Projet NF05 A13 :

Plateforme à interface graphique pour le calcul matriciel

Oriane Bouvier, Camille Benhaim, Louis Kleijn.

Introduction :

L'UV NF05 a pour but d'initier les élèves au langage C. Grâce aux cours et aux TP, nous avons appris et mis en pratique les bases de ce langage. Mais pour acquérir plus d'expérience et de connaissance, nous devons dans le cadre d'un projet, coder un programme beaucoup plus complet que ce que nous avons pu voir en TP ou en cours. Cette année, nous devons coder un programme qui permet, grâce à une interface graphique, d'effectuer des opérations sur des matrices de tailles définies par l'utilisateur. Ce projet nous permet entre autres d'améliorer la qualité de nos recherches sur Internet et/ou les ouvrages dédiés à l'informatique mais aussi de nous familiariser avec le langage C et la bibliothèque que nous avons choisi qui permet de créer une interface graphique autre que la console que nous avons utiliser pour les cours et les TP.

Comment avons-nous procéder, en tant que débutant en langage C, pour créer cette interface ?

Pour répondre à cela, nous allons en premier lieu décrire les algorithmes qui nous on permit de faire les différentes opérations demandées.
Ensuite, nous allons donner le mode d'emploi du programme et décrire les problèmes majeurs auxquels nous avons été confrontés.

I. Algorithmes utilisés

1. Choix de la bibliothèque.

Nous avons choisi la bibliothèque GTK+ pour coder notre programme. Elle n'est pas compliquée à apprendre, gratuite, assez intuitive et est très bien expliquée sur Internet.

Cette bibliothèque permet de coder une interface graphique en utilisant une méthode différente du codage vu en cours. En effet, lors d'une compilation d'un code simple, le compilateur suit les instructions en commençant par la première ligne et en terminant par la dernière en utilisant les fonctions appelées dans l'algorithme. Dans le cas de la bibliothèque GTK+, On utilise une boucle événementielle qui fait en sorte d'exécuter le programme jusqu'à ce que l'utilisateur choisisse de le fermer. On utilise une fonction principale « main » qui va appeler des autres fonctions selon le type d'événement, comme un clic sur un bouton ou un texte que l'on rentre … Cette fonction main permet de faire fonctionner le programme et permet d'effectuer d'autres actions qui vont elles mêmes en exécuter d'autres ou même s'appeler les une les autres. Ce type de codage se rapproche plus du langage orienté objet, et a été adapté dans beaucoup d'autres langages informatiques (C++, Python, PHP par exemple). Cette bibliothèque est connue pour être à l'origine de l'environnement GNOME (GNU Network Object Model Environment) utilisé sur Linux, un système d'exploitation libre.

2. Description des fonctions utilisées.

Pour le projet, nous avons choisi de donner à notre fonction « main » l'apparence d'une fenêtre qui a un rôle de menu principal. A partir de cette fenêtre, nous avons créé une barre des taches, comme sur la plupart des logiciels, décomposée en 2 parties :

- Un onglet « Opérations sur 2 matrices » qui regroupe l'addition, la soustraction et la multiplication de 2 matrices, ainsi que la résolution d'un système linéaire de la forme Ax =b. Cet onglet contient aussi le choix « Quitter » qui permet de quitter le programme.
- Un onglet « Opérations sur 1 matrice » qui regroupe l'inverse, la puissance, la trace, le déterminant ainsi que la possibilité d'enregistrer une matrice dans un fichier texte (format .txt) pour ensuite éventuellement l'imprimer.

Cet onglet contient aussi le choix « Quitter » qui permet de quitter le programme.
Sur le reste de la fenêtre est affiché un texte qui donne la démarche à suivre pour la première étape ainsi que le nom des membres du groupe.

Lorsque l'on clique sur un des 2 onglets, une liste de choix parmi les opérations présentées précédement apparaît. Lors du clic sur une de ces fonctions, un signal est envoyé à une autre fonction qui va s'activer. Nous avons choisi d'afficher, quelque soit le choix, une nouvelle fenêtre demandant les paramètres nécessaires à l'opération à effectuer.

