
ELSEIF (condition_2) THEN
    instruction_{2,1}
    ...
...
ELSEIF (condition_{n,1}) THEN
    instruction_{n,1}
ELSE
    instruction_1
ENDIF
```

Exercice 6 : Signe d'un entier
Étant donné un entier lu au clavier, indiquer s'il est nul, positif ou négatif.
Solution :

```
1  R0 : Afficher le signe d'un entier
2
3  tests :
4       2 -> positif
5       0 -> nul
6      -1 -> négatif
7
8  R1 : Raffinage De « Afficher le signe d'un entier »
9    | Saisir un entier n          n: out Entier
10   | Afficher le signe de n      n: in
11
12 R2 : Raffinage De « Afficher le signe de n »
13   | Si n > 0 Alors
14   |   | Écrire("positif");
15   | SinonSi  n < 0 Alors
16   |   | Écrire("positif");
17   | Sinon  { Non (n > 0) Et Non (n < 0)  donc  N = 0 }
18   |   | Écrire("nul");
19   | FinSi
```

Le principe est d'utliser un **`SinonSi`** car les trois cas sont exclusifs.

```
1  !*********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.1
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Afficher le signe d'un entier.
6  !*********************************************************************
7
```

```
8  PROGRAM main
9
10         INTEGER:: n   ! entier saisi au clavier
11         ! Saisir un entier n
12         PRINT*,"Valeur␣entiere␣:␣"
13         READ*,n
14         ! Afficher le signe de n
15         IF (n > 0)THEN
16                 PRINT*,"positif"
17         ELSEIF (n < 0)THEN
18                 PRINT*,"negatif"
19         ELSE
20                 PRINT*,"nul"
21         ENDIF
22 END PROGRAM main
```

9.2.4 Conditionnelle `Selon`

```
SELECT CASE (expression)
    CASE(choix_1)
            Séquence1
    CASE(choix_2)
            Séquence2
    ...
    CASE(choix_n)
            Séquencen
    CASE DEFAULT
            Séquence
END SELECT
```

expression est nécessairement de type entier ou chaîne de caractères. Un intervalle est séparé par « : » et si l'on veut plusieurs valeurs, on les séparent par une virgule. Bien sûr choix_1, choix_2, choix_n sont exclusifs.

Principe : L'exécution commence par les instructions de la 1re expression constante qui correspond à l'expression du **`SELECT CASE`** et le branchement est effectué au **`CASE`** correspondant.

Exercice 7 : Réponse

Écrire un programme qui demande à l'utilisateur de saisir un caractère et qui affiche « affirmatif » si le caractère est un « o » (minuscule ou majuscule), « négatif » si c'est un « n » (minuscule ou majuscule) et « ? ! ? ! ? ! ? » dans les autres cas.

Solution :

```
1  !*********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.1
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Répondre par « affirmatif », « négatif » ou « ?!?!?!? ».
6  !*********************************************************************
7
8  PROGRAM main
```

```
9    CHARACTER(LEN=1):: reponse  ! caractere lu au clavier
10   ! saisir le caractère
11   PRINT*,"Votre reponse (o/n) : "
12   READ*,reponse
13   !afficher la reponse
14   SELECT CASE (reponse)
15         CASE("o","O")
16                PRINT*,"Affirmatif"
17         CASE ("n","N")
18                PRINT*,"Negatif"
19         CASE DEFAULT
20                PRINT*,"?!?!?!?"
21   END SELECT
22 END PROGRAM main
```

9.3 Instructions de répétitions

9.3.1 Répétition `TantQue`

Cette structure n'existe pas directement en F. On l'écrit

```
DO
    IF (.NOT. condition) EXIT
    instruction
    ...
ENDDO
```

À la place de `(.NOT. condition)` on peut écrire la condition alternative (si `condition` est `a>b`, `.NOT. condition` est `a<=b`).

Exercice 8 : Somme des premiers entiers (TantQue)
Calculer la somme des n premiers entiers.
Solution : Une solution algorithmique sous forme de raffinages peut-être la suivante :

