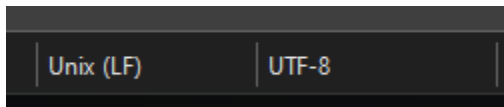


# Manuel Utilisateur - IRI

---

## WARNINGS :

- Il est primordial que la syntaxe du langage intermédiaire soit respectée (il est possible de voir plusieurs exemples de code intermédiaire (fichiers txt) dans le dossier tests et dans le fichier tri\_tableau.txt à la racine de l'interpréteur), l'interpréteur assume qu'aucune erreur est présente dans ce code.
- Les fichiers TXT contenant le langage intermédiaire doivent impérativement être sauvegardés en mode LF (Unix) et en encodage UTF-8 :



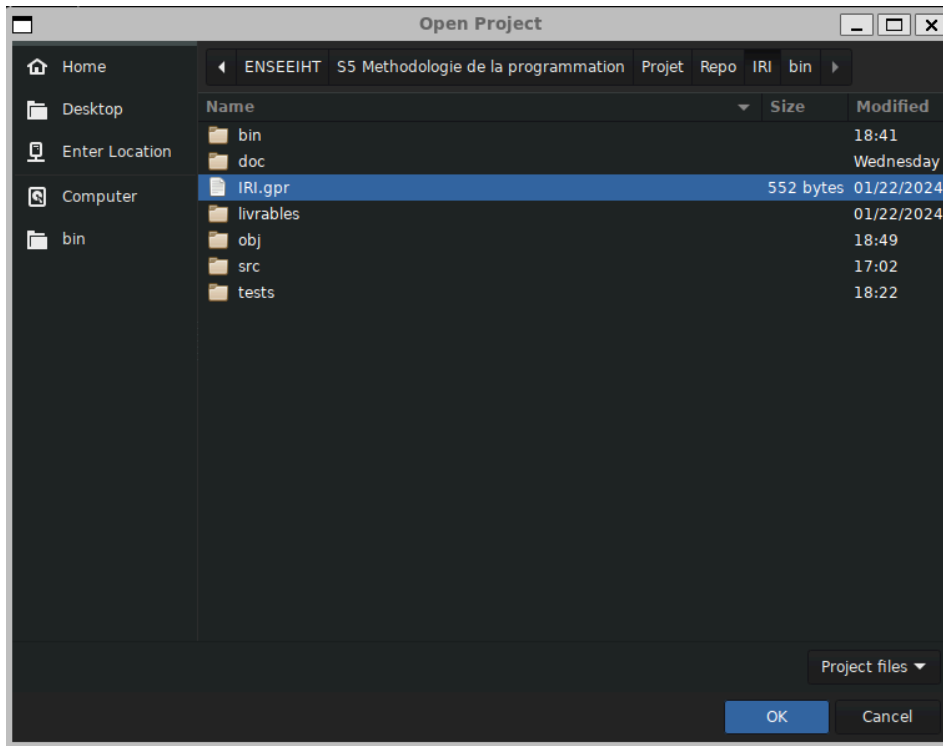
---

<b>Utilisation de l'interpréteur avec GnatStudio</b>	<b>1</b>
<b>Compilation et exécution de l'interpréteur à l'aide de gprbuild</b>	<b>4</b>
<b>Compilation et exécution des tests de l'interpréteur à l'aide de gprbuild</b>	<b>5</b>
<b>Compilation et exécution des tests de l'interpréteur à l'aide de GnatStudio</b>	<b>5</b>
<b>Comment écrire un programme en langage intermédiaire</b>	<b>7</b>

