Présentation de l'application

« Clef INSEE »

Sommaire

Résumé

Ce document présente le développement initiatique d'une petite application graphique en Ada sous Ubuntu. Après avoir décrit le fonctionnement voulu pour l'application, il détaille la conception de l'interface, l'écriture des sources du projet, quelques choix de programmation qui sont intéressants à commenter et les principaux problèmes rencontrés à la compilation, avec la solution qui a été trouvée. Les sources complètes sont fournies (testées et compilables sous l'environnement de développement décrit). Développée par un débutant en Ada, cette application n'est pas parfaite mais elle peut constituer un « kit de démarrage ».

Présentation de l'application « Clef INSEE »

Le « NIR », numéro d'inscription au registre des personnes physiques, plus connu en France sous le nom de « numéro de sécurité sociale », est géré par l'INSEE. Il est constitué de 13 chiffres décrivant la date et le lieu de naissance de chaque individu, auxquels s'ajoute une clef de contrôle à deux chiffres. Calculée à partir des 13 premiers chiffres, cette clef permet de vérifier l'absence d'erreur dans le numéro.

Il y a certaines conventions pour certains pays ou certaines régions de naissance mais, en général, le numéro de sécurité sociale a la structure et la signification suivantes :

X	XX	XX	XX	XXX	XXX		XX
Genre	An	Mois	Département	Commune	N° d'ordre		Clef

La clef est facile à calculer puisqu'elle est égale au complément à 97 du reste de la division du NIR par 97.

La formule de calcul pourrait presque tenir en une ligne de programme. Plus sérieusement, un programme en Ada travaillant à la console ne demande qu'une vingtaine de lignes de code (annexe 1). Ce programme montré en annexe 1 n'est pas un exemple à suivre pour une application réelle parce qu'il ne contrôle pas les données saisies par l'utilisateur et ne fait aucune gestion d'erreur. De ce fait, il est vulnérable et peu robuste. Néanmoins, il est parfaitement fonctionnel. Après compilation, il fournit un exécutable de 25 ko environ dont voici un exemple d'exécution :

```
$ ./clef_insee
Donner un identifiant INSEE sans sa clef (13 chiffres) :
1111111111111
Sa clef est = 20
```

Le développement d'une application graphique pour calculer cette clef de contrôle est pris comme prétexte pour s'initier à Ada dans des conditions simples mais néanmoins représentatives d'une application réelle fonctionnant dans un environnement fenêtré moderne. En effet, le logiciel présenté dans ce document met en œuvre les principaux mécanismes à la base de la programmation graphique événementielle « orientée objet » :

- construction d'une fenêtre d'interface pour l'application avec des composants pour les entrées-sorties et les actions à accomplir ;
- Importation de cette interface dans un programme sous la forme d'objets ;
- Association à l'interface de signaux émis lorsqu'un composant est utilisé ;
- Traitement de ces signaux par des gestionnaires de signaux, qui sont des sous-programmes exécutant les actions désirées ;
- Affichage des résultats de ces actions dans les composants idoines de l'interface ;
- Fermeture de la fenêtre le moment venu.

L'exemple détaillé ici est cependant seulement une introduction : d'une part, il n'entre pas dans des notions avancées du langage Ada et, d'autre part, quant à l'interface graphique, il n'utilise pas de menus, ni de gestion de fichiers, ni de composants pour effectuer une représentation graphique…

Description de l'application (cahier des charges)

Aspect de la fenêtre

La fenêtre de l'application comporte :

- la barre de titre avec le nom de l'application,
- une étiquette avec un texte souligné de consigne pour l'utilisateur,
- six champs pour saisir les six données du NIR,
- un unique bouton pour lancer le calcul de la clef,

- un champ pour afficher le résultat du calcul, surmonté par
- une étiquette indiquant son contenu,
- un panneau d'affichage destiné à donner des informations à l'utilisateur en cours d'exécution.

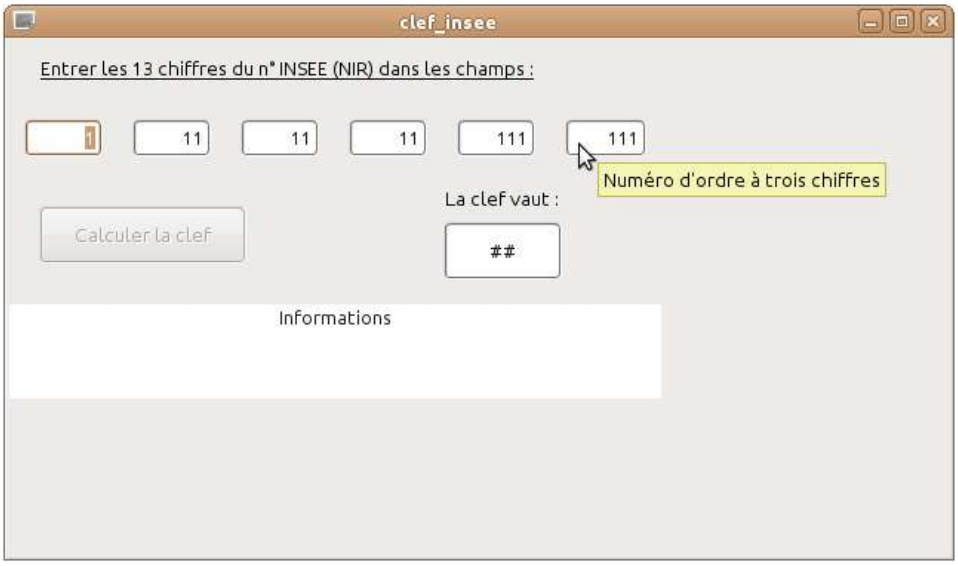
Comportement de la fenêtre

0. Le nom donné au programme est clef_insee. Il se lance dans un terminal ouvert dans son dossier par la commande ./clef_insee et, pendant l'exécution, ce terminal reçoit des informations sur le déroulement du programme et les éventuels messages d'erreur.

1. A l'ouverture de la fenêtre, les champs de saisie sont pré-remplis avec des chiffres « 1 » indiquant le format de saisie. Chaque champ est paramétré pour accepter au plus le nombre de caractères normalisé pour lui. Le bouton est inactif ; la valeur de clef est absente, son emplacement est garni de ## et ne peut pas être modifié. La seule action que l'utilisateur peut accomplir est donc la saisie d'un NIR ; il doit pour cela modifier au moins un chiffre dans l'un quelconque des champs de saisie.

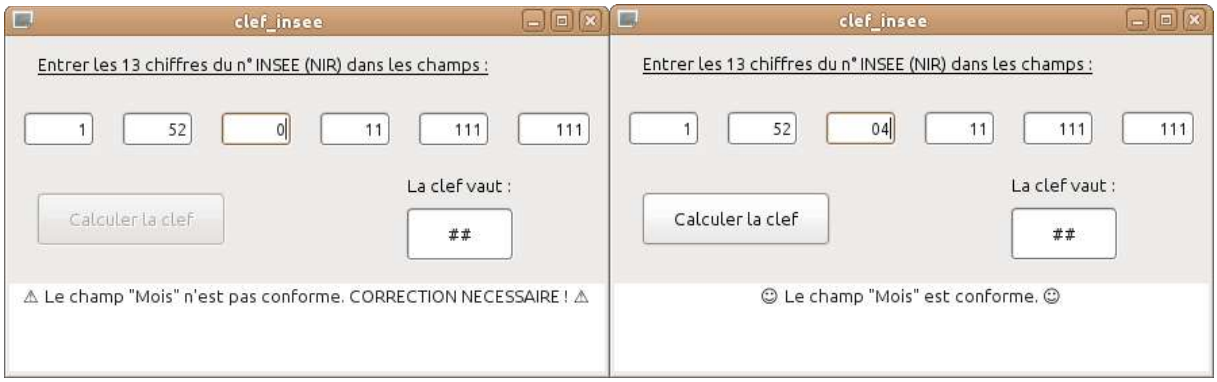
2. L'état décrit au §1 reste vrai si la fenêtre est déplacée ou redimensionnée.

3. Une bulle d'aide apparaît au survol des champs de saisie pour rappeler leurs caractéristiques.



4. Dès qu'au moins un chiffre a été saisi dans l'un des champs, le bouton devient actif et l'utilisateur peut à tout moment cliquer dessus pour lancer le calcul de la clef.

5. En cours de saisie d'un champ, le panneau d'affichage inférieur indique si l'état présent du champ est conforme au format attendu ou bien si une correction est attendue. C'est par exemple le cas si un seul chiffre est saisi dans un champ qui en attend deux ou bien si une lettre est saisie à la place d'un chiffre. Tant qu'un champ n'est pas conforme, le calcul n'est pas possible : soit le bouton de calcul est inactif, soit il le devient au moment du clic sans lancer le calcul.



6. Lorsque tous les champs sont remplis selon le souhait de l'utilisateur, un clic sur le bouton lance le calcul, l'affichage du résultat dans le champ prévu à cet effet et l'affichage d'un message de bonne exécution dans le panneau d'affichage inférieur.

7. Il est alors possible de faire un deuxième calcul sans relancer l'application. Dès que l'un des champs est modifié, la valeur de clef est remplacée par ## et le calcul est impossible tant que la saisie n'est pas valide.

8. Lorsque le calcul est terminé, il n'y a plus qu'à fermer la fenêtre en cliquant sur la croix de la barre de titre ou bien en tapant [Alt][F4] au clavier.

Développement

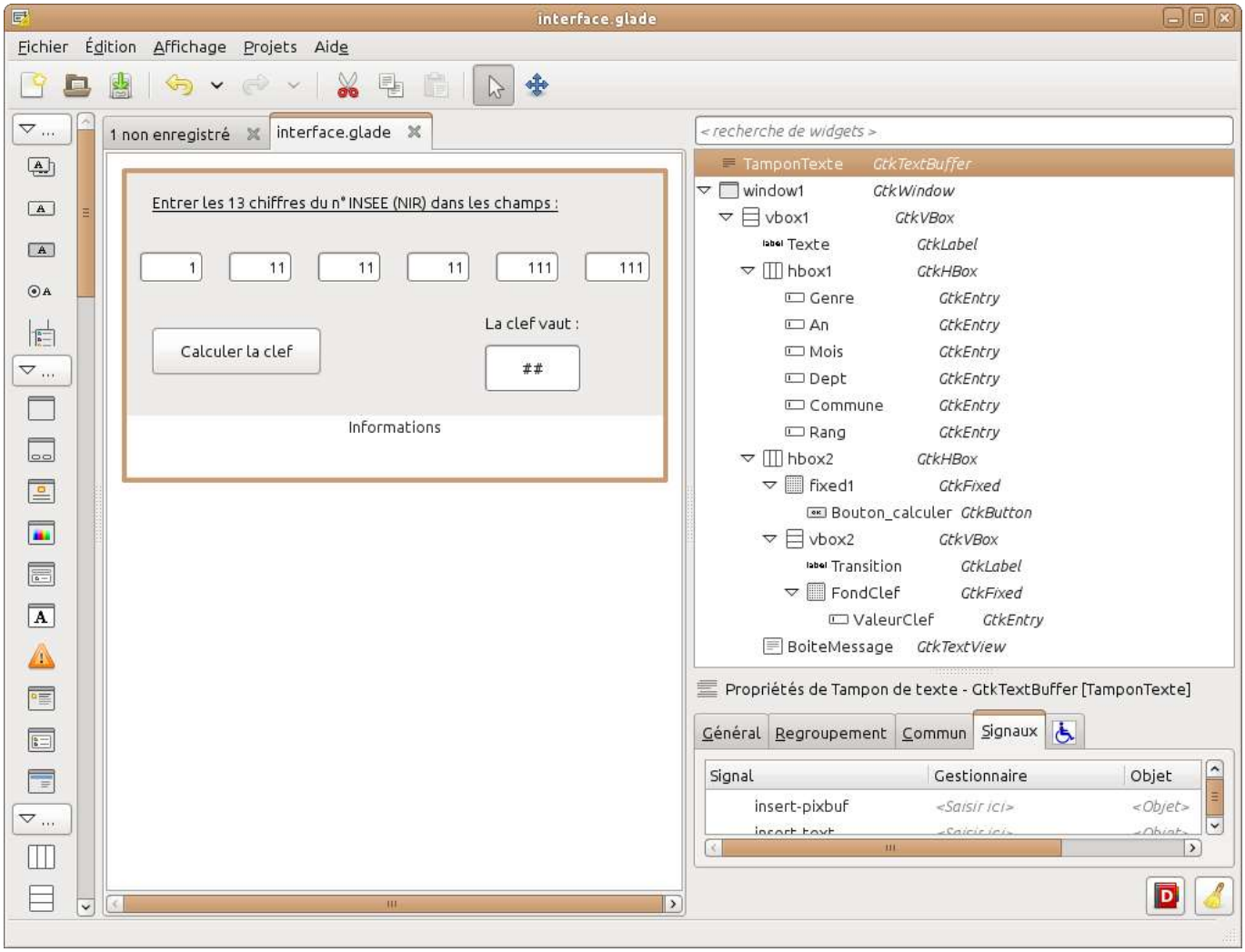
Configuration

Il est important de porter attention à la configuration de développement car la compatibilité ascendante des logiciels n'est pas toujours assurée lors des évolutions des systèmes. L'implémentation présentée ici est développée dans la configuration suivante.

- GTK 2 et 3 sont présents sur le système, en versions respectives 2.24.27-0ubuntu1~trusty1 et 3.10.8-0ubuntu1.4.
- Glade 3.16.1 est présent et capable d'utiliser GTK 3.10 mais, pour cette application, on a utilisé le module limité à GTK2 (glade-gtk2 version 3.8.0) réglé pour utiliser GTK 2.24 avec le format gtkbuilder.
- Le compilateur est Gnat et, bien que la version 4.8 soit disponible, c'est ici la version 4.6 qui a dû être employée.
- La liaison avec Ada, gtkada, a été installée par Synaptic en version 2.24.1-14.
- L'EDI *Gnat Programming studio* est en version GPS 5.0-16 (pour la famille Debian).

Conception de l'interface graphique

La fenêtre est assemblée dans Glade avec l'arborescence visible dans l'illustration.



