

ALGO 1.1 – Correction TD N°5.

Patrick Poulingeas.

Exercice 1.

Remarque : On ne s'occupe pas de la situation où l'utilisateur saisit un entier strictement négatif.

Rappel : $0! = 1$

Calcul de la factorielle d'un entier naturel (avec une structure itérative « Pour »).

Variables

```
n : entier
factorielle : entier
indice : entier
{

// Saisie de la donnée
Afficher (« Entrez un nombre entier positif : »)
Saisir(n)

factorielle ← 1
si n ≠ 0 alors
    Pour indice de 1 à n faire
        factorielle ← factorielle * indice

// Affichage du résultat
Afficher (« La factorielle vaut : »,factorielle)
}
```

Calcul de la factorielle d'un entier naturel (avec une structure itérative « Tant que »).

Variables

```
n : entier
factorielle : entier
indice : entier
{

// Saisie de la donnée
Afficher (« Entrez un nombre entier positif : »)
Saisir(n)

factorielle ← 1
indice ← 1 // Optimisation : initialiser indice à 2
```

```

Tant que indice ≤ n faire
{
    factorielle ← factorielle * indice
    indice ← indice + 1
}

// Affichage du résultat
Afficher(« La factorielle vaut : »,factorielle)
}

```

Calcul de la factorielle d'un entier naturel (avec une structure itérative « Répéter »).

Variables

```

n : entier
factorielle : entier
indice : entier
{

// Saisie de la donnée
Afficher (« Entrez un nombre entier positif : »)
Saisir(n)

factorielle ← 1
indice ← 1
Répéter
{
    factorielle ← factorielle * indice
    indice ← indice + 1
}
Jusqu'à indice > n

// Affichage du résultat
Afficher(« La factorielle vaut : »,factorielle)
}

```

Exercice 2.

Remarque : On ne s'occupe pas de la situation où l'utilisateur saisit un entier ≤ 0 .

Rappel : 1 n'est pas premier.

On dit qu'un nombre naturel $n \geq 2$ est premier s'il n'est divisible que par 1 et par lui-même.

2 est premier (C'est le seul nombre pair premier).

Le début de la suite (infinie) des nombres premiers est : 2, 3, 5, 7, 11, 13, 17,...

Test de primalité pour un entier strictement positif.

Variables

```

nombre : entier
indice : entier

```

```

premier : booléen
{
  // Saisie de la donnée
  Afficher(« Entrer un entier strictement positif : »)
  Saisir(nombre)

  premier ← vrai
  si nombre = 1 alors
    premier ← faux
  sinon
    {
      indice ← 2
      Tant que indice ≤ nombre - 1 et premier faire
        {
          si nombre mod indice = 0 alors
            premier ← faux
            indice ← indice + 1
          }
        }

  // Affichage du résultat
  si premier alors
    Afficher(nombre, « est premier. »)
  sinon
    Afficher(nombre, « n'est pas premier. »)
}

```

Exercice 3.

Remarque : On ne s'occupe pas de la situation où l'utilisateur saisit un entier ≤ 0 .

Détermination du caractère parfait ou non d'un nombre entier strictement positif.

Variables

```

nombre : entier
diviseur : entier
somme_diviseurs : entier
nombre_parfait : booléen
{
  // Saisie de la donnée
  Afficher(« Entrer un entier strictement positif : »)
  Saisir(nombre)

  somme_diviseurs ← 0
  diviseur ← 1
  Tant que diviseur < nombre faire
    {
      si nombre mod diviseur = 0 alors
        somme_diviseurs ← somme_diviseurs + diviseur
    }
}

```

```

        diviseur ← diviseur + 1
    }

nombre_parfait ← (nombre = somme_diviseur) et (nombre ≠ 1)

// Affichage des résultats
si nombre_parfait alors
    Afficher(nombre, « est un nombre parfait. »)
sinon
    Afficher(nombre, « n'est pas un nombre parfait. »)
}

```

Détermination des nombres parfaits entre 1 et n.

Variables

```

n : entier
nombre : entier
diviseur : entier
somme_diviseurs : entier
nombre_parfait : booléen

{
    // Saisie de la donnée
    Afficher(« Entrer un entier strictement positif : »)
    Saisir(n)

    Pour nombre de 1 à n faire
    {
        // On reprend l'algorithme déterminant si nombre est parfait
        somme_diviseurs ← 0
        diviseur ← 1
        Tant que diviseur < nombre faire
        {
            si nombre mod diviseur = 0 alors
                somme_diviseurs ← somme_diviseurs + diviseur
                diviseur ← diviseur + 1
            }
        nombre_parfait ← (nombre = somme_diviseur) et (nombre ≠ 1)

        si nombre_parfait alors
            Afficher(nombre)
        }
    }
}

```

Exercice 4.

Remarque : On ne s'occupe pas de la situation où l'utilisateur saisit un ou deux entiers négatifs.

Calcul du pgcd de deux nombres a et b strictement positifs par l'algorithme d'Euclide.

Variables

```
a,b : entier
q,r : entier
pgcd : entier
a1,b1 : entier // Variables auxiliaires sur lesquelles on effectue les calculs successifs
{

// Saisie des données
Afficher(« Entrez deux nombres strictement positifs. »)
Saisir(a,b)

// On affecte à b1 le maximum de a et b,
// et on affecte à a1 le minimum de a et b
si a < b alors
{
    b1 ← b
    a1 ← a
}
sinon
{
    b1 ← a
    a1 ← b
}

// Calcul du pgcd de a1 et b1 (qui est aussi celui de a et b)
Répéter
{
    q ← b1 / a1 // La division est une division entière
                // car b1 et a1 sont des entiers.
    r ← b1 mod a1 // On a : b1 = a1*q + r
    b1 ← a1
    a1 ← r
}
Jusqu'à r = 0

pgcd ← b1

// Affichage du résultat
Afficher(« Le pgcd de »,a, « et »,b, « est : »,pgcd)
}
```

Exercice 5.

Calcul d'une racine carrée par itérations.

Variables

```
x : réel
y_n : réel
epsilon : réel
choix : caractère
```

```

{
  répéter
  {
    Afficher(« Voulez-vous faire l'approximation d'une racine carrée (O/N) ? »)
    Saisir(choix)

    si choix = 'O' alors
    {
      // Saisie des données pour le calcul
      Afficher(« Entrez un réel positif : »)
      Saisir (x)
      Afficher (« Entrez la précision epsilon désirée : »)
      Saisir (epsilon)

      // Contrôle de la validité des données pour le calcul
      si (x < 0) ou (epsilon ≤ 0) alors
        Afficher(« Données saisies incorrectes. »)
      sinon
        // Algorithme de calcul
        {
          y_n ← 1

          répéter
          {
            y_n ← 0.5*(x/y_n + y_n)

            Afficher(« L'approximation de la racine carrée de »,x,
              « est actuellement : », y_n)
          }
          jusqu'à abs(racine_carrée(x) – y_n) ≤ epsilon

          Afficher(« On a atteint la précision voulue. »)
        } // Fin du bloc contenant l'algorithme de calcul
      } // Fin du bloc correspondant à choix = 'O'
    } // Fin du bloc de la structure itérative répéter
    jusqu'à choix = 'N'

    // Fin du programme quand l'utilisateur a saisi 'N' comme réponse
  }
}

```