```
1  R0 :  Afficher la somme des n premiers entiers
2
3  R1 : Raffinage De « Afficher la somme des n premiers entiers »
4      Saisir la valeur de n (pas de contrôle)     n: out Entier
5      { n >= 0 }
6      Calculer la somme des n premiers entiers    n: in; somme: out Entier
7      Afficher la somme                           somme: in Entier
8
9  R2 : Raffinage De « Calculer la somme des n premiers entiers »
10     somme <- 0                  somme: out
11     i <- 1                      i: out Entier
12     TantQue i <= n Faire        i, n: in
13         { Variant : n - i + 1 }
14         { Invariant : somme = sum_{j=0}^{i-1} j }
15         somme <- somme + i      somme, i: in; somme: out
16         i <- i + 1              i: in; i: out
17     FinTQ
```

Intéressons nous à la condition après le **TantQue**. On a la propriété suivante :

```
    (i > n)                -- sortie du TantQue : Non (i <= n)
Et (n - i + 1 >= 0)        -- variant >= 0
Et (somme = ∑_{j=1}^{i-1} j) -- invariant
```

Les deux premières expressions s'écrivent

```
(i > n ) Et (i <= n+1)
```

On en déduit : `i = n + 1`.

La troisième donne alors :

$$somme = \sum_{j=1}^{n} j$$

C'est bien le résultat demandé !

Bien entendu, il faut aussi prouver que le variant est toujours positif et qu'il décroit strictement (on incrémente i de 1 donc on diminue le variant de 1). Il faut également prouver que l'invariant est toujours vrai.

Commençons par le variant. Montrons par récurrence sur le nombre de passage dans la boucle que la **variant est toujours positif**.

Si le nombre de passage est nul, donc avant le premier passage, on a : $V_0 = n - i1 = n - 1 + 1 = n$. Par hypothèse sur n (saisie contrôlée), on a bien $V_0 \geq 0$

Supposons la propriété vraie pour le passage p. On a donc : $V_p =\geq 0$

Montrons que V_{p+1} est vraie. On notera avec des primes les variables de V_{p+1} au lieu d'utiliser des indices en p.

On a : $V_{p+1} = n' - i' + 1$

Si on parle de V_{p+1} c'est qu'on est passé dans la boucle. Donc la condition du **TantQue** est vraie. On a donc $i \leq n$.

Or on a $n' = n$ et $i' = i + 1$ (passage une fois dans la boucle).

Donc $V_{p+1} = n - (i + 1) + 1 = n - i + 1 - 1 = n - i$ Comme $i \leq n$, on a bien $V_{p+1} \geq 0$.

Par récurrence, on a montrer que le variant est toujours positif.

Montrons que le **variant décroit strictement**. On $V_{p+1} = n' - i' + 1 = n - i + 1 - 1 = (n - i + 1) - 1 = V_p - 1$. On a bien $V_{p+1} < V_p$.

On a donc montrer la terminaison de la boucle.

Remarque : Dans la formulation initiale du R1, j'avais oublié la propriété `{ n >= 0 }`. Elle était bien sûr implicite. Essayer de montrer que le variant était toujours positif m'a permis de penser à l'expliciter.

Montrons maintenant que l'**invariant est toujours vrai**. On utilise aussi une récurrence sur le nombre p de passage dans la boucle.

Avant le premier passage, on a $\sum_{j=0}^{i-1} j = \sum_{j=0}^{0} j = 0$ et on a $somme = 0$. Donc I_0 est vrai.

Supposons I_p vrai. On a donc : $somme = \sum_{j=0}^{i-1} j$

Montrons que I_{p+1} est vrai. Si on parle de I_{p+1}, c'est qu'on passe une nouvelle fois dans la boucle. On a donc : $i \geq n$.

Les valeurs de n, i et somme deviennent :

$s' = s + i$

$i' = i + 1$
$n' = n$
$s' = \sum_{j=0}^{i-1} j + i$ par hypothèse de récurrence. $s' = \sum_{j=0}^{i} j$ $s' = \sum_{j=0}^{(i+1)-1} j$ $s' = \sum_{j=0}^{i'-1} j$
Donc on a bien I_{p+1}.
Par récurrence, on montre donc que l'invariant est toujours vrai.

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.0
!* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
!* Objectif : Calculer la somme des n premiers entiers
!*            avec une instruction Tant Que.
!*********************************************************************

PROGRAM main

    INTEGER:: n  ! valeur lue au clavier
    INTEGER:: i  !parcourir les entiers de 1 à n
    INTEGER:: somme  ! somme des entiers de 0 à i
    ! saisir la valeur de n (sans controle)
    PRINT*,"Nombre d'entiers : "
    READ*,n
    ! calculer la somme des n premiers entiers
    somme = 0 ! initialisation de la somme a 0
    i = 1
    DO
        IF (i > n)EXIT
        !{ Variant : n   i + 1 }
        !{ Invariant : somme =  « somme des entiers de 0 à i-1 »}
        somme = somme + i
        i=i+1
    ENDDO
    ! afficher la somme
    PRINT*,"La somme est : ", somme
END PROGRAM main
```