La plupart du temps, il s'agit simplement de donner le nombre de colonne(s) et de ligne(s) de(s) matrice(s). On demande ces instructions grâce à des fonctions de la bibliothèque GTK+ qui affichent une zone de texte où l'on peut uniquement entrer un réel entre des bornes que nous avons choisi et qui récupèrent ensuite ces valeurs pour effectuer les opérations et pour donner la bonne dimension aux matrices.

Une fois que ces informations sont saisies, l'utilisateur clique sur le bouton contenu dans la fenêtre, que nous avons ajouté, et appelle ainsi une fonction qui permet de saisir les coefficients de la matrice en fonction des valeurs entrées précédemment. On utilise le même procédé que pour la taille des matrices :
On utilise une fonction de la bibliothèque qui permet d'insérer des tables en 2 dimensions dans la fenêtre. On utilise donc cette fonction pour insérer des tables de la dimension saisie précédemment par l'utilisateur. Une fois que ces informations sont saisies, l'utilisateur clique sur le bouton contenu dans la fenêtre, que nous avons ajouté, et appelle la fonction effectuant le calcul et qui ensuite affiche le résultat à l'écran. Chaque fonction à sa propre manière de calculer le résultat, nous allons donc les décrire une par une :

- Addition/Soustraction : Une des opérations les plus simples à faire. On utilise 2 boucles « for » afin de se déplacer dans la table et on ajoute/soustrait case par case. De la même façon, on créé 2 boucles « for » afin d'afficher le résultat sous forme de texte case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner le résultat sous la forme d'une matrice.
- Multiplication : Cette fois-ci, on utilise 3 boucles « for » . Sur le même principe que précédemment, on va calculer le résultat case par case. La 3ème boucle « for » sert à se déplacer sur la ligne de la matrice A et sur la colonne de la matrice B du produit A*B et

ajouter à chaque fois le valeur de chaque produit de case pour trouver la valeur finale de la case. On créé ensuite 2 boucles « for » afin d'afficher le résultat sous forme de texte case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner le résultat sous la forme d'une matrice.

- Résolution de système linéaire : On a demandé à l'utilisateur de saisir la matrice et le vecteur. On récupère les valeurs et on va résoudre le système obtenu avec l'équation Ax = B grâce à la méthode de Gauss-Jordan. Cette méthode est très pratique car elle demande peut de mémoire et le programme met donc peut de temps à calculer les solutions. Avant de procéder au calcul, on vérifie que le déterminant est bien différent de 0. Voici comment cette méthode fonctionne pour une matrice 3*3. Cet algorithme fonctionne pour toute matrice carrée mais il est plus simple de l'expliquer sur une matrice de cette dimension. Voici donc la matrice :

$$\begin{pmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \\ A_{1,3} & A_{2,3} & A_{3,3} \end{pmatrix} \qquad \begin{pmatrix} B_1 \\ B_2 \\ B_3 \end{pmatrix}$$

On va choisir la valeur $A_{1,1}$ comme pivot. On divise toute la ligne du pivot par le pivot (les valeurs de B sont comprises dans les lignes). Ensuite, on ajoute à la 2ème ligne un multiple de la 1ème ligne par la 2ème donc $A_{1,1}*A_{1,2}$. On ajoute donc de la même façon $A_{1,1}*A_{1,3}$ à la 3ème ligne. On fera de même pour toutes les autres lignes si la matrice est plus grande. Après cette étape on change de pivot. On prend $A_{2,2}$. De la même façon, on va diviser toute la ligne par ce pivot puis ajouter aux autres lignes la multiplication de ce pivot par les valeurs comprises dans la colonne du pivot.
On répète cette étape autant de fois qu'il y a de pivots et à la fin, la matrice A est devenue la matrice identité et les solutions du système sont la où étaient initialement les valeurs du vecteur. On créé ensuite 1 boucle « for » afin d'afficher le résultat sous forme de texte case par case, en séparant les zones de textes par des retours à la ligne pour donner les solutions sous la forme d'un vecteur.