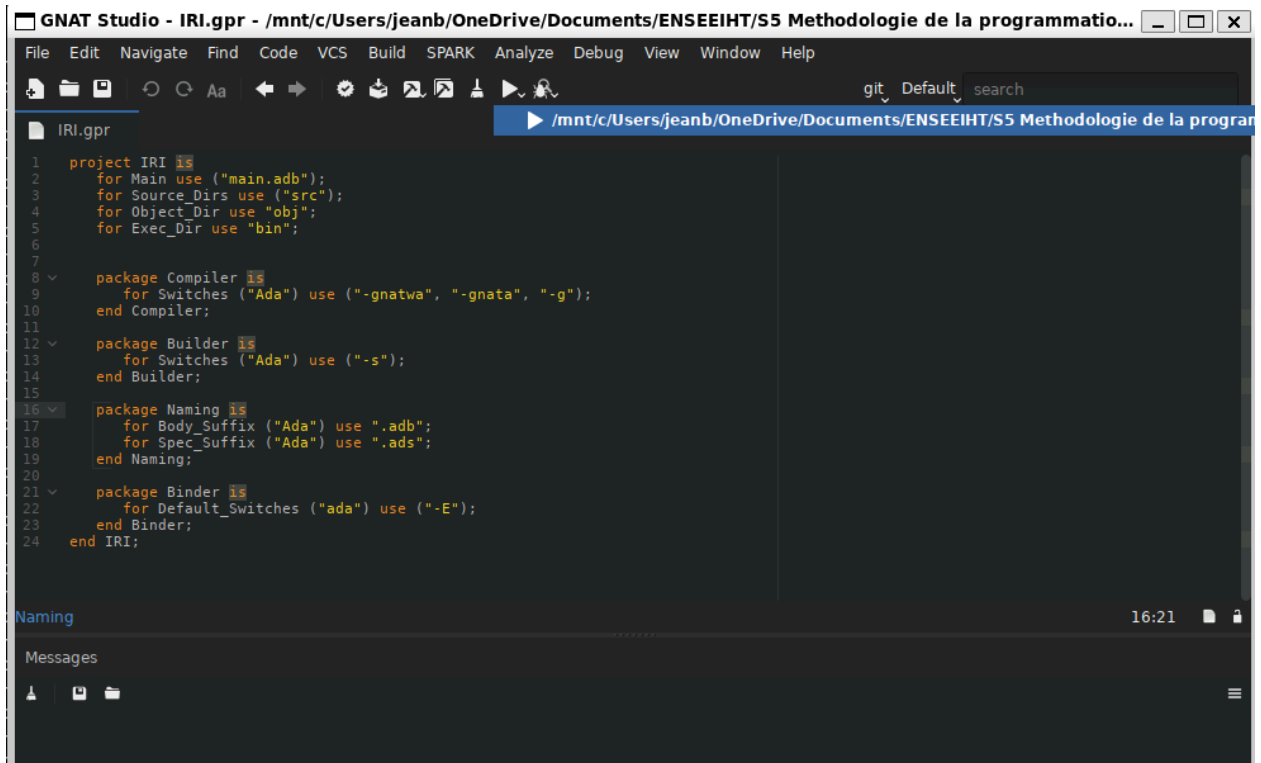
## Utilisation de l'interpréteur avec GnatStudio

Pour compiler et exécuter l'interpréteur avec Gnat Studio, il suffit de décompresser le fichier zip puis de naviguer dans le dossier créer.

Il faut ensuite ouvrir le fichier IRI.gpr avec GnatStudio :



Il suffit ensuite d'appuyer sur le bouton "Play" :




Ensuite, l'interpréteur vous demandera le chemin complet vers le fichier TXT que vous souhaitez interpréter :

```
24
25
26 -- Procédure pour lire le contenu du fichier
27 procedure Read_File_Content(Path : in String; Interpreteur : out T_Interpreteur) is
28   Handle : Reader.File_Handle; -- Handle du fichier
29   File_Content_String : Common_Types.String_List; -- Contenu du fichier sous forme de chaîne
30 begin
31   Reader.Open_File(Path => Path, Handle => Handle); -- Ouvre le fichier
32   File_Content_String := Reader.Get_Lines(Handle => Handle); -- Obtient les lignes du fichier
33   Reader.Close_File(Handle); -- Ferme le fichier
34   Lexer.Analyser_Lignes(File_Content_String, Interpreteur.Memoire); -- Analyse les lignes du fichier
35 end Read_File_Content;
36
37 -- Getter PC
```