Exercice 9 : Saisie contrôlée d'un numéro de mois

On souhaite réaliser la saisie du numéro d'un mois (compris entre 1 et 12) avec vérification. Le principe est que si la saisie est incorrecte, le programme affiche un message expliquant l'erreur de saisie et demande à l'utilisateur de resaisir la donnée.

9.1 Utiliser un **`TantQue`** pour réaliser la saisie contrôlée.

9.2 Généraliser l'algorithme au cas d'une saisie quelconque.

Solution :

```
!*********************************************************************
!* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
!* Version : 1.0
!* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
!* Objectif : Saisir le numero d'un mois avec controle.
!*********************************************************************
```

```
7
8  PROGRAM main
9      INTEGER:: mois   !  le numero du mois
10     ! Saisir le numero de mois
11     PRINT*,"Numero␣du␣mois␣:␣"
12     READ*,mois
13     ! Traiter les erreurs eventuelles
14     DO
15         IF(mois>=1 .AND. mois<=12) EXIT !{ mois correct }
16         ! Signaler l'erreur de saisie
17         PRINT*,"Vous␣devez␣donner␣un␣numero␣compris␣entre␣1␣et␣12␣!"
18         ! Saisir un nouveau numero de mois
19         PRINT*,"Numero␣du␣mois␣:␣"
20         READ*,mois
21     ENDDO
22     ! Afficher le numero saisie
23     PRINT*,"Le␣numero␣du␣mois␣est␣donc␣:␣", mois
24 END PROGRAM main
```

9.3.2 Répétition Répéter ... JusquÀ

```
DO
    instruction
    ...
    IF (condition) EXIT
ENDDO
```

Exercice 10 : Plusieurs sommes des n premiers entiers
Écrire un programme qui affiche la somme des n premiers entiers naturels, n étant un entier saisi au clavier. Le programme devra proposer la possibilité à l'utilisateur de recommencer le calcul pour un autre entier.
Solution : Le raffinage peut être décrit ainsi.

```
1  R0 : Afficher la somme des n premiers entiers avec possibilité de recommencer
2
3  R1 : Raffinage De « R0 »
4    | Répéter
5    |   | Afficher la somme des n premiers entiers
6    |   | Demander si l'utilisateur veut recommencer       reponse: out
7    | JusquÀ réponse est non
```