- Inverse d'une matrice : On va utiliser la même méthode que précédemment, sauf que le vecteur B sera au départ remplacé par la matrice identité de même dimension que la matrice

A. On vérifie que la matrice est inversible en calculant son déterminant qui doit être différent de 0. Dans le cas où il est égal à 0, on affiche une phrase annonçant que l'inversion n'est pas possible dans la fenêtre donnant normalement le résultant. En utilisant les mêmes étapes, la matrice est au final la où était la matrice identité et la matrice A est devenue la matrice identité. On créé ensuite 2 boucles « for » afin d'afficher le résultat sous forme de texte case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner le résultat sous la forme d'une matrice. Cette méthode est très pratique car simple, elle ne requiert que peu de mémoire contrairement au calcul de la transposée de la comatrice par récursivité.

- Puissance de matrice : On va utiliser pratiquement le même algorithme que pour la multiplication de 2 matrice. On rajoutera une boucle « for » au départ de l'algorithme qui va effectuer l'opération autant de fois que la puissance saisie le demande. L'autre changement est que l'on va utiliser une table intermédiaire qui permettra de stocker la matrice engendra par la multiplication précédente. Par exemple, pour une puissance 3, on calcule d'abord le carré de la matrice que l'on stocke dans la table intermédiaire, qui ensuite va être multipliée par la matrice initiale. On créé ensuite 2 boucles « for » afin d'afficher le résultat avec des texte case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner le résultat sous la forme d'une matrice.
- Trace d'une matrice : La fonction la plus simple à coder. En effet, il suffit d'additionner les cases de la diagonales de la matrice. On utilise pour cela 2 boucles « for » qui additionne les cases dont le numéro de la colonne est égal au numéro de la ligne. On utilise ensuite un label pour afficher le texte contenant le résultat.
- Déterminant d'une matrice : On va utiliser pratiquement la même technique que pour la résolution du système Ax = B ou l'inverse d'une matrice, sauf que dans ce cas qui est plus simple, on n'utilise qu'un seul pivot. On va trianguliariser la matrice grâce à la méthode du pivot de Gauss. On va chercher la plus grande valeur en valeur absolue de la première colonne et on va échanger la ligne correspondante avec la première ligne. Pour chaque échange de ligne, le déterminant prendra un signe « - ». On sélectionne une ligne, à laquelle on ajoute à chaque terme le multiplicateur (valeur de la case dans la ligne qui est dans la colonne du pivot)*(Valeur de la case située dans la ligne du pivot et dans la colonne du terme qui sera modifié). On effectue cette opération pour chaque ligne, ce qui permet d'obtenir des 0 sous la diagonale. A partir de la, le déterminant est égal au produit des termes de la diagonale. Cette méthode est plus pratique que celle utilisée en cours qui demande de

créer des matrices temporaires et d'utiliser la récursivité, ce qui engendre une énorme consommation de mémoire. Les calculs sont alors très longs contrairement à la méthode que nous avons choisi. On utilise ensuite un label pour afficher le texte contenant le résultat.

- Enregistrer une matrice : On récupère les valeurs de la matrice grâce à 2 boucles « for », on ouvre ensuite une nouvelle fenêtre qui demande le nom du fichier à ouvrir ou créer s'il ne l'est pas déjà. On ouvre ensuite le fichier, on rajoute l'extension .txt pour pouvoir imprimer grâce à un simple éditeur de texte et entre les valeurs avec des textes case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner les valeurs sous la forme d'une matrice.

Lorsque le résultat est affiché, quelque soit l'opération, il y a toujours au moins 2 options proposées. On peut, en cliquant sur le bouton « Retour au menu principal », utiliser une fonction qui va réafficher le main et détruire les autres fenêtres. On peut aussi cliquer sur le bouton « Quitter », qui affiche une boite de dialogue demandant à l'utilisateur s'il veut quitter le programme. Si oui, le programme se ferme, sinon, la boite de dialogue se ferme. De plus, pour toutes les fonctions qui affichent une matrice comme résultat ont un autre bouton permettant d'enregistrer la matrice. Lors du clic sur ce bouton, une fenêtre va apparaître et l'utilisateur devra entrer le nom du fichier à créer ou ouvrir s'il est déjà créé. Ensuite cette fenêtre se ferme et on rentre les valeurs avec des textes case par case, en séparant les zones de textes par des espaces ou des retours à la ligne pour donner les valeurs sous la forme d'une matrice.