Interpreteur.Init


Messages Locations Run: main

 /mnt/c/Users/jeanb/OneDrive/Documents/ENSEEIH/55 Methodologie de la programmation/Projet/Repo/IRI/bin/main  
Hello, World!  
Please input the full path to the file you want to evaluate  
|

Après avoir spécifié le chemin absolu vers le fichier txt, il suffit de choisir le mode d'exécution souhaité (normal ou debugger) :

Interpreteur.Init

Messages Locations Run: main

 /mnt/c/Users/jeanb/OneDrive/Documents/ENSEEIH/55 Methodologie de la programmation/Projet/Repo/IRI/bin/main  
Hello, World!  
Please input the full path to the file you want to evaluate  
/mnt/c/Users/jeanb/OneDrive/Documents/ENSEEIH/55 Methodologie de la programmation/Projet/Repo/IRI/tri\_tableau.txt  
Please enter the running mode (normal or debugger):  
|

En mode normal, le programme interprétera le fichier puis écrira le contenu du registre dans le terminal :

```

----- Register Content -----
-- Registry index value : 1 | Variable's name : L8 --- Value : 55 --- Type : T_LABEL
-----
-- Registry index value : 2 | Variable's name : L9 --- Value : 49 --- Type : T_LABEL
-----
-- Registry index value : 3 | Variable's name : L7 --- Value : 43 --- Type : T_LABEL
-----
-- Registry index value : 4 | Variable's name : L6 --- Value : 40 --- Type : T_LABEL
-----
-- Registry index value : 5 | Variable's name : L5 --- Value : 38 --- Type : T_LABEL
-----
-- Registry index value : 6 | Variable's name : L3 --- Value : 36 --- Type : T_LABEL
-----
-- Registry index value : 7 | Variable's name : L2 --- Value : 35 --- Type : T_LABEL
-----
-- Registry index value : 8 | Variable's name : L1 --- Value : 27 --- Type : T_LABEL
-----
-- Registry index value : 9 | Variable's name : I --- Value : 9 --- Type : T_ENTIER
-----
-- Registry index value : 10 | Variable's name : J --- Value : 2 --- Type : T_ENTIER
-----
-- Registry index value : 11 | Variable's name : Valeur --- Value : 8 --- Type : T_ENTIER
-----
-- Registry index value : 12 | Variable's name : Temp --- Value : 5 --- Type : T_ENTIER
-----

```

En mode debugger, le contenu de la mémoire est affiché une fois au début puis le contenu du registre et de IR sont affichés à chaque ligne qui est exécutée (il est possible de continuer l'exécution jusqu'à la fin sans interruption en entrant exec à la place d'appuyer sur entrée uniquement) :

```

----- Register Content -----
-- Registry index value : 1 | Variable's name : L8 --- Value : 55 --- Type : T_LABEL
-----
-- Registry index value : 2 | Variable's name : L9 --- Value : 49 --- Type : T_LABEL
-----

----- PC value : 3 -----
-- IR Tokens : Token 1 : LABEL --- Token 2 : L7 --- Token 3 : 43 --- Token 4 :
-----

Do you want to continue execution without breaks (write : exec) or execute only the next instruction (Press enter) ?

----- Register Content -----
-- Registry index value : 1 | Variable's name : L8 --- Value : 55 --- Type : T_LABEL
-----
-- Registry index value : 2 | Variable's name : L9 --- Value : 49 --- Type : T_LABEL
-----
-- Registry index value : 3 | Variable's name : L7 --- Value : 43 --- Type : T_LABEL
-----

----- PC value : 4 -----
-- IR Tokens : Token 1 : LABEL --- Token 2 : L6 --- Token 3 : 40 --- Token 4 :
-----

Do you want to continue execution without breaks (write : exec) or execute only the next instruction (Press enter) ?

