Recommandations de programmation

```
-------------------------------------------------------------------------------
-- Fichier          : Pizza.adb
-- Auteur           : R. Chelouah
-- Date de création : 25.09.2008
--
-- But              : Ce programme calcule le prix [Euro] d'une pizza selon
--                    une formule donnée.
--                    Cette formule fait intervenir  deux paramètres utilisateur:
--                    1. le diamètre [cm] souhaité de la pizza
--                    2. le nombre d'ingrédients souhaité sur la pizza
--
-- Remarque(s)      : 1. La pizza est supposée circulaire.
--                    2. La validité des entrées utilisateur n'est pas testée.
--                    3. Le prix affiché [Frs] pour la pizza correspond au prix
--                       exact de la pizza, arrondi à l'entier le plus proche.
-- Date de modifi   :
-- Raison           :
--
-- Modules appeles  : Ada.Text_Io,Ada.Integer_Text_IO,Ada.Float_Text_Io, Ada.Numerics
-- Mat. particulier :
--
-- Compilation      : gnat 3.14p
-- Edition de liens : gnat 3.14p
-- Mode d'execution : Console
-------------------------------------------------------------------------------

with Ada.Text_IO;           use Ada.Text_IO;
with Ada.Float_Text_IO;     use Ada.Float_Text_IO;
with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
with Ada.Numerics;          use Ada.Numerics;         -- pour la constante Pi

-- Calcul du prix d'une pizza
procedure Pizza is

   Coefficient : constant := 1.2;    -- Coefficient sans dimension intervenant dans
                                     -- le calcul du prix d'une pizza
   Fixe        : constant := 0.6;    -- Prix fixe [Euro]
   Base        : constant := 0.012;  -- Prix de base par unité de surface [Frs/cm2]
   Supplement  : constant := 0.0008; -- Prix par ingrédient et
                                     -- par unité de surface [Frs/cm2]

   Diametre       : Float;     -- Diamètre de la pizza choisi par l'utilisateur [cm]
   Nb_Ingredients : Integer;   -- Nombre d'ingrédients choisi par l'utilisateur
   Surface        : Float;     -- Surface de la pizza [cm2]
   Prix           : Float;     -- Prix de la pizza [Euro]
   Reponse        : Character; -- Réponse de l'utilisateur à la question:
                               -- "Voulez-vous quitter le programme? [O/N]"

begin -- Pizza

   -- Présentation du programme
   Put_Line("Ce programme calcule le prix [Euro] de votre pizza.");
   New_Line;
   Put("Ce prix dependra du diametre [cm]") ;
   Put_Line(" et du nombre d'ingredients que vous choisirez.");

   loop

      -- Saisies utilisateur
      New_Line;
      Put("Entrez le diametre de la pizza [cm]: ");
      Get(Diametre);
      Skip_Line;

      Put("Entrez le nombre d'ingredients voulu: ");
      Get(Nb_Ingredients);
      Skip_Line;

      -- Calcul de la surface (supposée circulaire) de la pizza [cm2].
      Surface := Pi * (Diametre / 2.0)**2;

      -- Calcul du prix (exact) de la pizza [Euro]
```

```
      Prix := Coefficient * (Fixe + Base * Surface +
                             Float(Nb_Ingredients) * Supplement * Surface);

      -- Affichage du prix (arrondi à l'entier le plus proche) de la pizza [Frs].
      Put("Le prix final de votre pizza est de ");
      Put(Integer(Prix), 0);
      Put_Line(" Euro ");

      -- L'utilisateur souhaite-t-il quitter le programme?
      New_Line;
      Put("Voulez-vous quitter le programme? [O/N] > ");
      Get(Reponse);
      Skip_Line;
      exit when Reponse = 'O' or Reponse = 'o';
      New_Line;

   end loop;

end Pizza;
```

Commentaires d'en-tête

- Utilisez l'en-tête type proposé dans le programme ci-dessus.
- Alignez les rubriques de l'en-tête comme dans le programme ci-dessus.
- Ne supprimer en aucun cas l'une ou l'autre de ces rubriques, même si celle-ci est vide.
- Dans la rubrique "But", décrivez PRECISEMENT ce que FAIT votre programme.
  N'importe quel lecteur de votre code doit être capable à la simple lecture de cette rubrique de comprendre précisément ce que fait votre programme.
- Dans la rubrique "Remarques", décrivez tous les compléments jugés nécessaires pour la bonne compréhension du programme (hypothèses faites, points volontairement non traités…).