A la racine du projet, il y a deux composants : le tampon de Texte *TamponTexte* (invisible mais qui sert à alimenter le panneau d'affichage inférieur) et la fenêtre proprement dite *window1*. Celle-ci est entièrement remplie par une boîte verticale à quatre tiroirs (*vbox1*) qui contient, de haut en bas :

- 1. l'étiquette de consigne (*Texte*) qui, statique, se met en forme dans Glade,
- 2. une boîte horizontale à six tiroirs (*hbox1*) contenant chacun un champ de saisie,
- 3. une boîte horizontale à deux tiroirs (*hbox2*) contenant, à gauche, une grille (*fixed*) supportant le bouton (*Bouton_calculer*) et, à droite, une boîte verticale à deux tiroirs (*vbox2*) ; cette dernière présente, en haut, une étiquette passive (*Transition*) avec le texte « La clef vaut : » et, en bas, une grille (*FondClef*) portant le champ destiné à recevoir la valeur de la clef (*ValeurClef*).
- 4. un composant de vue de texte (*BoiteMessage*) qui affiche le contenu du tampon de texte *TamponTexte*.

Les composants actifs émettent des messages. L'application utilise les messages suivants.

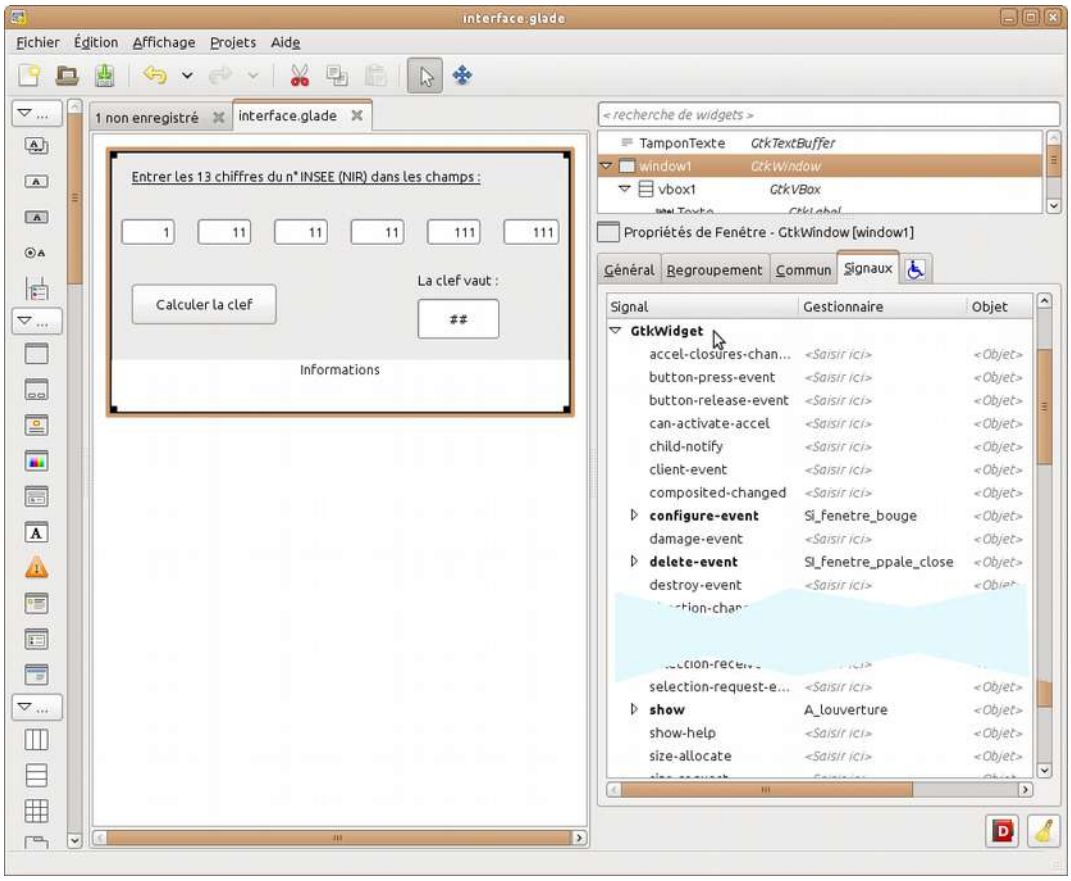
- Pour la fenêtre *window1*, trois signaux de l'ascendant GtkWidget sont exploités :

- *delete-event* qui est émis lorsque l'utilisateur ferme la fenêtre ; on lui connectera le gestionnaire *Si_fenetre_ppale_close* pour quitter le programme ;

- *show* qui est émis à l'affichage de la fenêtre, ce qui se produit dans notre cas une seule fois, au lancement de l'application ; il sera connecté au gestionnaire *A_louverture* qui a pour rôle a) de mettre le bouton de calcul à disposition de tous les gestionnaires de signaux, b) de mettre ce bouton dans l'état initial inactif, c) d'initialiser à « false » la variable *Edition_en_cours* qui permet de savoir si l'utilisateur est en train de saisir un NIR ou non ;

- *configure-event*, émis lorsque la fenêtre est déplacée ou redimensionnée ou lorsqu'elle change de profondeur ; il sera connecté au gestionnaire de signal *Si_fenetre_bouge* qui désactive le bouton lors d'un

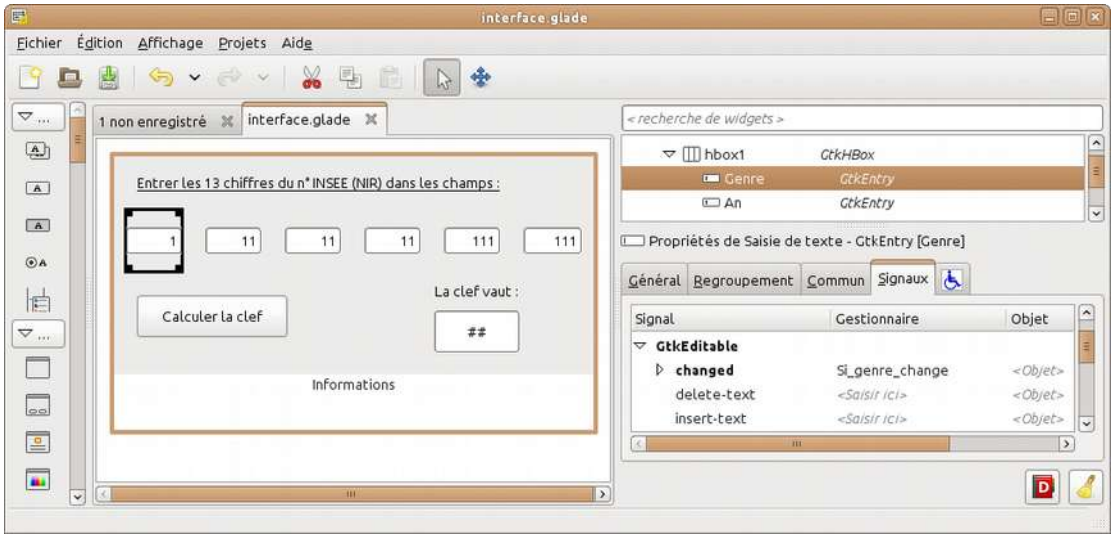
mouvement de fenêtre si toutefois l'utilisateur n'est pas en train de saisir un NIR ; ceci assure que le bouton ne devienne actif alors que l'utilisateur n'a pas encore commencé à saisir sa donnée.



Il n'est pas obligatoire de noter le nom du gestionnaire sur la ligne du signal (car la connexion est manuellement programmée[1] dans le programme principal en Ada) mais le faire améliore grandement la lisibilité et facilite le travail du programmeur.

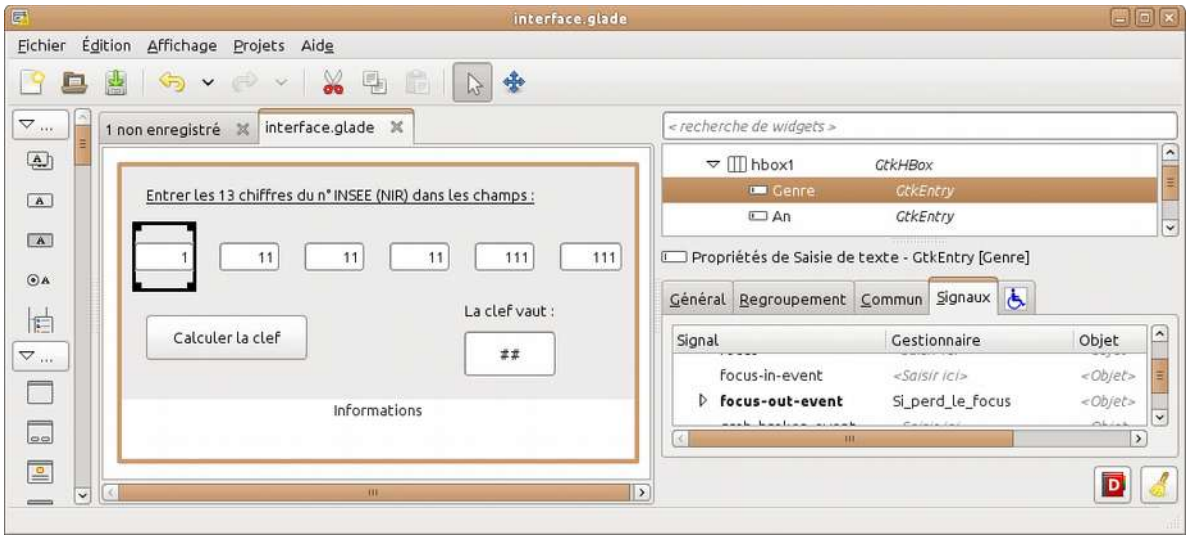
➢ Pour chaque champ de saisie « xxx », deux signaux sont employés

- *changed* de la rubrique GtkEditable, émis lorsque l'utilisateur apporte un changement au contenu du champ ; le gestionnaire connecté, *Si_xxx_change*, est propre à chaque champ de saisie puisqu'il a pour but principal de valider le contenu du champ par rapport au format attendu. Accessoirement, il met le bouton de calcul à disposition de tous les gestionnaires (pour le cas où cela n'aurait pas été fait avant), il passe la variable *Edition_en_cours* à « true » et il remet « ## » dans le champ de la valeur de clef si son contenu a été changé par un calcul antérieur.

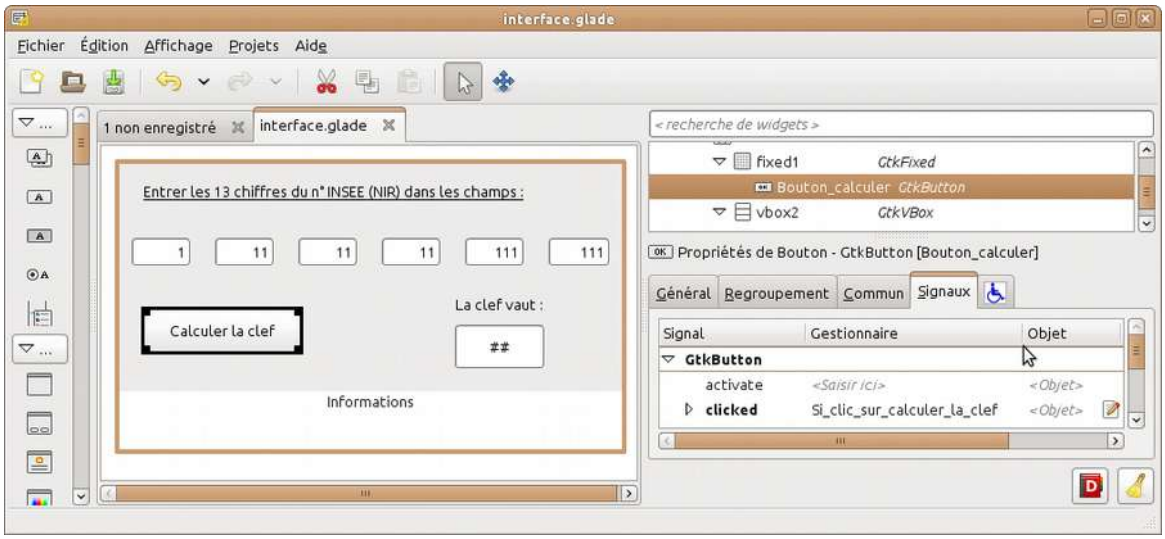


[1] Ce n'est pas toujours le cas ; voir un cas où l'on effectue une connexion automatique dans la page « Premier pas en programmation d'applications graphiques en python avec glade (sous GNU/Linux) » à l'adresse http://gja.frndz.pagesperso-orange.fr/technik/gladpyth/gladpyth.htm).

- *focus-out-event* de la rubrique GtkWidget, émis lorsque l'utilisateur quitte le champ de saisie ; le gestionnaire connecté est *Si_perd_le_focus*, le même pour tous les champs de saisie, et vérifie si l'ensemble du NIR est valide et, dans ce cas, active le bouton de calcul sinon, il le désactive.



➢ Enfin, pour le bouton de calcul, seul le signal *clicked* est exploité ; son gestionnaire, *Si_clic_sur_calculer_la_clef*, a pour rôle de calculer la clef et de l'afficher dans le champ *ad hoc*.



Chaque gestionnaire de signal appelé doit renseigner le panneau d'affichage et décrire à la console le déroulement du programme.

Structure du projet Ada

La structure du projet est conçue pour servir de modèle, c'est-à-dire pour permettre de réaliser une autre application simple à une seule fenêtre en ne modifiant que les parties clairement identifiées comme « spécifiques à l'application » mais en conservant tous les éléments génériques.

Dans la structure très simple adoptée ici, la disposition sur le disque est l'arborescence suivante :

> **Clef_INSEE_GUI**	(dossier du projet)
clef_insee_gui.gpr	(fichier décrivant le projet)
interface.glade	(fichier produit par Glade, description XML de la fenêtre)
> **src**	(dossier contenant les sources en langage Ada)
clef_insee.adb	(fichier source du programme principal)
actions_specifiques.ads	(fichier source spécifiant les sous-programmes spécifiques)
actions_specifiques.adb	(fichier source implémentant les sous-prog. spécifiques)
> **obj**	(dossier initialement vide, reçoit l'exécutable produit par la compilation et d'autres fichiers intermédiaires).

Pour ne pas avoir de mauvaises surprises, il faut écrire les noms des fichiers en minuscules et mettre le fichier interface.glade dans le même dossier que le fichier décrivant le projet.