```

## Compilation et exécution de l'interpréteur à l'aide de gprbuild

Pour compiler l'interpréteur sans passer par GnatStudio, il suffit de se mettre à la racine du dossier IRI décompresser auparavant et de taper la commande suivante (testée uniquement sous linux) :

```
gprbuild -P IRI.gpr src/main.adb -cargs -gnatf
```

Ensuite, il suffit d'exécuter l'interpréteur avec la commande suivante (testée uniquement sous linux) :

```
./bin/main
```

Vous pouvez vous référer au chapitre "Utilisation de l'interpréteur avec GnatStudio" pour en apprendre plus sur l'utilisation de l'interpréteur.

## Compilation et exécution des tests de l'interpréteur à l'aide de gprbuild

Pour compiler l'interpréteur sans passer par GnatStudio, il suffit de se mettre à la racine du dossier IRI décompresser auparavant et de taper la commande suivante (testée uniquement sous linux) :

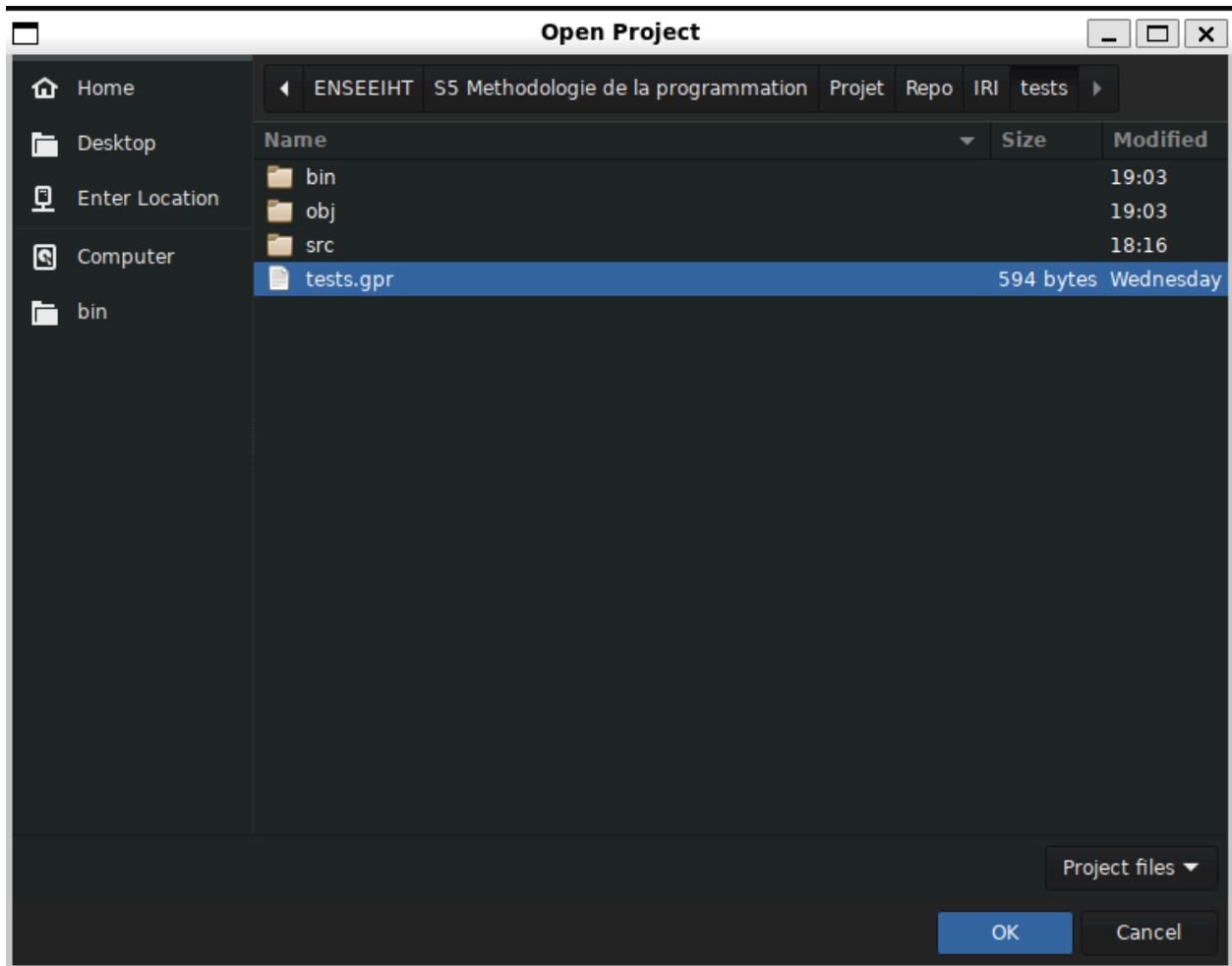
```
gprbuild -P tests/tests.gpr tests/src/main_tests.adb -cargs -gnatf
```