II. Mode d'emploi et difficultés rencontrées.

1. Mode d'emploi.

Le programme que nous avons codé est assez intuitif. Il suffit de lancer l'application et suivre les instructions décrites sur le programme. La première chose à faire est de cliquer sur un des menus de la barre des tâches pour ensuite sélectionner l'opération à effectuer. Ensuite, quelque soit l'opération à effectuer, il faudra saisir la taille de(s) la matrice(s) et dans certains cas il faudra saisir une autre donnée. Une fois les valeurs entrées, cliquer sur le bouton de la fenêtre permettant d'activer la fonction suivante qui va permettre à l'utilisateur de rentrer les valeurs. Nous avons mis -300 et +300 comme borne, avec plusieurs chiffres après la virgule. Nous avons fait cela car en général on n'utilise pas des valeurs aussi hautes ou basses, mais une simple modification dans la fonction permet de changer ces bornes. De même pour les matrices, nous avons limité la taille à 100*100, car de toute façon à partir d'un certain moment il n'y a pas assez de place sur l'écran. On peut saisir les valeurs au clavier ou cliquer sur les flèches à coté de chaque case pour faire varier les valeurs. Une fois les valeurs rentrées, cliquer sur le bouton dans la fenêtre et le résultat sera affiché. L'utilisateur pourra enregistrer une matrice dans un fichier, quitter le programme ou revenir au menu principal. Le programme ne se fermera pas tant que l'utilisateur n'aura pas choisi de le quitter.

2. Difficultés rencontrées

Les plus grosses difficultés que nous avons rencontré étaient d'utiliser correctement toutes les fonctions, et notamment les paramètres de ces fonctions « CALLBACK » appelées lors d'un clic sur un bouton par exemple. Les algorithmes sur lesquels nous avons passé le plus de temps sont le calcul du déterminant par triangularisation car la méthode employée est une méthode que nous avons beaucoup moins étudié en MTX3, nous avons donc dû apprendre cette méthode par nous même. De la même façon, l'élimination de Gauss-Jordan pour calculer l'inverse d'une matrice et pour résoudre un système linéaire est totalement différente de celle employée en cours, qui demanderait énormément de mémoire pour fonctionner correctement.

CONCLUSION

Pour coder ce programme correctement, nous avons dû apprendre complètement une bibliothèque permettant de coder une interface graphique qui fonctionne très différemment de ce que nous avons fait en cours et en TP. Cela a été une expérience nouvelle qui nous a permis de grandement améliorer notre adaptation à un nouveau langage, à la recherche sur Internet. Nous nous sommes aussi rendus compte que coder, ne serait-ce qu'un petit programme, demande une bonne organisation des algorithmes et énormément de temps. De plus, les méthodes utilisées en cours sont peut-être plus simple à comprendre et à appliquer sur feuille, mais sont beaucoup plus compliquées à coder en C et ne sont pas forcément les plus pratiques et rapide d’exécution. C'est ce que nous avons remarqué notamment sur l'inversion d'une matrice.

ANNEXE

- Bibliographie :

Source principale pour l'apprentissage de la bibliothèque GTK+ :

http://gtk.developpez.com/cours/gtk2/

Source principale pour le codage du calcul du déterminant :

http://fr.wikipedia.org/wiki/Calcul_du_d%C3%A9terminant_d%27une_matrice#M.C3.A9thode_du_pivot_de_Gauss

Source principale pour le codage de l'inversion d'une matrice :

