```
1   with Ada.Text_IO;
2   use Ada.Text_IO;
3
4   --  Affiche un message à l'écran.
5   procedure Hello is
6   begin
7      Put_Line ("Bonjour, de la part de Ada.");
8   end Hello;
```

Programme 1: hello.adb

Ligne 4 débute avec 2 tirets (signe moins), c'est un commentaire. Tout ce qui est après les tirets jusqu'à la fin de la ligne est ignoré par le compilateur.

En Ada, un programme principal peut être nommé de la façon dont vous voulez ou presque, par contre le nom du fichier doit correspondre à ce nom (en minuscule) et se terminer par ".adb". Dans notre exemple, le programme principal est appelé "Hello" (lignes 5 et 8) et le fichier est "hello.adb", adb signifiant "Ada Body."

Ligne 5 indique que notre programme est une procédure, ceci est 1 type de sous-programme en Ada, l'autre est la fonction. Ces 2 types sont utilisés pour des raisons spécifiques, une procédure ne retourne aucune valeur alors que la fonction en retourne toujours une. Les sous-programmes "main" sont des procédures.

Ligne 7 est un appel à un sous-programme du paquetage Ada.Test_IO, lignes 1 et 2. La procédure Put_Line affiche à l'écran tout ce qui est dans la chaîne de caractères (entre les guillemets ").

Ligne 1 indique que nous voulons utiliser les sous-programmes du paquetage Ada.Text_IO, et la ligne 2 indique au compilateur que nous ne voulons pas écrire leur nom complet, en d'autres mots, si la ligne 2 n'existait pas nous aurions dû taper **`Ada.Text_IO.Put_Line`**. Il y a des raisons pour ceci, nous les couvrirons une autre fois.

Étant donné que tout sous-programme doit avoir un début (ligne 6), il a aussi une fin, ligne 8. En Ada, tous les sous-programmes doivent spécifier ce qui est terminé en écrivant encore une fois le nom du sous-programme. Ada oblige ceci car c'est un outil pour rendre un programme plus lisible.

Tout programme Ada est composé d'un certain nombre d'instructions, chacune se terminant par un point-virgule (;). Toute instruction que vous tapez doit en avoir un ou le programme ne compilera pas.

En Ada, il y a des mots qui sont définis par le langage, ces mots sont appelés mots-clés, vous ne pouvez pas utiliser ces mots-clés pour définir vos propres types, variables ou sous-programmes.

compilateur Ada, comme vous le verrez au moment de compiler le programme, il appelle d'autres programmes, incluant gcc, gnatbind et gnatlink.

Vous pouvez maintenant exécuter le programme compilé avec la commande suivante :

```
$ ./hello
```

À l'exécution, le programme affiche sur le terminal ce qui est entre les guillemets dans le code, en d'autre mots, "Bonjour, de la part de Ada." Vous venez d'écrire votre premier programme en Ada !

Types simples et maths

Contrairement à d'autres langages, tel que C, Ada est un langage à typage fort. Qu'est-ce que le typage ? Bien, chaque valeur dans Ada a un type, par exemple, le nombre 10 est un nombre entier, alors si nous voulons mémoriser un nombre, nous définirons une variable de type entier, voir l'encadré pour plus d'informations sur les types. Quitter nano avec **`Ctrl-X`** et créez un nouveau fichier nommé simple_types.adb et tapez le code du programme 2.

Avec ce que vous avez appris de l'exemple précédent, tapez et sauvez ce code, compilez-le avec gnatmake et enfin exécutez le programme dans un terminal et regardez le résultat.