Ensuite, il suffit d'exécuter l'interpréteur avec la commande suivante (testée uniquement sous linux) :

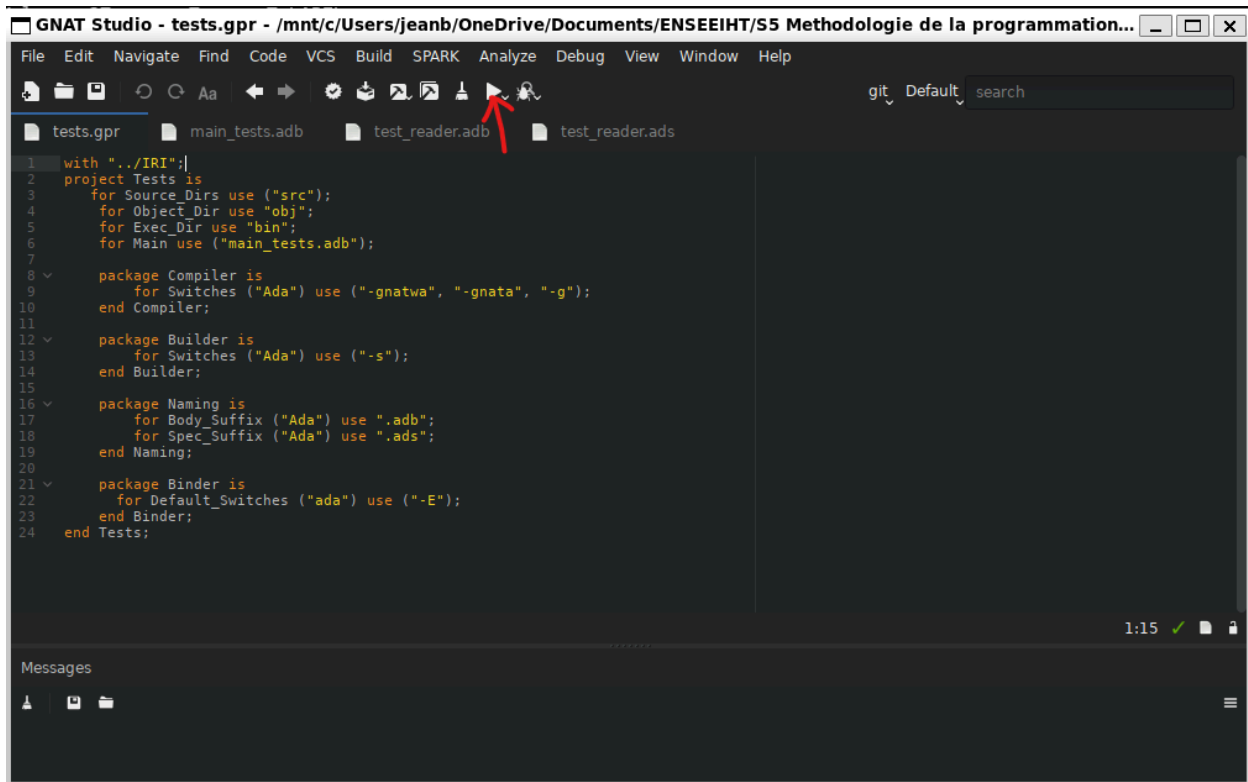
```
./tests/bin/main_tests
```

## Compilation et exécution des tests de l'interpréteur à l'aide de GnatStudio

Pour compiler et exécuter les tests de l'interpréteur avec Gnat Studio. Il suffit d'ouvrir le fichier tests.gpr dans le dossier tests avec GnatStudio :