Le fichier clef_insee.adb contient le programme principal. Par choix de conception modulaire, sa partie spécifique est essentiellement limitée à la connexion des signaux spécifiques aux gestionnaires de signaux correspondants.

Les fichiers actions_specifiques.ad* contiennent les spécifications et les implémentations vraiment particulières à l'application considérée.

Cette structure est définie par le fichier décrivant le projet, clef_insee.gpr, dont le contenu est clair, compte tenu des explications qui précèdent.

```
with "gtkada";

project clef_insee_gui is

  for Main use ("clef_insee.adb");
  for Source_Dirs use ("src");
  for Object_Dir use "obj";
  for Source_Files use ("actions_specifiques.adb", "actions_specifiques.ads",
"clef_insee.adb");
   for Languages use ("Ada");

end clef_insee_gui ;
```

Il importe de noter la clause **with "gtkada" ;** qui met à disposition du projet le paquet gtkada de portage de gtk sous ada. Ceci implique que l'exécutable produit par la compilation a ce paquet pour dépendance. Pour transporter l'application sur un autre système, il est plus sûr d'y d'installer gtkada 2 et d'y recompiler les sources.

L'annexe 2 donne les contenus des fichiers du projet.

Quelques remarques sur la programmation

Contrôle des données saisies par l'utilisateur

A titre d'illustration, le programme utilise deux façons de contrôler les données entrées par l'utilisateur dans les champs de saisie.

La **première méthode** consiste à vérifier directement, « manuellement », ce qui est écrit dans les champs. Par exemple, pour le champ Mois, on contrôle que la longueur de la chaîne saisie est 2, que le premier caractère est un chiffre '0' ou '1', que dans le premier cas le second caractère est un chiffre entre '0' et '9' mais que dans le second cas c'est un chiffre entre '0' et '2'. Ceci est programmé dans *Si_mois_change* sous la forme suivante.

```
if Object.Get_Text'Length < 2   -- Object est le champ "Mois"
then
     Mois_OK:= false ;
elsif (Object.Get_Text(1)='0' and Object.Get_Text(2) in '1'..'9')
          or (Object.Get_Text(1)='1' and Object.Get_Text(2) in '0'..'2')
then
     Mois_OK:= true ;
else
     Mois_OK:= false ;
end if ;
```

Cela marche mais c'est lourd et fastidieux et deviendrait impraticable pour des saisies plus longues.

La **deuxième méthode** consiste à se reposer sur les capacités de typage du langage. Il faut alors définir un (sous-)type caractérisant la donnée à vérifier, faire une tentative de conversion de la chaîne saisie dans une variable du type en question et mettre en œuvre la gestion d'erreur sur cette conversion. Par exemple, avec la déclaration préalable :

```
subtype CommRang is Positive range 1..999 ;
```

le gestionnaire Si_commune_change fait le contrôle comme suit :

```
procedure Si_commune_change
     (Object : access Gtk.Gentry.Gtk_Entry_Record'Class;
      BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
   function Is_of_subtype_CommRang (Chaux : in String) return Boolean is
      Auxiliaire : CommRang ;
   begin
      Auxiliaire := CommRang'Value (Chaux) ; — Tentative de conversion
      return true ;  —- Si la conversion réussit, la saisie est valide
   exception           —- Si elle provoque une erreur, la saisie est non
valide
      when others => return false;
   end Is_of_subtype_CommRang ;

begin
   […]
   if Object.Get_Text'Length < 3     -- Object est le champ "Commune"
   then
      Commune_OK:= false ;
   else
      Commune_OK:= Is_of_subtype_CommRang (Object.Get_Text) ;
   end if ;
   […]
```

C'est plus lisible, plus élégant et plus conforme à l'esprit du langage. Il est possible d'obtenir le même effet avec un code plus compact, au détriment de la lisibilité.

Pour mémoire : a) en Ada, il est illégal de surcharger l'opération « := » ; b) l'affectation avec conversion comme effectuée ci-dessus n'est évidemment pas légale entre tous les types.

Connexion des gestionnaires de signaux

Pour le débutant, la connexion des signaux est la tache de programmation la plus difficile à faire aboutir. Le compilateur déclare souvent une erreur de type qui ne renseigne pas sur la cause réelle du problème et cette dernière peut être variée. Il est donc utile d'essayer de comprendre le mécanisme.

La connexion est effectuée par des « services de connexion » fournis par le paquet *gtk.Handlers*. *Handler* signifie gestionnaire de signal ; on trouve aussi le mot *Callback* pour désigner le sous-programme qui effectue la gestion du signal. Ici on emploie « gestionnaire de signal » pour ces deux mots. La documentation du paquet donne des éléments utiles à la compréhension, quoique pas toujours finement rédigés.

Définitions

Signal : sorte de message envoyé par un objet (composant de l'interface) lors d'un événement.

Gestionnaire de signal : sous-programme (*procedure* ou *function*) que le programmeur « connecte », c'est-à-dire associe, à un signal d'un objet particulier, pour effectuer une action. Lorsqu'un signal est émis, tous les gestionnaires de signaux qui lui ont été connectés sont appelés, dans l'ordre où ils ont été connectés.

Remarque : certains signaux exigent un gestionnaire fonction (*focus-out-event* par exemple).

Description

Dans GtkAda, un gestionnaire de signal est défini de la manière la plus générale possible. Son **premier argument est toujours un pointeur (*access*) vers l'objet (émetteur du signal)** auquel il est connecté. Le **second objet est un tableau**[2] (d'indice commençant à 1) **de valeurs (pointeurs) qui**

[2] C'est ce que dit la documentation mais l'expérience montre qu'il est possible de passer directement une seule valeur (exemple : BTN: *Gtk.Button.Gtk_Button*) au lieu d'un tableau.

doivent être extraites et converties au bon type Ada par le gestionnaire de signal. Ces valeurs peuvent correspondre à des types différents.

Les gestionnaires de signaux n'étant pas toujours très faciles à utiliser[3], le paquet Gtk.Handlers fournit aussi quelques services pour connecter à leurs places des médiateurs (*marshaller*) qui se chargent du travail d'extraction[4] avant l'appel du gestionnaire de signal. Un tel médiateur n'est pas toujours obligatoire mais son insertion ne nuit pas. Pour le programmeur, leur emploi se résume à faire appel au paquet de connecteurs instancié pour remplacer le pointeur d'un gestionnaire de service *gestionnaire_de_service'Access* par l'expression :

```
paquet_de_connecteurs.To_marshaller(gestionnaire_de_signal'Access)
```

La fonction *To_marshaller* prend le gestionnaire de signal en entrée et renvoie un médiateur (*marshaller*) qui peut être connecté au signal.

Il y a quatre sortes de gestionnaires de signaux, selon qu'ils renvoient une valeur ou non et selon qu'une donnée spécifique à l'utilisateur leur est associée ou non. A chaque sorte de gestionnaire de signal correspond un paquet fournissant des connecteurs – on appelle ici « connecteur » un sous-programme établissant le lien entre le couple [objet émetteur-signal] et le gestionnaire de signal. Il est possible de disposer d'un champ prévu pour une donnée utilisateur mais son utilisation est optionnelle.

Paquet de connecteurs suivant que le gestionnaire de signaux…	renvoie une valeur (*function*)	ne renvoie pas de valeur (*procedure*)
n'a pas besoin d'une donnée utilisateur	*Return_Callback*	*Callback*
attend une donnée utilisateur	*User_Return_Callback*	*User_Callback*

Par exemple, le paquet fournissant le connecteur pour le gestionnaire du signal *delete-event* émis par un objet *Gtk_Window* prévoit le renvoi d'une valeur et a un paramètre supplémentaire (de type *Gint*) . Pour connecter ce signal, il y a deux cas :

- s'il n'y a pas de donnée utilisateur à transmettre au gestionnaire de signal, alors le paquet de connexion à instancier est le *Return_Callback* ; par exemple :

```
Gtk.Handlers.Return_Callback (Gtk.Window.Gtk_Window_Record, Boolean)
```

- si, au contraire, le champ de donnée utilisateur est nécessaire, c'est le *User_Return_Callback* qu'il faut employer.

```
Gtk.Handlers.User_Return_Callback
          (Gtk.Window.Gtk_Window_Record, Boolean, User_type)
```

Dans les cas où aucune valeur n'est renvoyée, les paquets à instancier sont respectivement *Gtk.handlers.Callback* et *Gtk.Handlers.User_Callback*.

Le paramètre générique commun à ces 4 paquets est le type du composant (*widget*) émetteur.

Ces quatre paquets sont pareillement organisés. Chacun fournit plusieurs sortes de connecteurs.

- *.Connect()*

Il y a plusieurs versions disponibles du connecteur *Connect()* qui se distinguent

- par leurs natures (*function* renvoyant l'identifiant *Handler_id* du lien créé ou *procedure* en faisant abstraction),

- par le fait qu'elles attendent la connexion directe du gestionnaire de signal ou bien sa connexion par l'intermédiaire d'un médiateur (*marshaller*).

- *.Object_Connect*()

Ce connecteur ne propose pas la possibilité de transmettre une donnée utilisateur. Au lieu de cela, il a un paramètre supplémentaire appelé *Slot_Object*. Lors de l'appel du gestionnaire de signal, l'objet

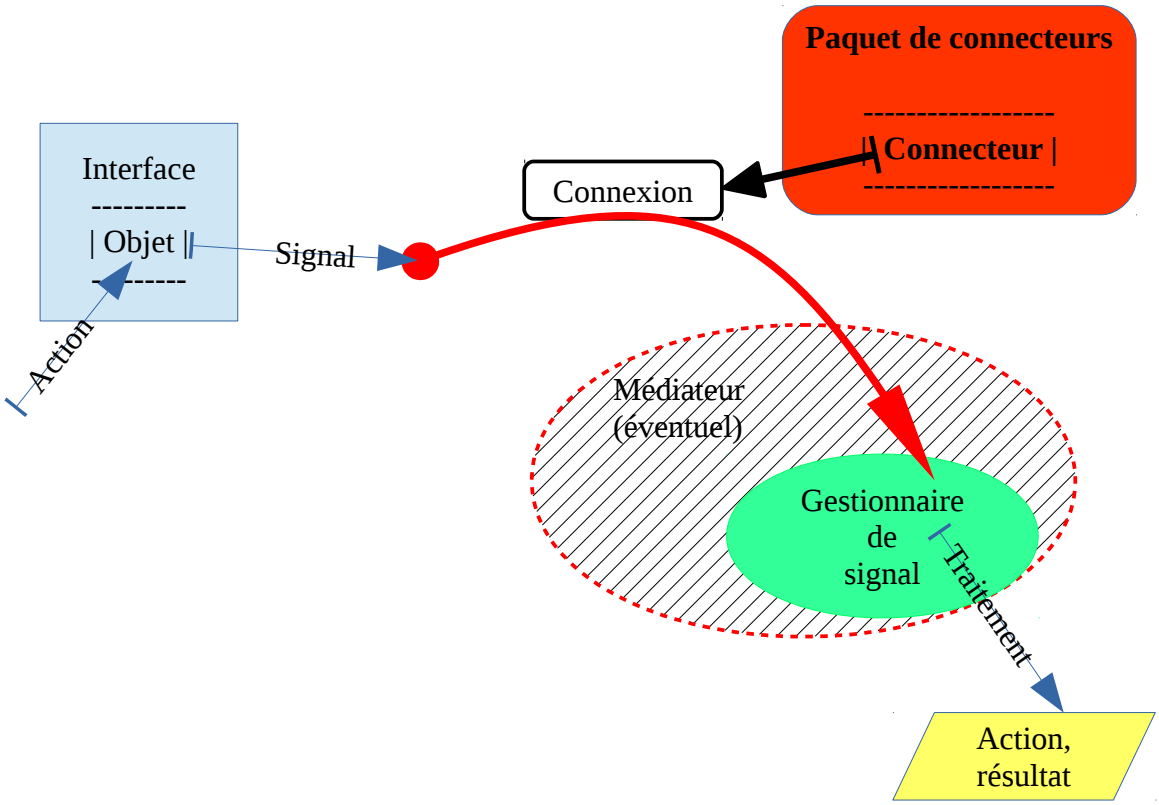
[3] On suppose que la documentation veut dire « pas faciles à utiliser par le compilateur ».

[4] extraction, sans doute, du signal et des objets à transmettre au gestionnaire.

émetteur est remplacé par l'objet[5] transmis par ce paramètre[6]. Ces gestionnaires de signaux sont toujours automatiquement déconnectés dès que l'un des deux objets concernés est détruit.

Ce connecteur existe également en plusieurs versions, comme *Connect*().

Synthèse graphique du dispositif



Les problèmes du compilateur

Ce dispositif oblige le compilateur à reconnaître la version du connecteur employé. Il peut certes le faire sans ambiguïté d'après le nombre et le type des arguments inscrits par le programmeur, à condition que ceux-ci soient exactement conformes à ceux attendus par l'une des versions du connecteur requis.

Si tel n'est pas le cas, le compilateur déclare l'erreur :

```
no candidate interpretations match the actuals:
```

disant par là qu'il n'arrive à faire correspondre aucune des versions du connecteur concerné avec ce qui est écrit par le programmeur. Ensuite il donne la raison qu'il identifie, qui est le plus souvent :

```
expected type "Handler" […]
found type access to function "Ici_le_nom_du_gestionnaire" […]
```

Autrement dit, le compilateur incrimine une erreur de type : il se plaint de recevoir l'accès au gestionnaire de signal au lieu du gestionnaire de signal qu'il attend…

Outre sa formulation un peu ésotérique, le message d'erreur n'explique pas le pourquoi de cette « erreur » de typage. Le problème est que des raisons diverses peuvent causer ce comportement. Par exemple, un nombre erroné d'arguments peut empêcher le compilateur de reconnaître le connecteur ; ou bien une clause *with* absente dans un autre fichier peut rendre le gestionnaire de signal indisponible ; ou bien, une erreur de type sur un autre argument peut induire le compilateur en erreur ; …

---

5 Cependant, la comparaison de la documentation de *gtk-handlers* et du manuel de référence de *gtkada* laisse penser que les types des deux objets doivent être compatibles – des descendants de Gtk_Object – ce qui impose possiblement au gestionnaire de signal de gérer des conversions. Du coup, il semble plus simple d'utiliser les *User_*_Callback*.