http://fr.wikipedia.org/wiki/%C3%89limination_de_Gauss-Jordan

- Code commenté :
  - fichier main :

```
#include <stdlib.h>
#include <math.h>
#include <gtk/gtk.h>
#include "projet.h"

int main(int argc, char **argv)
{
   Structure prog;
   gtk_init(&argc, &argv);

   /* Création de la fenêtre */
   prog.pWindowMain = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog.pWindowMain), "Projet NF05 P13");
   gtk_window_set_default_size(GTK_WINDOW(prog.pWindowMain), 320, 200);
   g_signal_connect(G_OBJECT(prog.pWindowMain), "destroy", G_CALLBACK(gtk_main_quit),
NULL);

   /* Création de la Box qui contiendra les elements */
   prog.pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog.pWindowMain), prog.pVBox);

   /* Création du Texte */
   prog.pLabel = gtk_label_new("Bonjour!\n Pour commencer, veuillez selectionner une operation a
effectuer.\n\nCe programme a ete code par:\n-Oriane Bouvier\n-Camille Benhaim\n-Louis Kleijn.");

   /* Création du menu */

   /* On connecte les differents objets du menu aux fonctions qui leur sont associés grace a la
fonction g_signal_connect */
   /* Premier menu */
   prog.pMenuBar = gtk_menu_bar_new();
   prog.pMenu = gtk_menu_new();
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Addition de 2 matrices");
   g_signal_connect(G_OBJECT(prog.pMenuItem1), "activate",
G_CALLBACK(Taille_Matrice_Addition), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem1);
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Soustraction de 2 matrices");
   g_signal_connect(G_OBJECT(prog.pMenuItem1), "activate",
G_CALLBACK(Taille_Matrice_Soustraction), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem1);
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Multiplication de 2 matrices");
   g_signal_connect(G_OBJECT(prog.pMenuItem1), "activate",
G_CALLBACK(Taille_Matrice_Multiplication), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem1);
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Resolution de systeme lineaire");
   g_signal_connect(G_OBJECT(prog.pMenuItem1), "activate",
```

```
G_CALLBACK(Taille_Matrice_Systeme), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem1);
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Quitter");
   g_signal_connect(G_OBJECT(prog.pMenuItem1), "activate", G_CALLBACK(OnQuitter),
(GtkWidget*) prog.pWindowMain);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem1);
   prog.pMenuItem1 = gtk_menu_item_new_with_label("Operations sur 2 matrices");


   gtk_menu_item_set_submenu(GTK_MENU_ITEM(prog.pMenuItem1), prog.pMenu);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenuBar), prog.pMenuItem1);

   /* Second menu */

   prog.pMenu = gtk_menu_new();
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Inverse d'une matrice");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate",
G_CALLBACK(Taille_Matrice_Inverse), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Puissance de matrice");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate",
G_CALLBACK(Taille_Matrice_Puissance), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Trace d'une matrice");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate",
G_CALLBACK(Taille_Matrice_Trace), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Determinant d'une matrice");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate",
G_CALLBACK(Taille_Matrice_Determinant), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Enregistrer une matrice");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate",
G_CALLBACK(Taille_Matrice_Enregistrer), &prog);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Quitter");
   g_signal_connect(G_OBJECT(prog.pMenuItem2), "activate", G_CALLBACK(OnQuitter),
(GtkWidget*) prog.pWindowMain);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenu), prog.pMenuItem2);
   prog.pMenuItem2 = gtk_menu_item_new_with_label("Operations sur 1 matrice");

   gtk_menu_item_set_submenu(GTK_MENU_ITEM(prog.pMenuItem2), prog.pMenu);
   gtk_menu_shell_append(GTK_MENU_SHELL(prog.pMenuBar), prog.pMenuItem2);

   /* Ajout du menu a la fenetre */
   gtk_box_pack_start(GTK_BOX(prog.pVBox), prog.pMenuBar, FALSE, TRUE, 0);


   gtk_box_pack_start(GTK_BOX(prog.pVBox), prog.pLabel, TRUE, FALSE, 0);
```

```
   gtk_widget_show_all(prog.pWindowMain);

   gtk_main(); /*Cette fonction permet de lancer la boucle évenementielle qui permet de faire
tourner le programme "à l'infini" */

   return EXIT_SUCCESS;
}
   /* Fonction qui permet de quitter le programme */

void OnQuitter(GtkWidget* widget, gpointer data)
{
   GtkWidget *pQuestion;
   pQuestion = gtk_message_dialog_new(GTK_WINDOW(data),
      GTK_DIALOG_MODAL,
      GTK_MESSAGE_QUESTION,
      GTK_BUTTONS_YES_NO,
      "Voulez vous vraiment\n"
      "quitter le programme?");

   switch(gtk_dialog_run(GTK_DIALOG(pQuestion)))
   {
      case GTK_RESPONSE_YES:
         gtk_main_quit();
         break;
      case GTK_RESPONSE_NONE:
      case GTK_RESPONSE_NO:
         gtk_widget_destroy(pQuestion);
         break;
   }
}

/* Fonction appelée lors du clic sur "Retourner au menu principal" */

void Retour(GtkWidget* widget, gpointer data)
{
   Structure *prog=(Structure *)data;  /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* détruit toutes les fenetres sauf le main et permet de revenir a la fenetre principale */

   gtk_widget_destroy(prog->pWindowResultat);
   gtk_widget_destroy(prog->pWindowDonnees);
   gtk_widget_destroy(prog->pWindowTaille);
   gtk_widget_show_all(prog->pWindowMain);
   gtk_main();
}

/* Fonction executee lors du signal "clicked" sur "Addition de 2 matrices*/

void Taille_Matrice_Addition(GtkWidget *widget, gpointer data)
```

```
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille des matrices");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes des matrices a