Il suffit ensuite d'appuyer sur le bouton "Play" :



(Si vous avez des erreurs d'accès aux fichiers de tests, il suffit d'ouvrir un terminal à la racine du dossier IRI puis de taper cette commande (testée uniquement sous linux) :

```
./tests/bin/main_tests)
```

## Comment écrire un programme en langage intermédiaire

Pour écrire un programme en langage intermédiaire, suivez les règles suivantes :

1. **Déclaration du programme** : Commencez par déclarer le programme avec un nom significatif, suivi de la liste des variables utilisées dans le programme, précisant leur type. Les types disponibles sont les suivants (attention à bien respecter la syntaxe) :
  - Entier
  - Booleen
  - Caractere
  - String
  - Tableau (Capacite) DE Type

Utilisez le mot-clé "Programme" suivi du nom du programme et de la déclaration des variables.

2. **Instructions dans le bloc de début** : Les instructions du programme sont placées à l'intérieur du bloc "Debut" et "Fin". Assurez-vous de respecter la syntaxe du langage intermédiaire et d'indenter correctement les instructions.
3. **Affectation des valeurs** : Utilisez l'opérateur "<-" pour affecter des valeurs aux variables. Veillez à ce que les types des variables correspondent aux valeurs assignées.
4. **Gestion des structures de contrôle** : Utilisez des structures de contrôle telles que les boucles "Tant Que" et les branchements conditionnels "IF GOTO" pour contrôler le flux d'exécution du programme. Utilisez des étiquettes pour marquer les points de saut dans le programme. Les étiquettes doivent commencer par la lettre L majuscule et doivent être suivies d'un chiffre.
5. **Utilisation des fonctions et des procédures** : Vous pouvez utiliser des fonctions prédéfinies comme Lire(x) ou Ecrire(x).
6. **Commentaires** : Il est possible d'écrire des commentaires sur les lignes qui ne contiennent pas de code. Il n'est pas possible d'écrire des commentaires sur une même ligne à la suite de code

Exemple de code intermédiaire :

```
-- Code intermédiaire pour le tri à bulles
Programme TriBulle est
    I, J, Valeur, Temp, T1, T2, T3, T4, T5, T6, T7, T8, T9 : Entier
    Tab : Tableau (8) DE Entier
Debut
    -- Initialisation du tableau
    I <- 1
    Ecrire(Entrez_une_valeur_a_rajouter_au_tableau)
L1 Lire(Valeur)
    Tab(I) <- Valeur
    I <- I + 1
    T1 <- I < 8
    T2 <- I = 8
    T3 <- T1 OR T2
    IF T3 GOTO L1

    -- Tri à bulles
    I <- 1
L2 J <- 1
L3 T8 <- J + 1
    T9 <- Tab(J) > Tab(T8)

L5 IF T9 GOTO L6
    GOTO L7
```



```

L6  Temp <- Tab(J)
     Tab(J) <- Tab(T8)
     Tab(T8) <- Temp

L7  J <- J + 1
     T4 <- 8 - I
     T5 <- J < T4
     T6 <- J = T4
     T7 <- T5 OR T6
     IF T7 GOTO L3

L9  I <- I + 1
     T1 <- I < 8
     T2 <- I = 8
     T3 <- T1 OR T2
     IF T3 GOTO L2

     -- Affichage du tableau trié
     I <- 1
L8  Ecrire(Tab(I))
     I <- I + 1
     T1 <- I < 8
     T2 <- I = 8
     T3 <- T1 OR T2
     IF T3 GOTO L8

Fin

```