6 Le premier paramètre du connecteur est l'objet émetteur de signal mais il se pourrait que certains gestionnaires de signaux n'en ait pas l'usage alors qu'ils ont besoin d'un autre objet. Il semble que le connecteur *Object_connect* réponde à ce besoin.

Dans ces conditions, la levée de la panne s'apparente à un jeu de piste. D'autant que les « modèles » de versions de connecteurs ne sont aisément accessibles, notamment pour le débutant.

Pour réduire la difficulté, il semble préférable de

a) écrire d'abord le gestionnaire de signal (implémentation et spécification),

- d'après le signal (http://beru.univ-brest.fr/~singhoff/DOC/LANG/ADA/gtkada_rm/gtkada_rm_122.html), est-ce une *function* nécessitant un paquet de connecteurs **_Return_Callback* ou bien une *procedure* impliquant un **_Callback ?*

b) d'après les données nécessaires au gestionnaire de signal, déduire le connecteur à utiliser :

- a-t-il besoin d'accéder à un objet autre que l'émetteur du signal ?

- si ce n'est pas le cas, recourir au connecteur *Connect* d'un paquet de connecteurs *Return_Callback* ou *Callback*,

- si c'est le cas, a-t-il besoin d'accéder aussi à l'objet émetteur du signal ?

  • Dans ce sous-cas, utiliser le connecteur *Connect* d'un paquet de connecteurs *User_Return_Callback* ou *User_Callback*,

  • Sinon, utiliser le connecteur *Object_connect* d'un paquet de connecteurs *Return_Callback* ou *Callback.*

c) instancier, au niveau de la bibliothèque, le paquet choisi en tenant compte des types des objets mis en jeu,

d) écrire l'instruction de connexion utilisant l'instance de paquets de connecteurs créée et le connecteur choisi.

D'après la documentation,

- il existe une manière plus simple de procéder, pour les gestionnaires de signaux pour lesquels un médiateur prédéfini existe, capable de tenir compte d'un argument supplémentaire ; on peut donc toujours tenter une connexion du type

```
Instance_du_paquet.Connect (Button, "clicked", On_Clicked'Access);
```

et recourir à la méthode précédente en cas d'échec.

- il y a aussi une méthode plus générale, notamment utiles pour la création de nouveaux types d'objets, et non abordée ici.

Sauf impératif particulier, l'auteur s'en tient à la méthode explicite décrite ci-dessus, qui peut se résumer par la table synoptique suivante.

Connecteur suivant que le gestionnaire de signal…		renvoie une valeur (*function*)	ne renvoie pas de valeur (*procedure*)
n'a pas besoin d'un objet « utilisateur » autre que l'objet émetteur de signal		*Return_Callback.Connect*	*Callback.Connect*
attend un tel objet « utilisateur » supplémentaire et	a besoin de l'objet émetteur du signal	*User_Return_Callback.Connect*	*User_Callback.Connect*
	n'a pas besoin de l'objet émetteur du signal	*Return_Callback.Object_Connect* [ø]	*Callback.Object_Connect* [ø]

Table théorique de choix du connecteur en fonction du gestionnaire de signal.
[ø] peu commode, utiliser plutôt la ligne précédente, comme si le gestionnaire avait besoin de l'objet émetteur.

En définitive, dans les cas simples, il est possible de se faciliter la programmation en utilisant la table pratique suivante.

Connecteur suivant que le gestionnaire de signal…	renvoie une valeur (*function*)	ne renvoie pas de valeur (*procedure*)
n'a pas besoin d'un objet « utilisateur » autre que l'objet émetteur de signal	*Return_Callback.Connect*	*Callback.Connect*
a besoin d'un tel objet « utilisateur » supplémentaire	*User_Return_Callback.Connect*	*User_Callback.Connect*

Table pratique de choix du connecteur en fonction du gestionnaire de signal.

*Test de l'égalité d'un objet à **null***

L'objet « champ de résultat » est connu du paquet *Actions_specifiques* seulement à partir du moment où il lui est transmis par le gestionnaire de signal *Si_clic_sur_calculer_la_clef* (paramètre VClef). Il faut donc pour cela qu'au moins un calcul ait été lancé. Au premier clic sur le bouton, la variable globale (objet) Valeur_clef_glocal du paquet reçoit pour valeur l'objet VClef dans *Si_clic_sur_calculer_la_clef* et le met ainsi à disposition des autres gestionnaires de signaux. Avant cela, ces derniers ne peuvent pas y avoir accès et cet objet Valeur_clef_glocal est donc initialisé à ***null*** lors de sa déclaration dans le paquet *Actions_specifiques*.

Cependant, les gestionnaires de signaux des champs de saisie réécrivent dans le champ de résultat la valeur « ## » lorsque l'utilisateur est en train de modifier le NIR. Ceci est à la fois inutile avant le premier calcul (puisque le champ contient déjà cette valeur « ## ») et impossible (puisque Valeur_clef_glocal n'est pas encore disponible). Cette réécriture n'est donc effectuée que si Valeur_clef_glocal n'est pas ***null***.

Toutefois, l'instruction

```
if not (Valeur_Clef_glocal=null)
     then
          Valeur_Clef_glocal.Set_Text(Text => "##") ;
end if ;
```

provoque une erreur du compilateur. En effet, il déclare que l'opérateur de comparaison n'est pas directement visible et ajoute "*use clause would make operation legal*". Ce conseil du compilateur n'est pas très judicieux car cette clause réduit la lisibilité, n'est pas fine et ne protège pas contre une éventuelle surcharge plus englobante. Un bien meilleur contrôle est obtenu en renommant seulement l'opérateur et seulement dans le paquet *Actions_specifiques* par l'instruction suivante :

```
function "=" (Left, Right : Gtk.GEntry.Gtk_Entry) return Boolean
     renames Gtk.GEntry."=" ;
```

Distribution

La distribution de l'application, c'est-à-dire son transport sur un autre ordinateur, appelle deux observations.

D'une part, l'application dépend de la bibliothèque gtkada, qui doit donc être installée sur la machine cible. Cependant, transporter directement l'exécutable ne fonctionne pas (différences de versions, liens non créés, …). Ainsi, sur la machine cible, il faut d'abord installer gtkada, copier les fichiers sources du projet et recompiler l'application (à cet effet, il est commode d'installer gprbuid).

D'autre part, pour développer l'application, il est commode d'importer l'interface depuis un fichier glade externe par :

```
Import_glade_OK := Gtk.Builder.Add_From_File
     (Builder  => mon_interface,
     Filename => "interface.glade");
```

Cependant cette méthode utilisant Add_from_file fait dépendre l'exécutable du fichier glade qui doit donc l'accompagner. Pour éviter cet inconvénient, une fois l'interface figée, avec un éditeur de texte

comme gedit, il est aisé de fabriquer la chaîne de description XML de l'interface et de l'affecter à une variable mon_interface_texte pour l'incorporer au programme principal. L'importation se fait alors par :

```
Import_glade_OK := Gtk.Builder.Add_From_String
     (Builder=> mon_interface,
     Buffer=> mon_interface_texte,
     Length => mon_interface_texte'Length) ;
```

Une méthode de fabrication de la chaîne testée avec gedit consiste à :

a) ouvrir le fichier .glade avec l'éditeur de texte ;

b) copier tout son contenu dans un nouveau document de l'éditeur (pour ne pas modifier le fichier glade) ;

c) refermer le fichier .glade ;

d) dans le nouveau document, chercher tous les retours à la ligne (\n) et les remplacer par une chaîne vide ;

e) rechercher toutes les occurrences d'un double espace et les remplacer par un espace unique ; recommencer jusqu'à élimination complète des doubles espaces ;

f) rechercher toutes les occurrences de > < (avec un espace au milieu) et les remplacer par ><

A cette étape, noter le nombre de caractères de la chaîne.

g) rechercher toutes les occurrences d'un guillemet " et les remplacer par deux guillemets ""

h) rechercher toutes les occurrences de >< et les remplacer par >"**\n\t&** "<

i) ajouter un guillemet " au tout début et un autre " à la toute fin du document ;

j) copier le tout au second membre de l'initialisation de mon_interface_texte et inscrire le nombre de caractères dans la contrainte du type String.

Cela donne une instruction similaire à la suivante (ici tronquée, « […] » ne fait pas partie de la chaîne) :

```
mon_interface_texte: String(1..1477) := "<?xml version=""1.0"" encoding=""UTF-8""?>"
          & "<interface>"
          & "<requires lib=""gtk+"" version=""2.24""/>"
[…]
          & "<packing>"
          & "<property name=""expand"">True</property>"
          & "<property name=""fill"">True</property>"
          & "<property name=""position"">1</property>"
          & "</packing>"
          & "</child>"
          & "</object>"
          & "</child>"
          & "</object>"
          & "</interface>" ;
```

NOTA: chaque double guillemet dans la chaîne compte pour un seul caractère (mécanisme d'échappement de Ada).

Conclusion

Développée par un débutant en Ada, cette application n'est pas parfaite. À cet égard, ce n'est pas un modèle au sens d'un exemple à suivre absolument.

Cependant, elle présente des solutions aux problèmes épineux qui consomment beaucoup de temps au débutant. Par ailleurs, elle peut servir de modèle au sens où la structure employée peut être ré-utilisée pour une autre application simple à une seule fenêtre, en changeant les sections indiquées comme « spécifiques à l'application ».

La comparaison des deux annexes montrent que la version graphique de l'application demande un peu plus de code que la version en mode texte;-).

Annexe 1 - Un programme élémentaire au terminal

```
-- ========================= Clef INSEE au terminal ==========================================================
-- Ce programme calcule la clef de validation d'un identifiant INSEE (NIR) depuis ses 13 premiers chiffres.
-- Exemple: 1111111111111 a pour clef 20. La clef est 97 - reste de la division par 97.
-- Elle se calcule facilement en prenant des tranches de 2 chiffres. Exemple: 1 11 11 11 11 11 11 = T6 T5 T4 T3 T2 T1 T0.
-- Ainsi le nombre est la somme de Ti*100^i= Ii*(97+3)^i = Ti*(97^i + Ci(i-1) 97^(i-1)*3 + ... + Ci1*97*3^(i-1) + 3^i)= Ti*3^i + multiple de 97.
-- Des lors, le reste du nombre est identique au reste de T0+3*T1+9*T2+27*T3+81*T4+243*T5+729*T6. De plus, 243 = 2*97 + 49 et 729 = 7*97 + 50.
-- Donc le reste est identique à celui de T0+3*T1+9*T2+27*T3+81*T4+49*T5+50*T6. Finalement, le calcul ne met en jeu que des petits nombres.

with Ada.Text_IO, Ada.Integer_Text_IO ; -- Paquets necessaires pour les entrees-sorties au terminal
use Ada;                                -- Prefixe  pour les appels de sous-programmes de ces paquets

procedure Clef_INSEE is
   Num_INSEE: String(1..13) ;
   T0, T1, T2, T3, T4, T5, T6, T: Integer ;
begin
   Text_IO.Put_Line("Donner un identifiant INSEE (NIR) sans sa clef (13 chiffres) :");
   Num_INSEE:= Text_IO.Get_Line;
   -- Tranches du nombre
   T6:= Integer'Value(Num_INSEE(1..1)) ;   -- La fonction Integer'Value attend un STRING non un CHARACTER donc ne pas utiliser Num_INSEE(1) !
   T5:= Integer'Value(Num_INSEE(2..3)) ;   -- Elle le convertit en entier.
   T4:= Integer'Value(Num_INSEE(4..5)) ;
   T3:= Integer'Value(Num_INSEE(6..7)) ;
   T2:= Integer'Value(Num_INSEE(8..9)) ;
   T1:= Integer'Value(Num_INSEE(10..11)) ;
   T0:= Integer'Value(Num_INSEE(12..13)) ;

   -- Calcul de la clef
   T:= T0 + 3*T1 + 9*T2 + 27*T3 + 81*T4 + 49*T5 + 50*T6 ;      -- Calcul du nombre T ayant le même reste que le NIR dans la division par 97
   T:= T mod 97 ;                                              -- Calcul du reste
   T:= 97 - T ;                                                -- Calcul de la clef

   -- Affichage de la clef
   -- Note: Text_IO.Put_Line("Clef =" & Integer'Image(T));     -- met une espace avant un nombre positif, pas toujours commode
   Text_IO.Put("Sa clef est =");
   Integer_Text_IO.Put(T, Width => 3);                         -- Ne met pas une espace initiale, utilise la largeur sans 0 initial
   Text_IO.New_Line;

end Clef_INSEE ;
```

Annexe 2 : fichiers sources et glade du projet

Fichier projet clef_insee_gui.gpr

```
with "gtkada";

project clef_insee_gui is

   for Main use ("clef_insee.adb");
   for Source_Dirs use ("src");
   for Object_Dir use "obj";
   for Source_Files use
     ("actions_specifiques.adb",
      "actions_specifiques.ads",
      "clef_insee.adb");
   for Languages use ("Ada");

end clef_insee_gui ;
```

Comme le montre la capture ci-dessus, bien que le fichier projet ne soit pas écrit en Ada, sa syntaxe est inspirée de celle du langage. Le fichier projet indique les sources du projet : ici, à proprement parler, ce sont les trois fichiers *.adb et *.ads.