additionner");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes des matrices a additionner");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Addition),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction executée lors du clic sur "Ok!" */

void Addition(GtkWidget *widget,gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Récupération du nombre de lignes et colonnes */
```

```
   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création d'une fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création de 2 tables permettant de saisir les differentes valeurs de la matrice. On insere le
signe + entre les 2 */
   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   prog->pTable2=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   prog->pLabel = gtk_label_new("+");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable2, TRUE, FALSE, 0);

   /* On utilise une boucle for pour créer une case en fonction du nombre de lignes et colonnes
qu'on insere dans la Table */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
         prog->Case2[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case2[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable2), prog->Case2[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);

      }
   }

   /* On insere un bouton qui va appeler la fonction donnant le resultat */

   prog->Button = gtk_button_new_with_label("Additionner!");
```

```
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Resultat_Addition),
prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction donnant le résultat de l'addition. Appelée lors du clic sur le bouton "Additionner!" */

void Resultat_Addition(GtkWidget *widget,gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une nouvelle fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création d'une nouvelle box */
   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   /* On crée une table qui affichera la matrice résultat */

   prog->pTable_Result=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);

   /* On utilise une boucle for pour calculer la somme des matrices case par case */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
         prog->Valeur2[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case2[prog->i][prog->j]));
         prog->Valeur_Resultat[prog->i][prog->j]=prog->Valeur1[prog->i][prog->j]+prog-
>Valeur2[prog->i][prog->j];
         prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
         prog->pLabel = gtk_label_new(prog->Text);
         gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel,prog->i-1, prog->i,
prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
         g_free(prog->Text);
      }
   }
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);
```

```
   /* Création d'une box contenant 3 boutons */

   prog->pHBox=gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   /* Création des boutons */

   prog->Button = gtk_button_new_with_label("Enregistrer dans un fichier .txt");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_dialogue_fichier_addition_soustraction),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Retourner au menu principal");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Quitter");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowResultat);


}

/* Pour la soustraction, la procédure est exactement la même sauf que l'on soustraie case par case */

void Taille_Matrice_Soustraction(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille des matrices");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);
   gtk_widget_show_all(prog->pWindowTaille);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* On crée les différents éléments de zones de textes et textes puis on les insere dans la box selon
l'affichage que l'on souhaite */
```

```
   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes des matrices a
soustraire");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes des matrices a soustraire");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   /* Bouton qui appelle la fonction qui permet de saisir les valeurs */

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Soustraction),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fontion appelée lors du clic sur "Ok!" de la fonction précédente */

void Soustraction(GtkWidget* widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupère les valeurs des lignes et colonnes pour créer les tables avec les bonnes dimensions
*/

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création et insertion de la box dans la fenetre */
```

```
   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création des tables en fonction des valeurs précédentes et insertion dans la box */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   prog->pTable2=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   prog->pLabel = gtk_label_new("-");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable2, TRUE, FALSE, 0);

   /* on créé 2 tables de même dimensions grace à 2 boucles for */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
         prog->Case2[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case2[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable2), prog->Case2[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);

      }
   }

   /* Création du bouton qui va appeler la fonction donnant le résultat */

   prog->Button = gtk_button_new_with_label("Soustraire!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Resultat_Soustraction), prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée par le clic sur "Soustraire!" */

void Resultat_Soustraction(GtkWidget* widget, gpointer data)
{