Le raffinage de « Afficher la somme des n premiers entiers » a déjà été donné dans un exercice précédent. On peut donc directement en déduire l'algorithme.

```
1  !*****************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.0
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Calculer la somme des n premiers entiers avec possibilite
6  !*            de recommencer.
```

```
7  !**********************************************************************
8
9  PROGRAM main
10
11     CHARACTER(LEN=1) :: reponse ! reponse de l'utilisateur
12     INTEGER:: n  ! valeur lue au clavier
13     INTEGER:: i  !parcourir les entiers de 1 à n
14     INTEGER:: somme  ! somme des entiers de 0 à i
15     ! saisir la valeur de n (sans contrôle)
16     DO
17       PRINT*,"Nombre d'entiers : "
18       READ*,n
19       ! calculer la somme des n premiers entiers
20       somme = 0  !IMPORTANT  initialisation de la somme
21       i = 1
22       DO
23          IF ( i > n ) EXIT
24          !{ Variant : n   i + 1 }
25          !{ Invariant : somme = }
26          somme = somme + i
27          i=i+1
28       ENDDO
29     ! afficher la somme
30       PRINT*,"La somme est : ", somme
31       PRINT*,"Encore(o/n) ?"
32       READ*,reponse
33       IF(reponse == "n" .OR. reponse =="N")EXIT
34     ENDDO
35
36 END PROGRAM main
```

Exercice 11 : Saisie contrôlée d'un numéro de mois

On souhaite réaliser la saisie du numéro d'un mois (compris entre 1 et 12) avec vérification. Le principe est que si la saisie est incorrecte, le programme affiche un message expliquant l'erreur de saisie et demande à l'utilisateur de resaisir la donnée.

On utilisera un **`Répéter`** pour réaliser la saisie contrôlée.

Généraliser l'algorithme au cas d'une saisie quelconque.

Solution :

```
1  !**********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.0
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Saisir le numéro d'un mois avec controle.
6  !**********************************************************************
7
8  PROGRAM main
9      INTEGER:: mois   !  le numero du mois
10     ! Saisir le numero de mois
11     ! Traiter les erreurs eventuelles
12     DO
```

```
13          PRINT*,"Numero␣du␣mois␣:␣"
14          READ*,mois
15          ! Signaler l'erreur de saisie
16          PRINT*,"Vous␣devez␣donner␣un␣numero␣compris␣entre␣1␣et␣12␣!"
17          ! Saisir un nouveau numero de mois
18          PRINT*,"Reessayez␣:␣"
19          READ*,mois
20          IF(mois>=1 .AND. mois<=12)EXIT !{ mois correct }
21      ENDDO
22      ! Afficher le numéro saisie
23      PRINT*,"Le␣numero␣du␣mois␣est␣donc␣:␣", mois
24  END PROGRAM main
```

9.3.3 Répétition `Pour`

```
DO val = val_min, val_max, pas
    instruction
    ...
ENDDO
```

Le **`DO`** du F est compatible avec le **`Pour`** algorithmique. Le pas peut être positif, ou négatif et différent de 1. Par contre la variable de la boucle est obligatoirement de type entier. Notons que si le pas est de un, il peut être omis.

```
DO val = val_min, val_max
    instruction
    ...
ENDDO
```

Exercice 12 : Somme des premiers entiers
Calculer la somme des n premiers entiers.
Solution :

```
1  !**********************************************************************
2  !* Auteur : Denis Barreteau <Denis.Barreteau@ensiacet.fr>
3  !* Version : 1.0
4  !* Revision : Xuan Meyer < XuanMi.Meyer@ensiacet.fr>
5  !* Objectif : Calculer la somme des n premiers entiers
6  !*            avec une instruction Repeter Pour
7  !**********************************************************************
8
9  PROGRAM main
10
11     INTEGER:: n  ! valeur lue au clavier
12     INTEGER:: i  !parcourir les entiers de 1 a n
13     INTEGER:: somme  ! somme des entiers de 0 a i
14     ! saisir la valeur de n (sans controle)
15     PRINT*,"Nombre␣d'entiers␣:␣"
16     READ*,n
17     ! calculer la somme des n premiers entiers
18     somme = 0
```

```
19      DO i=1,n,1
20          !{ Variant : n   i + 1 }
21          !{ Invariant : somme = }
22          somme = somme + i
23      ENDDO
24      ! afficher la somme
25      PRINT*,"La somme est : ", somme
26  END PROGRAM main
```

9.3.4 Quelle répétition choisir ?