Les pages suivantes montrent ces trois fichiers :

- actions_specifiques.ads,
- actions_specifiques.adb,
- clef_insee.adb,

puis le fichier XML de glade :

- interface.glade.

```
-- Ce fichier spécifie les actions spécifiques à l'application
-- Le but de ce fichier de spécification est d'AUTORISER l'écriture des corps des sous-programmes spécifiés…
--      en définissant le "contrat" qu'ils doivent remplir.
with Gtk.GEntry ;
with Gtk.Button ;
with Gtk.Text_Buffer ;
with Gtk.Window ;

Package Actions_specifiques is
   -- Gestionnaires de signaux qui sont des PROCEDURES
   procedure Si_genre_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;
   procedure Si_an_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;
   procedure Si_mois_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;
   procedure Si_dept_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;
   procedure Si_commune_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;
   procedure Si_rang_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) ;

   procedure Si_clic_sur_calculer_la_clef (Object : access Gtk.Button.Gtk_Button_Record'Class; VClef: Gtk.GEntry.Gtk_Entry) ;
   procedure A_louverture (Object : access Gtk.Window.Gtk_Window_Record'Class; BTN: Gtk.Button.Gtk_Button) ;

   -- Gestionnaires de signaux qui sont des FONCTIONS
   function Si_perd_le_focus (Object : access Gtk.GEntry.Gtk_Entry_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean ;
   function Si_fenetre_bouge (Object : access Gtk.Window.Gtk_Window_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean ;

end Actions_specifiques ;
```

Fichier des implémentations actions_specifiques.aadb

```
-- Ce fichier implémente les actions spécifiques à l'application.
with Gtk.GEntry ;
with Gtk.Button ;
with Gtk.Text_Buffer ;
with Gtk.Window ;
with Ada.Text_IO ;      -- pour les sorties à la console
use Ada ;

Package body Actions_specifiques is
   -- L'inclusion du présent paquet dans le programme principal fait des objets déclarés ici des variables
   -- CONSERVEES d'un appel de gestionnaire à l'autre MAIS NON VUES PAR LE PROGRAMME PRINCIPAL.
   -- ==========================================================================
   -- Types utiles pour contrôler les produits (nombres) des champs de saisie du NIR
   subtype Genre is Positive range 1..2 ;
   subtype An is Natural range 0..99 ;
   subtype Mois is Positive range 1..12 ;
   subtype Dept is Positive range 1..99 ;
   subtype CommRang is Positive range 1..999 ;
   subtype Clef is Positive range 1..97 ;
   -- Variables de validation et de lecture des champs de saisie (logiques et chaînes de caractères)
   Genre_OK: boolean := true ;
   ValeurGenre: String(1..1) := "1" ;

   An_OK: boolean := true ;
   ValeurAn: String(1..2) := "11" ;

   Mois_OK: boolean := true ;
   ValeurMois: String(1..2) := "11" ;

   Dept_OK: boolean := true ;
   ValeurDept: String(1..2) := "11" ;

   Commune_OK: boolean := true ;
   ValeurCommune: String(1..3) := "111" ;

   Rang_OK: boolean := true ;
   ValeurRang: String(1..3) := "111" ;

   ValeurNIR: String(1..13) := "1111111111111" ;
   function NIR_OK (GOK, AOK, MOK, DOK, COK, ROK: boolean) return boolean is
   begin
      return GOK and AOK and MOK and DOK and COK and ROK ;
   end NIR_OK ;

   Edition_en_cours: boolean := false ;

   -- ==========================================================================
   -- L'astuce utilisée pour mettre un objet (ou une variable) du programme principal à disposition des gestionnaires de signaux consiste à
   -- a) déclarer ici un tel objet **global pour le présent paquet** contenu dans ce fichier
   -- b) lui affecter comme valeur l'objet du programme principal transmis par les gestionnaires adéquats ou par
   --       un gestionnaire d'initialisation de la fenetre
```

Actions_specifiques.adb, page 1

```
Bouton_global: Gtk.Button.Gtk_Button ;                      -- affecté par le gestionnaire "A_louverture"
Tampon_texte_local: Gtk.Text_Buffer.Gtk_Text_Buffer ;       -- affecté par chaque gestionnaire de champ de saisie "si_xxx_change"
-- ============================================================
Valeur_Clef_glocal: Gtk.GEntry.Gtk_Entry := null ;
-- Dans les gestionnaires de signaux, l'égalité de Valeur_Clef_glocal avec null est testée pour savoir si un premier calcul a déjà été fait.
-- Toutefois, le compilateur déclare que l'opérateur de comparaison n'est pas directement visible et ajoute "use clause would make operation
-- legal". Cependant cette clause réduit la lisibilité, n'est pas fine et ne protège pas contre une éventuelle surcharge plus englobante.
-- Un bien meilleur contrôle est obtenu en renommant seulement l'opérateur et seulement dans ce paquet par l'instruction suivante.
function "=" (Left, Right : Gtk.GEntry.Gtk_Entry) return Boolean renames Gtk.GEntry."=" ;

-- ============================================================
procedure Si_genre_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
-- Cette procédure a besoin de deux arguments: l'actionneur, toujours présent, et la cible des actions de lecture et d'écriture
-- Dans le programme principal, la connexion doit donc faire appel à Gtk.Handlers.User_Callback, qui accepte deux types

begin
   Tampon_texte_local:= BAM_Tampon ;         -- récupération en objet global du tampon défini dans le programme principal et transmis
   if Object.Get_Text/="1" and Object.Get_Text/="2"          -- Object est le champ "Genre" qui doit être "1" ou "2"
   then                                                      -- Non validation de la saisie utilisateur
      BAM_Tampon.Set_Text("⚠ Le champ ""Genre"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
      Genre_OK:= false ;
   else                                                      -- Validation de la saisie
      BAM_Tampon.Set_Text("☺ Le champ ""Genre"" est conforme. ☺") ;
      Genre_OK:= true ;
      ValeurGenre:= Object.Get_Text ;
      Bouton_global.Set_Sensitive(true) ;
   end if ;
   -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
   if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
   -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
   Edition_en_cours:= true ;
exception
   when others =>    -- Attrape toutes les erreurs
      Traite_SGC:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_genre_change"". ⚠") ;
         return ;
      end Traite_SGC ;
end Si_genre_change ;
-- ============================================================
procedure Si_an_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
   function Is_of_subtype_An (Chaux : in String) return Boolean is
      -- Cette fonction valide la saisie du champ An
      Auxiliaire : An ;
   begin
      Auxiliaire := An'Value (Chaux) ;       -- Tentative de tirer une valeur de sous-type An de la chaîne Chaux
      return true ;                          -- si cela marche, la chaîne est conforme
   exception
      when others => return false ;          -- si cela engendre une erreur, la chaîne n'est pas conforme
   end Is_of_subtype_An ;
```

```
begin
   Tampon_texte_local:= BAM_Tampon ;
   if Object.Get_Text'Length < 2             -- Object est le champ "An", qui doit d'abord être de longueur 2
   then
      An_OK:= false ;
   else
      An_OK:= Is_of_subtype_An(Object.Get_Text) ;
   end if ;

   if not An_OK
   then
      BAM_Tampon.Set_Text("⚠ Le champ ""An"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
   else
      BAM_Tampon.Set_Text("☺ Le champ ""An"" est conforme. ☺") ;
      ValeurAn:= Object.Get_Text ;
      Bouton_global.Set_Sensitive(true) ;
   end if ;
   -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
   if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
   -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
   Edition_en_cours:= true ;
exception
   when others =>    -- Attrape toutes les erreurs
      Traite_SAC:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_an_change"". ⚠") ;
         return ;
      end Traite_SAC ;
end Si_an_change ;
-- ============================================================
procedure Si_mois_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
begin
   Tampon_texte_local:= BAM_Tampon ;
   if Object.Get_Text'Length < 2             -- Object est le champ "Mois", qui doit d'abord être de longueur 2
   then
      Mois_OK:= false ;
   elsif (Object.Get_Text(1)='0' and Object.Get_Text(2) in '1'..'9') or (Object.Get_Text(1)='1' and Object.Get_Text(2) in '0'..'2')
   then                                 -- et qui doit être entre 1 et 12
      Mois_OK:= true ;                  -- Validation de la saisie du champ Mois
   else
      Mois_OK:= false ;
   end if ;

   if not Mois_OK
   then
      BAM_Tampon.Set_Text("⚠ Le champ ""Mois"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
   else
      BAM_Tampon.Set_Text("☺ Le champ ""Mois"" est conforme. ☺") ;
      ValeurMois:= Object.Get_Text ;
      Bouton_global.Set_Sensitive(true) ;
   end if ;
```

```
      -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
      if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
      -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
      Edition_en_cours:= true ;
   exception
      when others =>     -- Attrape toutes les erreurs
         Traite_SMC:
         begin
            Text_IO.Put_Line("⚠ Erreur dans ""Si_mois_change"". ⚠") ;
            return ;
         end Traite_SMC ;
   end Si_mois_change ;
   -- ============================================================
   procedure Si_dept_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
      function Valide_saisie_Dept (Chaux : in String) return Boolean is
         -- Histoire de varier les plaisirs, à la différence des fonctions Is_of_subtype_Xxx des autres gestionnaires de signaux, cette fonction
         -- regroupe toute la validation de l'entrée: chaîne de longueur 2 et conformité de son contenu au type Dept. Son nom est donc adapté.
         -- C'est plus compact et plus modulaire mais moins lisible quant à la vérification du type.
         Auxiliaire : Dept ;
      begin
         if Chaux'Length /=2                    -- Test de la longueur de chaîne
         then
            return false ;                      -- Si la longueur n'est pas 2, la saisie est non conforme
         else                                   -- Validation du type
            Auxiliaire := Dept'Value (Chaux) ;  -- Tentative de conversion
            return true ;                       -- Si la conversion réussit, la donnée saisie est valide
         end if ;
                                                -- mais
      exception                                 -- si la conversion provoque une erreur,
         when others => return false;           --      alors la saisie est non conforme
      end Valide_saisie_Dept ;

   begin
      -- Contrôle de validité de la saisie utilisateur
         Dept_OK:= Valide_saisie_Dept (Object.Get_Text) ;

      if not Dept_OK
      then
         BAM_Tampon.Set_Text("⚠ Le champ ""Département"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
      else
         BAM_Tampon.Set_Text("☺ Le champ ""Département"" est conforme. ☺") ;
         ValeurDept:= Object.Get_Text ;
         Bouton_global.Set_Sensitive(true) ;
      end if ;

      -- Mise à disposition des autres gestionnaires du tampn de texte de la boîte à message
      Tampon_texte_local:= BAM_Tampon ;
      -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
      if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
      -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
      Edition_en_cours:= true ;
```

```
exception
   when others =>     -- Attrape toutes les erreurs
      Traite_SDC:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_dept_change"". ⚠") ;
         return ;
      end Traite_SDC ;
end Si_dept_change ;

-- ============================================================
procedure Si_commune_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
   function Is_of_subtype_CommRang (Chaux : in String) return Boolean is
      Auxiliaire : CommRang ;
   begin
      Auxiliaire := CommRang'Value (Chaux) ;
      return true ;
   exception
      when others => return false;
   end Is_of_subtype_CommRang ;

begin
   Tampon_texte_local:= BAM_Tampon ;
   if Object.Get_Text'Length < 3                    -- Object est le champ "Commune", qui doit d'abord être de longueur 3
   then
      Commune_OK:= false ;
   else
      Commune_OK:= Is_of_subtype_CommRang (Object.Get_Text) ;
   end if ;

   if not Commune_OK
   then
      BAM_Tampon.Set_Text("⚠ Le champ ""Commune"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
   else
      BAM_Tampon.Set_Text("☺ Le champ ""Commune"" est conforme. ☺") ;
      ValeurCommune:= Object.Get_Text ;
      Bouton_global.Set_Sensitive(true) ;
   end if ;
   -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
   if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
   -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
   Edition_en_cours:= true ;
exception
   when others =>     -- Attrape toutes les erreurs
      Traite_SCC:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_commune_change"". ⚠") ;
         return ;
      end Traite_SCC ;
end Si_commune_change ;

-- ============================================================
```

```
procedure Si_rang_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer) is
   function Is_of_subtype_CommRang (Chaux : in String) return Boolean is
      Auxiliaire : CommRang ;
   begin
      Auxiliaire := CommRang'Value (Chaux) ;
      return true ;
   exception
      when others => return false;
   end Is_of_subtype_CommRang ;

begin
   Tampon_texte_local:= BAM_Tampon ;
   if Object.Get_Text'Length < 3                              -- Object est le champ "Rang", qui doit être de longueur 3
   then
      Rang_OK:= false ;
   else
      Rang_OK:= Is_of_subtype_CommRang (Object.Get_Text) ;
   end if ;

   if not Rang_OK
   then
      BAM_Tampon.Set_Text("⚠ Le champ ""N° d'ordre"" n'est pas conforme. CORRECTION NECESSAIRE ! ⚠") ;
   else
      BAM_Tampon.Set_Text("☺ Le champ ""N° d'ordre"" est conforme. ☺") ;
      ValeurRang:= Object.Get_Text ;
      Bouton_global.Set_Sensitive(true) ;
   end if ;
   -- L'intruction suivante remet à "##" la valeur de clef si l'on fait un autre calcul sans relancer l'application
   if not (Valeur_Clef_glocal=null) then Valeur_Clef_glocal.Set_Text(Text => "##") ; end if ; -- Légal grâce au renommage de Gtk.GEntry."="
   -- Puisque le champ change, c'est qu'une saisie de NIR est en cours
   Edition_en_cours:= true ;
exception
   when others =>    -- Attrape toutes les erreurs
      Traite_SRC:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_rang_change"". ⚠") ;
         return ;
      end Traite_SRC ;
end Si_rang_change ;
-- ============================================================
procedure Si_clic_sur_calculer_la_clef (Object : access Gtk.Button.Gtk_Button_Record'Class; VClef: Gtk.GEntry.Gtk_Entry) is
   pragma Unreferenced (Object) ;    -- directive de compilation pour dire que c'est sciemment que Object n'est pas utilisé dans la fonction
   T0, T1, T2, T3, T4, T5, T6, T: Integer ; -- Tranches de NIR et auxiliaire
   ValeurClef: Clef ;
   ValeurClefChaine: String(1..2) := "00" ;
begin
   if NIR_OK(GOK => Genre_OK, AOK => An_OK, MOK => Mois_OK, DOK => Dept_OK, COK => Commune_OK, ROK => Rang_OK)
   then      -- Si toutes les saisies de champs sont OK, alors
      begin  -- le programme procède au calcul de la clef
         -- Si l'utilisateur a cliqué sur le bouton de calcul, c'est qu'il n'est plus en train de faire une saisie
         Edition_en_cours:= false ;
```

```
            -- Le NIR est d'abord obtenu par concaténation des champs validés
            ValeurNIR:= ValeurGenre & ValeurAn & ValeurMois & ValeurDept & ValeurCommune & ValeurRang ;
            Text_IO.Put_Line("Début du calcul pour le NIR " & ValeurNIR) ;      -- Sortie à la console pour info
            -- Le NIR est ensuite découpé en tranches de deux chiffres en partant de la droite. La tranche la plus à gauche contient 1 chiffre.
            T6:= Integer'Value(ValeurNIR(1..1)) ;                              -- Début du calcul
            T5:= Integer'Value(ValeurNIR(2..3)) ;
            T4:= Integer'Value(ValeurNIR(4..5)) ;
            T3:= Integer'Value(ValeurNIR(6..7)) ;
            T2:= Integer'Value(ValeurNIR(8..9)) ;
            T1:= Integer'Value(ValeurNIR(10..11)) ;
            T0:= Integer'Value(ValeurNIR(12..13)) ;
            T:= T0 + 3*T1 + 9*T2 + 27*T3 + 81*T4 + 49*T5 + 50*T6 ;             -- Calcul d'un nombre de reste identique
            T:= T mod 97 ;                                                     -- Calcul du reste
            ValeurClef:= 97 - T ;                                              -- Fin du calcul de la clef numérique
            Text_IO.Put_Line("La valeur numérique calculée est " & ValeurClef'Img) ;
            if ValeurClef < 10                                                 -- Traduction en chaine de longueur 2
            then
               ValeurClefChaine:= "0" & ValeurClef'Img(2..2) ;                 -- il y a une espace au début de Img !
            else
               ValeurClefChaine:= ValeurClef'Img(2..3) ;                       -- il y a une espace au début de Img !
            end if ;
            Text_IO.Put_Line("La valeur chaîne calculée est " & ValeurClefChaine) ;
            Vclef.Set_Text(Text => ValeurClefChaine) ;                         -- Écriture de la chaîne dans le champ Clef
            Valeur_clef_glocal:= Vclef ;
            Tampon_texte_local.Set_Text("☺ Tout va bien ! ☺") ;
         end ;
      -- la branche else ne peut pas être atteinte parce que le gestionnaire Si_perd_le_focus désactive le bouton si NIR n'est pas OK
      else      -- Si au moins l'un des champs n'est pas valide, alors NIR n'est pas OK
         null ; -- et donc les instructions ici ne seraient jamais atteintes
      end if ;
   exception
      when others =>    -- Attrape toutes les erreurs
         Traite_SCCC:
         begin
            Text_IO.Put_Line("⚠ Erreur dans ""Si_clic_sur_calculer_la_clef"". ⚠") ;
            return ;
         end Traite_SCCC ;
   end Si_clic_sur_calculer_la_clef ;

   -- ============================================================
   function Si_perd_le_focus (Object : access Gtk.GEntry.Gtk_Entry_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean is
      pragma Unreferenced (Object) ;    -- directive de compilation pour dire que c'est sciemment que Object n'est pas utilisé dans la fonction

   begin
      if NIR_OK(GOK => Genre_OK, AOK => An_OK, MOK => Mois_OK, DOK => Dept_OK, COK => Commune_OK, ROK => Rang_OK)
      then                              -- Si le NIR est conforme, alors
         ValeurNIR:= ValeurGenre & ValeurAn & ValeurMois & ValeurDept & ValeurCommune & ValeurRang ;    -- on le constitue en une seule chaîne
         Text_IO.Put_Line("NIR soumis = " & ValeurNIR & " : conforme.") ;                               -- Sortie à la console pour info
         if Edition_en_cours
         then                           -- et si une saisie est bien en cours alors
            BTN.Set_Sensitive(Sensitive => true) ;      -- Le bouton est rendu actif
```

```
      else
         null ;                                    -- le bouton reste inactif
      end if ;

   else                              -- Si le NIR N'EST PAS conforme
      BTN.Set_Sensitive(Sensitive => false) ;        -- le bouton est rendu inactif
      ValeurNIR:= "             " ;                 -- initialisation avec 13 espaces (pas vraiment indispensable)
      if Genre_OK then ValeurNIR(1..1):= ValeurGenre ; else ValeurNIR(1..1):= "." ; end if ;         -- Pour chaque champ, on recopie
      if An_OK then ValeurNIR(2..3):= ValeurAn ; else ValeurNIR(2..3):= ".." ; end if ;              -- la valeur si le champ est OK
      if Mois_OK then ValeurNIR(4..5):= ValeurMois ; else ValeurNIR(4..5):= ".." ; end if ;          -- ou des . sinon.
      if Dept_OK then ValeurNIR(6..7):= ValeurDept ; else ValeurNIR(6..7):= ".." ; end if ;
      if Commune_OK then ValeurNIR(8..10):= ValeurCommune ; else ValeurNIR(8..10):= "..." ; end if ;
      if Rang_OK then ValeurNIR(11..13):= ValeurRang ; else ValeurNIR(11..13):= "..." ; end if ;
      Text_IO.Put_Line("NIR soumis = " & ValeurNIR & " : non conforme.") ;           -- La sortie console montre les champs non valides
   end if ;
   return false ; -- Pour que le gestionnaire fonctionne sans avertissement, il faut renvoyer FALSE !
   -- Ensuite on attend que l'utilisateur corrige sa saisie !
exception
   when others =>    -- Attrape toutes les erreurs
      Traite_SPLF:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""Si_perd_le_focus"". ⚠") ;
         return true ;
      end Traite_SPLF ;
end Si_perd_le_focus ;

-- ============================================================
function Si_fenetre_bouge (Object: access Gtk.Window.Gtk_Window_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean is
   pragma Unreferenced (Object) ;    -- directive de compilation pour dire que c'est sciemment que Object n'est pas utilisé dans la fonction
   -- Cette fonction doit rendre le bouton inactif lorsque la fenêtre bouge (c'est-à-dire est déplacée, redimensionnée, ...)
   -- sauf si l'utilisateur est en train de saisir un NIR, ce qui est indiqué par la variable logique Edition_en_cours ;
begin
   if Edition_en_cours
   then
      null ;
   else
      Bouton_global.Set_Sensitive(false) ;
   end if ;
   return true ;
end Si_fenetre_bouge ;

-- ============================================================
procedure A_louverture (Object : access Gtk.Window.Gtk_Window_Record'Class; BTN: Gtk.Button.Gtk_Button) is
   pragma Unreferenced (Object) ;    -- directive de compilation pour dire que c'est sciemment que Object n'est pas utilisé dans la fonction

begin
   Bouton_global:= BTN ;     --  la variable "globale" (pour ce paquet) Bouton_global contient maintenant Bouton_calculer
   Bouton_global.Set_Sensitive(false) ;      -- A l'ouverture, le bouton est désactivé pour que seule une saisie soit possible
   Text_IO.Put_Line("Etat de la sensibilité du bouton à l'ouverture de la fenêtre :" & Bouton_global.Is_Sensitive'Img) ;     -- Idem
   -- A l'ouverture, l'utilisateur n'est pas en train de faire une saisie de NIR donc:
   Edition_en_cours:= false ;
```

```
exception
   when others =>    -- Attrape toutes les erreurs
      Traite_ALO:
      begin
         Text_IO.Put_Line("⚠ Erreur dans ""A_louverture"". ⚠") ;
         return ;
      end Traite_ALO ;
   end A_louverture ;

end Actions_specifiques ;
```