Clause de contexte

- Vérifiez ABSOLUMENT que tous les paquetages déclarés dans la clause de contexte sont effectivement utilisés.
  SUPPRIMEZ les déclarations de ceux qui ne seraient pas utilisés.
- Ne commentez les paquetages déclarés dans la clause de contexte que lorsque cela amène vraiment quelque chose.
  Commenter Ada.Text_IO, par exemple, n'amène rien! On peut sans autre faire l'hypothèse que toute personne amenée à lire votre programme est censée savoir de quoi il s'agit…
- Utilisez de préférence (car facilite la lisibilité) l'une ou l'autre forme ci-dessous pour votre clause de contexte:

```
with Ada.Text_Io,
     Ada.Integer_Text_Io,
     Ada.Float_Text_Io;

use Ada.Text_Io,
    Ada.Integer_Text_Io,
    Ada.Float_Text_Io;
```

```
with Ada.Text_Io;          use Ada.Text_Io;
with Ada.Integer_Text_Io;  use Ada.Integer_Text_Io;
with Ada.Float_Text_Io;    use Ada.Float_Text_Io;
```

Présentation et mise en page

- Indentez votre programme
- Utilisez les conventions d'écriture Ada 95.
- Aérez votre programme au moyen de lignes blanches
- Alignez verticalement vos déclarations (cela facilite la lecture):

MAUVAIS :

```
Diametre : Float;          -- Diamètre de la pizza choisi par l'utilisateur [cm]
Nb_Ingredients : Integer;  -- Nombre d'ingrédients choisi par l'utilisateur
Surface : Float;           -- Surface de la pizza [cm2]
Prix : Float;              -- Prix (exact) de la pizza [Euro]
Reponse : Character;       -- Réponse de l'utilisateur à la question:
                           -- "Voulez-vous quitter le programme? [O/N]"
```

BON :

```
Diametre       : Float;     -- Diamètre de la pizza choisi par l'utilisateur [cm]
Nb_Ingredients : Integer;   -- Nombre d'ingrédients choisi par l'utilisateur
Surface        : Float;     -- Surface de la pizza [cm2]
Prix           : Float;     -- Prix (exact) de la pizza [Euro]
Reponse        : Character; -- Réponse de l'utilisateur à la question:
                            -- "Voulez-vous quitter le programme? [O/N]"
```

- Laissez toujours au moins un espace blanc entre les divers éléments

MAUVAIS :	**BON :**
`Supplement:constant:=0.0008;`	`Supplement : constant := 0.0008;`

- Ecrivez une seule instruction par ligne (cela facilite la lecture)

MAUVAIS :	**BON :**
`Put(Prix, 0); New_Line;`	`Put(Prix, 0);` `New_Line;`

Format des identificateurs et des mots reservés

Les mots réservés sont **toujours** en minuscules. Les identificateurs par contre doivent avoir la première lettre en majuscule. Si l'identificateur est composé de plusieurs mots, on les sépare avec un '_' et chaque mot commence par une majuscule.

Mot réservés : procedure begin end loop

Identificateurs : `Borne_Inf Est_Affiche Text_IO Enumeration_IO`
Note : un identificateur est le nom de quelque chose (constante ou variable, type, exception, procédure ou fonction, attribut, paquetage, …).

Choix des identificateurs

- Utilisez des identicateurs PARLANTS, que ce soit pour vos constantes, variables, …

MAUVAIS : Exemples de mauvais identificateurs: `Diam, Surf`

BON : Exemples de bons identificateurs: `Diametre, Surface`

- Lorsqu'il s'agit de quelque chose genre "nombre d'ingrédients":

MAUVAIS : `Ingredients, NbrIngedients, Nombre_D_Ingredients`

BON : `Nb_Ingredients ou Nbre_Ingredients`

Dans tous les programmes, modules et sous-programmes, et quel que soit le langage, chaque variable :

- doit posséder un non significatif
- est expliquée par un commentaire
- en général doit être la seule présente sur une ligne

L'explication se met si possible à côté de la déclaration

```
Borne_Inf : Iteger;    -- limite inférieure du traitement
Borne_Sup : Integer;   -- limite supérieure du domaine de définition
```

Déclaration des variables

- Prenez garde à supprimer de votre code toutes les déclarations de constantes et variables inutilisées!
- N'initialisez pas INUTILEMENT des variables comme dans le cas de l'extrait de code suivant:

```
Nbre_Ingredients := 0; -- TOTALEMENT INUTILE!!!
Put("Entrez le nombre d'ingredients > ");
Get(Nbre_Ingredients);
Skip_Line;
```

- Prenez garde à ne pas déclarer une foule de variables destinées uniquement à stocker certains résultats intermédiaires de calcul.