Actions_specifiques.adb, page 9

Fichier clef_insee.adb (programme principal)

```
-- Le nom du fichier doit être EN MINUSCULES
-- ==========================================================
-- ========================= Clef INSEE =====================
-- ==========================================================
-- Cette application calcule la clef de contrôle du numéro d'inscription au
-- répertoire des personnes physiques, communément abrégé en NIR et aussi
-- appelé numéro de sécurité sociale. En réalité, elle prend le prétexte de
-- ce calcul très simple pour illustrer l'emploi combiné du langage ada, du
-- portage sous ada de gtk (gtkada) et du concepteur d'interface graphique
-- glade pour programmer une application graphique sous Linux.
-- ==========================================================
-- Plus précisément, l'application est développée sous la distribution Ubuntu
-- Trusty Tahr 14.04.5 LTS 64 bits avec le bureau MATE Desktop Environment
-- 1.8.2. La configuration de l'environnement de programmation est installée
-- depuis les dépots de la distribution pour en garantir la cohérence.
-- Les versions disponibles installées à l'heure où ce commentaire est écrit
-- sont les suivantes :
--    * GTK 2 et 3 sont présents sur le système, en versions respectives
--          2.24.27-0ubuntu1~trusty1 et 3.10.8-0ubuntu1.4
--    * Glade 3.16.1 est présent et capable d'utiliser GTK 3.10 mais, pour
--          cette application, on a utilisé le module limité à
--          GTK2 (glade-gtk2 version 3.8.0) réglé pour utiliser GTK 2.24
--          avec le format gtkbuilder.
--    * Le compilateur est Gnat et, bien que la version 4.8 soit disponible,
--          c'est ici la version 4.6 qui a dû être employée
--    * La liaison avec Ada, gtkada, a été installée par Synaptic en version
--          2.24.1-14
--    * L'EDI Gnat Programming studio est en version GPS 5.0-16 (pour la
--          famille Debian)
-- OpenGL n'est pas utile ici.
-- ==========================================================
-- Ce programme principal est la tour de contrôle de l'application. Outre
-- l'initialisation de l'application et le lancement de la boucle d'attente
-- événementielle, son rôle principal est la connexion des signaux émis par
-- l'interface aux gestionnaires de signaux constituant l'application.
-- La table suivante guide le choix des connecteurs à mettre en œuvre.
--
--  +-----------------------+-----------------------+-----------------------+-----------------------+
--  |                       | Connecteur suivant que| renvoie une valeur    | ne renvoie pas une valeur |
--  |                       | le gestionnaire de signal…| (function)        | (procedure)           |
--  +-----------------------+-----------------------+-----------------------+-----------------------+
--  | n'a pas besoin d'un objet "utilisateur" | a besoin de l'objet | Return_Callback.Connect | Callback.Connect |
--  | autre que l'objet émetteur du signal | émetteur du signal | | |
--  +-----------------------+-----------------------+-----------------------+-----------------------+
--  | attend un tel objet "utilisateur" | a besoin de l'objet émetteur du signal | User_Return_Callback.Connect | User_Callback.Connect |
--  +-----------------------+-----------------------+-----------------------+-----------------------+
--  | n'a pas besoin de l'objet émetteur du signal ET supplémentaire | Return_Callback.Object_Connect (Peu commode, utiliser plutôt la ligne précédente) | Callback.Object_Connect |
--  +-----------------------+-----------------------+-----------------------+-----------------------+
--
--                Table théorique de choix du connecteur en fonction du gestionnaire de signal
-- L'emploi de Object_Connect étant peu commode, on fait comme si le GS avait toujours besoin de l'objet émetteur => User_*_Callback.Connect
```

```
-- ============================================================
-- ====================== INCLUSIONS ==========================
-- ============================================================
with Gtk.Main;           -- fournit la boucle de détection d'événement
with Gtk.Handlers;       -- fournit les gestionnaires de signaux dont le rappel de sortie
with Text_IO;            -- nécessaire pour les écritures à la console
with Gtk.Window ;        -- nécessaire pour la déclaration de la fenêtre principale

with Gtk.Builder ;       -- POUR GLADE: pour construire l'interface graphique sous forme d'objets
with Gtk.Widget ;        -- POUR GLADE: pour importer la fenêtre principale dans un onjet(widget)
with Glib.Error ;        -- POUR GLADE: pour récupérer une éventuelle erreur comme résultat de l'importation de l'interface

-- ============================================================
-- ==== DEBUT DES INCLUSIONS SPECIFIQUES A L'APPLICATION =====
-- ============================================================
-- Le choix est ici fait de déporter dans un module externe toutes les actions spécifiques à l'application, qu'il faut donc importer:
with Actions_specifiques ;       -- Pour inclure le paquet décrivant les actions spécifiques à l'application
-- Le programme principal va seulement connecter les actionneurs de l'interface à leurs gestionnaires de signaux
-- en leur transmettant les objets actifs et le composant commun sur lequel ils doivent agir, ce qui nécessite les inclusions suivantes.
with Gtk.Button ;                -- Parce qu'il y a dans l'application : un actionneur bouton …
with Gtk.GEntry ;                -- ... des actionneurs champs de saisie
with Gtk.Text_Buffer ;           -- ... un tampon de texte qu'il faut remplir pour alimenter la boîte à texte d'information de l'application
-- ============================================================
-- ==== FIN DES INCLUSIONS SPECIFIQUES A L'APPLICATION =======
-- ============================================================
-- ================== FIN DES INCLUSIONS ======================
-- ============================================================

-- ============================================================
-- ============= DEBUT DU PROGRAMME PRINCIPAL =================
-- ============================================================
procedure clef_insee is
-- ==================== DECLARATIONS ==========================
   -- Les déclarations initiales de cette section sont génériques (identiques pour toute application simple à une seule fenêtre)
   -- ON COMMENCE PAR INSTANCIER LE PAQUET QUI FOURNIT LES UTILITAIRES DE CONNEXION POUR LA FENETRE PRINCIPALE
   -- Le "handler" function Si_fenetre_ppale_close (Object : access Gtk.Window.Gtk_Window_Record'Class) return Boolean
           -- est un gestionnaire de signal "function" => paquet de connecteurs *_Return_Callback
           -- n'a besoin d'un objet autre que l'émetteur de signal => Return_Callback choix Connect
   package Connecteurs_de_fenetre_bouton is new Gtk.Handlers.Return_Callback (Gtk.Window.Gtk_Window_Record, Boolean) ;
   -- ======== Gestionnaire de signal pour quitter ============
   -- C'est le seul gestionnaire de signal générique indispensable.
   -- FONCTION  QUI SERA APPELEE SUR DETECTION DE Delete_Event
   function Si_fenetre_ppale_close (Object : access Gtk.Window.Gtk_Window_Record'Class) return Boolean is     -- implementation
      pragma Unreferenced (Object) ;    -- directive de compilation pour dire que c'est sciemment que Object n'est pas utilisé dans la fonction
   begin
      Text_IO.Put_Line("Fermeture de la fenêtre.") ;    -- Sortie à la console pour information.
      Gtk.Main.Gtk_Exit (0);                            -- Achèvement du processus
      return True ;                                     -- Renvoi de la bonne fin d'exécution
   end Si_fenetre_ppale_close ;
   -- ==== Fin du gestionnaire de signal pour quitter ========
```

```
   Fenetre_principale : Gtk.Window.Gtk_Window ;         -- Pointeur de la fenêtre principale, créé avant de lui associer l'objet correspondant
   -- Variables et objets POUR GLADE :
   Import_glade_OK: Glib.Error.GError ;     -- POUR GLADE : déclaration de l'erreur pour récupérer le résultat de la fonction Add_From_File
   mon_interface: Gtk.Builder.Gtk_Builder ; -- POUR GLADE : déclaration de l'objet pour recueillir la description de l'interface
   FP_widget: Gtk.Widget.Gtk_Widget ;       -- POUR GLADE : widget pour extraire la fenêtre principale de l'interface ^avec le type Gtk_Widget^

-- ============================================================
-- ==============FIN DES DECLARATIONS ==========================
-- ============================================================

begin
-- ==================== IMPLEMENTATION ========================
   -- L'implémentation comporte trois parties: la première et la dernière sont génériques et invariantes.
   -- La deuxième est spécifique à l'application. C'est la seule à faire évoluer pour écrire une autre application simple.

   -- ======================================================================
   -- Début de la partie générique initiale du corps du programme (PARTIE I)
   -- ======================================================================
   -- Initialisation
   Gtk.Main.Set_Locale ;                                -- Chargement des propriétés de localisation du système
   Gtk.Main.Init ;                                      -- Initialisation du programme
   Text_IO.Put_Line("Initialisation effectuée.") ;      -- Sortie à la console pour suivre l'avancement du programme et informer l'utilisateur

   -- ====== CREATION DE LA FENETRE PRINCIPALE - DEBUT DES INSTRUCTIONS POUR L'EMPLOI DE GLADE ===========
   -- Avec Gtk directement, on créerait la nouvelle fenêtre associée au pointeur Fenetre_principale par
   -- Gtk.Window.Gtk_New (Fenetre_principale, Gtk.Enums.Window_Toplevel) ;
   -- et il faudrait ensuite lui ajouter par programmation tous ses composants. Ceci devient vite fastidieux pour une application réelle.
   -- Avec Glade, on crée l'interface graphiquement et Glade fournit sa description sous la forme XML. Il faut donc d'abord importer l'interface
   -- depuis cette description.
   -- Pour cela, en amont, il faut des with supplémentaires et il faut déclarer 2 objets en plus: une erreur et le conteneur pour l'interface
   -- Pour une meilleure lisibilité, on a aussi déclaré ici un objet intermédiaire FP_widget pour mettre en évidence la conversion de type
   Gtk.Builder.Gtk_New(Builder => mon_interface) ;              -- Création de l'objet pour recevoir l'interface
   Import_glade_OK := Gtk.Builder.Add_From_File(Builder  => mon_interface, Filename => "interface.glade"); -- Importation de l'interface
   FP_widget := mon_interface.Get_Widget("window1") ;           -- Récupération du widget de la fenêtre (!! Get_Widget disparait dans gtk3 !!)
   Fenetre_principale := Gtk.Window.Gtk_Window (FP_widget) ;    -- Création de la fenêtre par conversion en type Gtk_Window (pointeur)
   -- ==== FIN DE CREATION DE LA FENETRE PRINCIPALE - FIN DES INSTRUCTIONS POUR L'EMPLOI DE GLADE =======
   Text_IO.Put_Line("Fenêtre principale créée.") ;              -- Sortie à la console pour information

   -- Connexion du signal de fermeture de la fenêtre au gestionnaire de signal pour quitter
   Connecteurs_de_fenetre_bouton.Connect        -- Connecteur
     (Fenetre_principale,                       -- Objet actionneur
      "delete-event",                           -- Nom du signal
      Si_fenetre_ppale_close'Access             -- gestionnaire de signal sans médiateur
     );
   Text_IO.Put_Line("Signal de fermeture connecté.") ;                      -- Sortie à la console pour information

   -- ======================================================================
   -- = Fin de la partie générique initiale du corps du programme (PARTIE I)
   -- ======================================================================
```

```
   -- ======================================================================
   -- = DEBUT DE LA PARTIE SPECIFIQUE A L'APPLICATION (PARTIE II)        =
   -- ======================================================================
   -- Pour faciliter l'adaptation et améliorer la lisibilité, les éléments spécifiques (déclarations et implémentation) sont groupés ci-dessous
   -- dans un bloc "declare" nommé Actionneurs_specifiques.
   Actionneurs_specifiques :
   declare
   -- Dans les déclarations de ce bloc figurent les instances de Gtk.Handlers.*Callback* nécessaires et les objets actionneurs
   -- ======================================================================
      -- Le "handler" procedure Si_genre_change (Object : access Gtk.GEntry.Gtk_Entry_Record'Class; BAM_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer)
           -- est un gestionnaire de signal "procedure" => paquet de connecteurs *_Callback
           -- a besoin d'un objet BAM_Tampon autre que l'émetteur de signal Object ET a besoin de ce dernier => User_Callback choix Connect
      -- Il en est de même pour tous les gestionnaires de champ Si_xxx_change, donc un seul paquet de connecteurs est déclaré à cet usage
      package Connecteurs_de_champ_message is new Gtk.Handlers.User_Callback(Gtk.GEntry.Gtk_Entry_Record, Gtk.Text_Buffer.Gtk_Text_Buffer) ;
      -- ======================================================================
      -- Le "handler" function Si_perd_le_focus (Object : access Gtk.GEntry.Gtk_Entry_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean
           -- est un gestionnaire de signal "function" => paquet de connecteurs *_Return_Callback
           -- a besoin d'un objet BTN autre que l'émetteur de signal Object                                => User_Return_Callback choix Connect
      package Connecteurs_de_champ_bouton is new Gtk.Handlers.User_Return_Callback
                (Widget_Type => Gtk.GEntry.Gtk_Entry_Record,    -- type de d'objet émetteur
                 Return_Type => boolean,                        -- valeur renvoyée par la fonction gestionnaire de signal
                 User_Type   => Gtk.Button.Gtk_Button           -- type de l'objet "utilisateur"
                ) ;
      -- ======================================================================
      -- Le "handler" procedure Si_clic_sur_calculer_la_clef (Object : access Gtk.Button.Gtk_Button_Record'Class; VClef: Gtk.GEntry.Gtk_Entry)
           -- est un gestionnaire de signal "procedure" => paquet de connecteurs *_Callback
           -- a besoin d'un objet VClef autre que l'émetteur de signal Object => USer_Callback choix Connect
      package Connecteurs_de_bouton_champ is new Gtk.Handlers.User_Callback(Gtk.Button.Gtk_Button_Record, Gtk.GEntry.Gtk_Entry) ;
      -- ======================================================================
      -- Le "handler" procedure A_louverture (Object : access Gtk.Window.Gtk_Window_Record'Class; BTN: Gtk.Button.Gtk_Button)
           -- est un gestionnaire de signal "procedure" => paquet de connecteurs *_Callback
           -- a besoin d'un objet BTN autre que l'émetteur de signal Object => User_Callback choix Connect
      package Connecteurs_de_fenetre_bouton is new Gtk.Handlers.User_Callback(Gtk.Window.Gtk_Window_Record, Gtk.Button.Gtk_Button) ;
      -- ======================================================================
      -- Le "handler" function Si_fenetre_bouge (Object : access Gtk.Window.Gtk_Window_Record'Class ; BTN: Gtk.Button.Gtk_Button) return boolean
           -- est un gestionnaire de signal "function" => paquet de connecteurs *_Return_Callback
           -- a besoin d'un objet BTN autre que l'émetteur de signal Object => User_Return_Callback choix Connect
      package Connecteurs_de_fct_fenetre_bouton is new Gtk.Handlers.User_Return_Callback(Gtk.Window.Gtk_Window_Record, Boolean,
Gtk.Button.Gtk_Button) ;
      -- Gtk.Handlers.User_Callback ou User_Return_Callback permet de passer un objet supplémentaire au gestionnaire de signal.
      -- Pointeurs des objets associés aux objets "champs de saisie" du NIR :
      Champ_genre: Gtk.GEntry.Gtk_Entry ;       -- Sexe
      Champ_an: Gtk.GEntry.Gtk_Entry ;          -- Année de naissance
      Champ_mois: Gtk.GEntry.Gtk_Entry ;        -- Mois de naissance
      Champ_dept: Gtk.GEntry.Gtk_Entry ;        -- Département de naissance
      Champ_commune: Gtk.GEntry.Gtk_Entry ;     -- Commune de naissance
      Champ_rang: Gtk.GEntry.Gtk_Entry ;        -- Numéro d'ordre de naissance
      -- Pointeurs des objets de l'interface chargés par le programme principal à l'usage des gestionnaires de signaux
      Bouton_calculer: Gtk.Button.Gtk_Button ;                          -- Création de l'objet bouton "Calculer la clef"
      Boite_a_message_Tampon: Gtk.Text_Buffer.Gtk_Text_Buffer ;         -- Tampon de texte pour alimenter la boîte à texte d'information
```

```
begin
   -- Importation du tampon de texte de la boîte à message et du bouton
   Boite_a_message_Tampon:= Gtk.Text_Buffer.Gtk_Text_Buffer(mon_interface.Get_Object("TamponTexte")) ; -- Extraction et conversion du tampon
   Bouton_calculer:= Gtk.Button.Gtk_Button(mon_interface.Get_Object("Bouton_calculer")) ;             -- ... et du bouton
   -- Ensuite, on va traiter les actionneurs un par un, en connectant les gestionnaires de signaux nécessaires
   -- ==============================
   -- 0. La fenêtre principale
   -- ==============================
   -- Le gestionnaire de signal A_louverture a pour rôle de mettre l'objet bouton à disposition des autres gestionnaires de signaux,
   -- de désactiver le bouton de calcul pour que seule une saisie soit possible
   Connecteurs_de_fenetre_bouton.Connect             -- Procédure de connexion
     (Fenetre_principale                             -- le widget actionneur = la fenêtre de l'application
      , "show"                                       -- le nom du signal émis par la fenêtre lors de son 1er affichage
      , Actions_specifiques.A_louverture'Access      -- gestionnaire de signal sans médiateur
      , Bouton_calculer                              -- Objet "utilisateur" transmis pour le désactiver
     );
   Text_IO.Put_Line("Signal show connecté à Fenetre_principale.") ;          -- Sortie à la console pour information

   -- Le gestionnaire de signal Si_fenetre_bouge assure que le bouton ne change pas d'état lorsque la fenêtre est déplacée ou redimensionnée
   Connecteurs_de_fct_fenetre_bouton.Connect         -- Connecteur
     (Fenetre_principale                    -- le widget actionneur = la fenêtre de l'application
      , "configure-event"                   -- le nom du signal émis par la fenêtre lorsqu'elle bouge
      , Actions_specifiques.Si_fenetre_bouge'Access -- lien vers le gestionnaire de signal
      , Bouton_calculer                     -- Objet "utilisateur" pour le désactiver si une saisie n'est pas en cours
     );
    Text_IO.Put_Line("Signal show connecté à Fenetre_principale.") ;         -- Sortie à la console pour information

   -- Pour chaque champ de saisie "xxx", il y a deux gestionnaires de signaux à connecter:
   -- a/ Si_xxx_change vérifie si la donnée juste saisie par l'utilisateur rang le champ conforme à la norme du champ et affiche un message
   -- dans la boîte à message de la fenêtre pour valider l'entrée ou demander une correction. Il y a un tel gestionnaire par champ.
   -- b/ Si_perd_le_focus désactive le bouton de calcul si le champ est non conforme lorsque l'utilisateur en sort
   --        Dans ce second cas, le même gestionnaire est utilisé pour tous les champs.
   -- ==============================
   -- 1. Le champ "Genre"
   -- ==============================
   Champ_genre:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("Genre")) ;   -- Importation de l'objet pour le champ de saisie du genre
   Connecteurs_de_champ_message.Connect                      -- Connecteur choisi en fonction du besoin de si_xxx_change
     (Champ_genre                                            -- le widget actionneur, qui sera lu
      , "changed"                                            -- le nom du signal
      , Actions_specifiques.Si_genre_change'Access           -- gestionnaire de signal cible, sans marshaller
      , Boite_a_message_Tampon                               -- Objet "donnée utilisateur", pour y écrire
     );

   Connecteurs_de_champ_bouton.Connect                          -- Connecteur
     (Widget    => Champ_genre                                  -- le widget actionneur
      , Name      => "focus-out-event"                          -- le nom du signal
      , Cb     => Actions_specifiques.Si_perd_le_focus'Access   -- gestionnaire de signal
      , User_Data => Bouton_calculer                            -- objet "donnée utilisateur" pour changer son état actif/inactif
      -- , After     =>                                         -- facultatif
     ) ;
    Text_IO.Put_Line("Signal focus-out-event connecté à champ_genre.") ;      -- Sortie à la console pour information
```

```
   -- ================================
   -- 2. Le champ "An"
   -- ================================
Champ_an:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("An")) ; -- Importation de l'objet pour le champ de saisie de l'année
Connecteurs_de_champ_message.Connect              -- Connecteur
  (Champ_an                                       -- le widget actionneur
   , "changed"                                    -- le nom du signal
   , Actions_specifiques.Si_an_change'Access      -- gestionnaire de signal
   , Boite_a_message_Tampon                       -- Objet "donnée utilisateur", pour y écrire
  );

Connecteurs_de_champ_bouton.Connect                             -- Connecteur
  (Widget    => Champ_an                                        -- le widget actionneur
   , Name      => "focus-out-event"                             -- le nom du signal
   , Cb     => Actions_specifiques.Si_perd_le_focus'Access      -- gestionnaire de signal
   , User_Data => Bouton_calculer                               -- objet "donnée utilisateur" pour changer son état actif/inactif
   -- , After     =>                                            -- facultatif
  ) ;
 Text_IO.Put_Line("Signal focus-out-event connecté à Champ_an.") ;              -- Sortie à la console pour information

   -- ================================
   -- 3. Le champ "Mois"
   -- ================================
Champ_mois:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("Mois")) ;
Connecteurs_de_champ_message.Connect              -- Connecteur
  (Champ_mois                                     -- le widget actionneur
   , "changed"                                    -- le nom du signal
   , Actions_specifiques.Si_mois_change'Access    -- gestionnaire de signal
   , Boite_a_message_Tampon                       -- Objet "donnée utilisateur", pour y écrire
  );

Connecteurs_de_champ_bouton.Connect                             -- Connecteur
  (Widget    => Champ_mois                                      -- le widget actionneur
   , Name      => "focus-out-event"                             -- le nom du signal
   , Cb     => Actions_specifiques.Si_perd_le_focus'Access      -- gestionnaire de signal
   , User_Data => Bouton_calculer                               -- objet "donnée utilisateur" pour changer son état actif/inactif
   -- , After     =>                                            -- facultatif
  ) ;
 Text_IO.Put_Line("Signal focus-out-event connecté à Champ_mois.") ;            -- Sortie à la console pour information

   -- ================================
   -- 4. Le champ "Dept"
   -- ================================
Champ_dept:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("Dept")) ;
Connecteurs_de_champ_message.Connect                            -- Connecteur
  (Champ_dept                                                   -- le widget actionneur
   , "changed"                                                  -- le nom du signal
   , Actions_specifiques.Si_dept_change'Access                  -- gestionnaire de signal
   , Boite_a_message_Tampon                                     -- Objet "donnée utilisateur", pour y écrire
  );
```

```
Connecteurs_de_champ_bouton.Connect                                  -- Connecteur
  (Widget    => Champ_dept                                           -- le widget actionneur
   , Name      => "focus-out-event"                                  -- le nom du signal
   , Cb     => Actions_specifiques.Si_perd_le_focus'Access          -- gestionnaire de signal
   , User_Data => Bouton_calculer                                    -- objet "donnée utilisateur" pour changer son état actif/inactif
   -- , After     =>                                                 -- facultatif
  ) ;
 Text_IO.Put_Line("Signal focus-out-event connecté à Champ_dept.") ;             -- Sortie à la console pour information

   -- ================================
   -- 5. Le champ "Commune"
   -- ================================
Champ_commune:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("Commune")) ;
Connecteurs_de_champ_message.Connect                                 -- Connecteur
  (Champ_commune                                                     -- le widget actionneur
   , "changed"                                                       -- le nom du signal
   , Actions_specifiques.Si_commune_change'Access                    -- gestionnaire de signal
   , Boite_a_message_Tampon                                          -- Objet "donnée utilisateur", pour y écrire
  );

Connecteurs_de_champ_bouton.Connect                                  -- Connecteur
  (Widget    => Champ_commune                                        -- le widget actionneur
   , Name      => "focus-out-event"                                  -- le nom du signal
   , Cb     => Actions_specifiques.Si_perd_le_focus'Access          -- gestionnaire de signal
   , User_Data => Bouton_calculer                                    -- objet "donnée utilisateur" pour changer son état actif/inactif
   -- , After     =>                                                 -- facultatif
  ) ;
 Text_IO.Put_Line("Signal focus-out-event connecté à Champ_commune.") ;          -- Sortie à la console pour information

   -- ================================
   -- 6. Le champ "Rang"
   -- ================================
Champ_rang:= Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("Rang")) ;
Connecteurs_de_champ_message.Connect                                 -- Connecteur
  (Champ_rang                                                        -- le widget actionneur
   , "changed"                                                       -- le nom du signal
   , Actions_specifiques.Si_rang_change'Access                       -- gestionnaire de signal
   , Boite_a_message_Tampon                                          -- Objet "donnée utilisateur", pour y écrire
  );

Connecteurs_de_champ_bouton.Connect                                  -- Connecteur
  (Widget      => Champ_rang                                         -- le widget actionneur
   , Name      => "focus-out-event"                                  -- le nom du signal
   , Cb        => Actions_specifiques.Si_perd_le_focus'Access        -- gestionnaire de signal
   , User_Data => Bouton_calculer                                    -- objet "donnée utilisateur" pour changer son état actif/inactif
   -- , After     =>                                                 -- facultatif
  ) ;
 Text_IO.Put_Line("Signal focus-out-event connecté à Champ_rang.") ;             -- Sortie à la console pour information
```

```
         -- ================================
         -- 7. Le bouton "Calculer_la_clef"
         -- ================================
         -- Enfin, il y a un gestionnaire de signal pour le bouton: il lui incombe de calculer la valeur de la clef.
      Text_IO.Put_Line("Bouton Calculer_la_clef créé.") ;                              -- Sortie à la console pour information
      Connecteurs_de_bouton_champ.Connect                                              -- Connecteur
        (Widget      => Bouton_calculer                                                -- * le widget actionneur
         , Name      => "clicked"                                                      -- * le nom du signal
         , Cb        => Actions_specifiques.Si_clic_sur_calculer_la_clef'Access        -- * gestionnaire de signal, sans médiateur
         , User_Data => Gtk.GEntry.Gtk_Entry(mon_interface.Get_Object("ValeurClef"))    -- * objet "utilisateur" pour y mettre le résultat
         -- , After     =>                                                            -- facultatif
        ) ;
      Text_IO.Put_Line("Signal de clic sur Calculer_la_clef connecté.") ;                          -- Sortie à la console pour information
      Text_IO.Put_Line("Etat de la sensibilité du bouton à la connexion :" & Bouton_calculer.Is_Sensitive'Img) ;        -- Idem

   end Actionneurs_specifiques ;

   -- ======================================================================
   -- = FIN DE LA PARTIE SPECIFIQUE A L'APPLICATION (PARTIE II)           =
   -- ======================================================================

   -- ======================================================================
   -- Début de la partie générique finale du programme principal (PARTIE III)
   -- ======================================================================
   -- Rendre visible la fenêtre
   Gtk.Window.Show_All (Fenetre_principale);     -- Affichage de la fenêtre principale
   Text_IO.Put_Line("Fenêtre affichée.") ;       -- Sortie à la console pour information
   -- Lancer la boucle de détection d'événement
   Text_IO.Put_Line("Lancement de la boucle de détection d'événements.") ;      -- Sortie à la console pour information
   Gtk.Main.Main;                                -- Boucle principale
   -- ======================================================================
   -- Fin de la partie générique finale du programme principal (PARTIE III)
   -- ======================================================================

end clef_insee ;
```