Commentaires

Les commentaires se mettent lors de l'écriture du code et non après
Un commentaire expliquant une partie de programme vient avant cette partie. Un commentaire qui n'apporte aucune information ne sert à rien.L'indentation

MAUVAIS	BON
-- Incrementer I de 1 I := I + 1;	*-- Passage à l'élément suivant du tableau* I := I + 1;
-- Si I plus grand que 0 alors **if** I > 0 **then** …	*-- Si l'indice existe* **if** I > 0 **then** …

```
-- Presentation d'un programme
Put_Line(" calcul de la surface  d'un cercle ");
-- Boucle d'affichage de l'alphabet minuscule
   for I in 'a'..'z'  loop
      Put(I);
      New_Line;
   end loop;
```

- Attention à l'ORTHOGRAPHE!!!
  Même le programme le mieux conçu laissera une impression de "bâclé" à la personne le lisant si ce dernier est "truffé" de fautes d'orthographe.
- Une première manière d'éviter les fautes d'orthographe consiste à RELIRE SOIGNEUSEMENT ses commentaires, une fois le programme terminé.
- Evitez ABSOLUMENT les commentaires inutiles (càd. n'apportant absolument rien de plus que le code pour la compréhension du programme)
- N'oubliez pas de toujours préciser (s'il y a lieu) dans vos commentaires dans quelle unité vous travaillez ([Kg], [Euro], …). C'est très important!!!
- Attention aussi à être précis dans ce que vous dites! Un commentaire imprécis est un commentaire inutile, voire dangereux!
- Attention à supprimer les commentaires qui pourraient ne plus être valables. Ici aussi cela peut facilement s'éviter si l'on relit soigneusement son code, une fois le travail achevé.

Choix des types

- Prenez garde de définir le type adéquat pour vos constantes et variables (c'est-à-dire le type permettant de traduire au plus près la vraie nature de la constante ou de la variable)

FAUX :	**JUSTE :**
`Nb_Ingredients : Float;` car que peut bien signifier 3.45 ingrédients ?	`Nb_Ingredients : Integer;`

Tester et encore tester...

- Une fois terminé, il s'agit de tester le programme. Tester signifie tester tous les cas particuliers plus quelques cas de fonctionnement normal.

Divers

- Incluez dans votre programme une "présentation du programme", à l'usage de l'utilisateur, dans laquelle vous décrirez brièvement ce que fait ledit programme.

MAUVAIS :
```
-- Présentation du programme
Put_Line("Bienvenue chez Speedy Pizza");
```

… car l'utilisateur de votre programme n'en retire aucune information pertinente!!!

BON :
```
-- Présentation du programme
Put_Line("Ce programme calcule le prix [Euro] de votre pizza.");
New_Line;
Put("Ce prix dependra du diametre [cm]") ;
Put_Line(" et du nombre d'ingredients que vous choisirez.");
```

- Dans vos dialogues avec l'utilisateur, mettez-vous à la place de ce dernier. La formulation de votre question est-elle suffisamment précise pour que l'utilisateur sache clairement quoi répondre?

MAUVAIS :
```
Put("Entrez le diametre souhaite pour votre pizza > ");
```

… car comment l'utilisateur peut-il deviner que vous attendez une réponse formulée en cm?

BON :
```
Put("Entrez le diametre souhaite pour votre pizza [cm] > ");
```

- Attention à formatter correctement les résultats (en particulier numériques) que vous affichez à l'écran.

- Prenez garde à ce que le nom du fichier ada soit IDENTIQUE à celui de votre programme principal (cohérence!!!).

Procédures et fonctions.

```
procedure Calculer_Somme(Somme       : in out Integer;
                         Nombre      : in Integer;
                         Limite      : in Integer;
                         Depassement : out Boolean) is …..
```

- Le nom d'une procédure est devrait être toujours à l'infinitif

- Il est raisonnable de donner le même nom à un programme et au fichier qui le contient

- Il est déconseillé de mettre des caractères accentués dans les commentaires