Fichier XML de l'interface interface.glade

```
<?xml version="1.0" encoding="UTF-8"?>
<interface>
  <requires lib="gtk+" version="2.24"/>
  <!-- interface-naming-policy project-wide -->
  <object class="GtkTextBuffer" id="TamponTexte">
    <property name="text" translatable="yes">Informations</property>
  </object>
  <object class="GtkWindow" id="window1">
    <property name="can_focus">False</property>
    <property name="window_position">center</property>
    <property name="default_width">440</property>
    <property name="default_height">250</property>
    <signal name="show" handler="A_louverture" swapped="no"/>
    <signal name="configure-event" handler="Si_fenetre_bouge" swapped="no"/>
    <signal name="delete-event" handler="SI_fenetre_ppale_close" swapped="no"/>
    <child>
      <object class="GtkVBox" id="vbox1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <child>
          <object class="GtkLabel" id="Texte">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="xalign">0.20000000298023224</property>
            <property name="label" translatable="yes">Entrer les 13 chiffres du n° INSEE (NIR) dans les champs :</property>
            <attributes>
              <attribute name="underline" value="True"/>
            </attributes>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkHBox" id="hbox1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkEntry" id="Genre">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="is_focus">True</property>
                <property name="has_tooltip">True</property>
                <property name="tooltip_text" translatable="yes">Genre:
1 = masculin
2 = féminin</property>
                <property name="max_length">1</property>
                <property name="invisible_char">•</property>
                <property name="width_chars">0</property>
                <property name="text" translatable="yes">1</property>
                <property name="xalign">1</property>
                <property name="primary_icon_activatable">False</property>
                <property name="secondary_icon_activatable">False</property>
                <property name="primary_icon_sensitive">True</property>
                <property name="secondary_icon_sensitive">True</property>
                <signal name="changed" handler="Si_genre_change" swapped="no"/>
                <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="padding">10</property>
                <property name="position">0</property>
              </packing>
            </child>
            <child>
              <object class="GtkEntry" id="An">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="has_tooltip">True</property>
                <property name="tooltip_text" translatable="yes">Les deux derniers chiffres
de l'année de naissance</property>
                <property name="max_length">2</property>
                <property name="invisible_char">•</property>
                <property name="width_chars">0</property>
                <property name="text" translatable="yes">11</property>
                <property name="xalign">1</property>
                <property name="primary_icon_activatable">False</property>
                <property name="secondary_icon_activatable">False</property>
                <property name="primary_icon_sensitive">True</property>
                <property name="secondary_icon_sensitive">True</property>
                <signal name="changed" handler="Si_an_change" swapped="no"/>
                <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
```

```
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="padding">10</property>
                <property name="position">1</property>
              </packing>
            </child>
            <child>
              <object class="GtkEntry" id="Mois">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="has_tooltip">True</property>
                <property name="tooltip_text" translatable="yes">Les deux chiffres
du mois de naissance</property>
                <property name="max_length">2</property>
                <property name="invisible_char">•</property>
                <property name="width_chars">0</property>
                <property name="text" translatable="yes">11</property>
                <property name="xalign">1</property>
                <property name="primary_icon_activatable">False</property>
                <property name="secondary_icon_activatable">False</property>
                <property name="primary_icon_sensitive">True</property>
                <property name="secondary_icon_sensitive">True</property>
                <signal name="changed" handler="Si_mois_change" swapped="no"/>
                <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="padding">10</property>
                <property name="position">2</property>
              </packing>
            </child>
            <child>
              <object class="GtkEntry" id="Dept">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="has_tooltip">True</property>
                <property name="tooltip_text" translatable="yes">Le code à deux chiffres
du département de naissance
(ou du lieu de naissance hors métropole)</property>
                <property name="max_length">2</property>
                <property name="invisible_char">•</property>
                <property name="width_chars">0</property>
                <property name="text" translatable="yes">11</property>
                <property name="xalign">1</property>
                <property name="primary_icon_activatable">False</property>
                <property name="secondary_icon_activatable">False</property>
                <property name="primary_icon_sensitive">True</property>
                <property name="secondary_icon_sensitive">True</property>
                <signal name="changed" handler="Si_dept_change" swapped="no"/>
                <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="padding">10</property>
                <property name="position">3</property>
              </packing>
            </child>
            <child>
              <object class="GtkEntry" id="Commune">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="has_tooltip">True</property>
                <property name="tooltip_text" translatable="yes">Code à trois chiffres
de la commune de naissance
(ou code conventionnel hors métropole) </property>
                <property name="max_length">3</property>
                <property name="invisible_char">•</property>
                <property name="width_chars">0</property>
                <property name="text" translatable="yes">111</property>
                <property name="xalign">1</property>
                <property name="primary_icon_activatable">False</property>
                <property name="secondary_icon_activatable">False</property>
                <property name="primary_icon_sensitive">True</property>
                <property name="secondary_icon_sensitive">True</property>
                <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="padding">10</property>
                <property name="position">4</property>
              </packing>
```

interface.glade, page 2

```
          </child>
          <child>
            <object class="GtkEntry" id="Rang">
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="has_tooltip">True</property>
              <property name="tooltip_text" translatable="yes">Numéro d'ordre à trois chiffres</property>
              <property name="max_length">3</property>
              <property name="invisible_char">•</property>
              <property name="width_chars">0</property>
              <property name="text" translatable="yes">111</property>
              <property name="xalign">1</property>
              <property name="primary_icon_activatable">False</property>
              <property name="secondary_icon_activatable">False</property>
              <property name="primary_icon_sensitive">True</property>
              <property name="secondary_icon_sensitive">True</property>
              <signal name="focus-out-event" handler="Si_perd_le_focus" swapped="no"/>
            </object>
            <packing>
              <property name="expand">True</property>
              <property name="fill">True</property>
              <property name="padding">10</property>
              <property name="position">5</property>
            </packing>
          </child>
        </object>
        <packing>
          <property name="expand">True</property>
          <property name="fill">True</property>
          <property name="position">1</property>
        </packing>
      </child>
      <child>
        <object class="GtkHBox" id="hbox2">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <child>
            <object class="GtkFixed" id="fixed1">
              <property name="width_request">160</property>
              <property name="height_request">60</property>
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <child>
                <object class="GtkButton" id="Bouton_calculer">
                  <property name="label" translatable="yes">Calculer la clef</property>
                  <property name="width_request">140</property>
                  <property name="height_request">40</property>
                  <property name="visible">True</property>
                  <property name="can_focus">True</property>
                  <property name="receives_default">True</property>
                  <property name="use_action_appearance">False</property>
                  <signal name="clicked" handler="Si_clic_sur_calculer_la_clef" swapped="no"/>
                </object>
                <packing>
                  <property name="x">20</property>
                  <property name="y">20</property>
                </packing>
              </child>
            </object>
            <packing>
              <property name="expand">True</property>
              <property name="fill">True</property>
              <property name="position">0</property>
            </packing>
          </child>
          <child>
            <object class="GtkVBox" id="vbox2">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <child>
                <object class="GtkLabel" id="Transition">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="label" translatable="yes">La clef vaut :</property>
                  <property name="justify">center</property>
                </object>
                <packing>
                  <property name="expand">True</property>
                  <property name="fill">True</property>
                  <property name="position">0</property>
                </packing>
              </child>
              <child>
                <object class="GtkFixed" id="FondClef">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
```

interface.glade, page 3

```
                    <child>
                      <object class="GtkEntry" id="ValeurClef">
                        <property name="width_request">80</property>
                        <property name="height_request">40</property>
                        <property name="visible">True</property>
                        <property name="can_focus">True</property>
                        <property name="has_tooltip">True</property>
                        <property name="tooltip_text" translatable="yes">Valeur de la clef de contrôle
(complément à 97 du reste
de la division du NIR par 97)</property>
                        <property name="editable">False</property>
                        <property name="max_length">2</property>
                        <property name="invisible_char">•</property>
                        <property name="width_chars">0</property>
                        <property name="text" translatable="yes">##</property>
                        <property name="xalign">0.5</property>
                        <property name="primary_icon_activatable">False</property>
                        <property name="secondary_icon_activatable">False</property>
                        <property name="primary_icon_sensitive">True</property>
                        <property name="secondary_icon_sensitive">True</property>
                      </object>
                      <packing>
                        <property name="x">67</property>
                      </packing>
                    </child>
                  </object>
                  <packing>
                    <property name="expand">True</property>
                    <property name="fill">True</property>
                    <property name="position">1</property>
                  </packing>
                </child>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkTextView" id="BoiteMessage">
            <property name="height_request">35</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="pixels_above_lines">1</property>
            <property name="pixels_below_lines">1</property>
            <property name="editable">False</property>
            <property name="wrap_mode">word</property>
            <property name="justification">center</property>
            <property name="left_margin">1</property>
            <property name="right_margin">1</property>
            <property name="buffer">TamponTexte</property>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="padding">1</property>
            <property name="position">3</property>
          </packing>
        </child>
      </object>
    </child>
  </object>
</interface>
```

interface.glade, page 4