   Structure *prog=(Structure *)data;  /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une nouvelle fenêtre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
```

```
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   /* Création de la table résultat */

   prog->pTable_Result=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);

   /* On soustrait case par case grâce à 2 boucles for et on affiche le résultat dans la table résultat */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
         prog->Valeur2[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case2[prog->i][prog->j]));
         prog->Valeur_Resultat[prog->i][prog->j]=prog->Valeur1[prog->i][prog->j]-prog-
>Valeur2[prog->i][prog->j];
         prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
         prog->pLabel = gtk_label_new(prog->Text);
         gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel,prog->i-1, prog->i,
prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
         g_free(prog->Text);
      }
   }
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);

   /* Création d'une box contenant 3 boutons */

   prog->pHBox=gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   /* Création des boutons */

   prog->Button = gtk_button_new_with_label("Enregistrer dans un fichier .txt"); //Permet
d'appeler la fonction qui enregistre dans un fichier
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_dialogue_fichier_addition_soustraction),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Retourner au menu principal");// Permet de revenir
au menu principal
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
```

```
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Quitter"); // Ouvre une boite de dialogue pour
quitter le programme
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
   gtk_widget_show_all(prog->pWindowResultat);
}

/* Fonction appelée lors du clic sur "Enregistrer dans un fichier .txt" */

void Boite_dialogue_fichier_addition_soustraction(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowEnregistrer = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowEnregistrer), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowEnregistrer), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowEnregistrer), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowEnregistrer), prog->pVBox);

   /* Création des différents éléments permettant de saisir le nom du fichier */

   prog->pLabel= gtk_label_new("Entrer le nom de votre fichier");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

   prog->Entry= gtk_entry_new();
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Entry, TRUE, FALSE, 0);

   /* Création du bouton appelant la fonction qui va créer le fichier et rentrer les valeurs dans celui-
ci */

   prog->Button= gtk_button_new_with_label("Ok!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Enregistrer_Resultat_Addition_Soustraction),prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowEnregistrer);
}

/* Fonction appelée lors du clic sur "Ok!" */
```

```
void Enregistrer_Resultat_Addition_Soustraction(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Récupération du texte. On y ajoute l'extention .txt et on l'ouvre */

   prog->Text = gtk_entry_get_text(prog->Entry);
   prog->Text = g_strdup_printf("%s.txt", prog->Text);
   prog->fichier = fopen(prog->Text, "w+");

   /* On utilise 2 boucles for pour rentrer chaque valeur de la matrice */

   for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {
         prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
         fprintf(prog->fichier, "%.2f ", prog->Valeur_Resultat[prog->i][prog->j]);
      }
      fprintf(prog->fichier, "\n");
   }

   /* On ferme le fichier et la fenetre. On revient a la fenetre précédente et le fichier est enregistré !
*/

   fclose(prog->fichier);
   gtk_widget_destroy(prog->pWindowEnregistrer);
   gtk_main();
}

/* Fonction demandant la taille des 2 matrices à multiplier. Appelée depuis le menu principal */

void Taille_Matrice_Multiplication(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille des matrices");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
```

```
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la premiere
matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne1 = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne1, TRUE,
FALSE, 0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la premiere matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne1 = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne1, TRUE, FALSE,
0);

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la deuxieme
matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne2 = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne2, TRUE,
FALSE, 0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la deuxieme matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne2 = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne2, TRUE, FALSE,
0);

   /* On demande la dimension commune qui est nécessaire, la multiplication n'est pas possible
sinon ! Normalement, le nombre de colonnes de la premiere doit etre le même
   que le nombre de lignes de la deuxième */

   prog->pLabel = gtk_label_new("Entrer la dimension commune aux matrices a multiplier. Il est
necessaire qu'il y en ai une, la multiplication n'est pas possible sinon!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

   prog->Entry_Dim = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Dim, TRUE, FALSE, 0);

   /* Création du bouton appelant la fonction permettant de rentrer les valeurs */

   prog->Button = gtk_button_new_with_label("Ok!");
```

```
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Multiplication),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction précédente */

void Multiplication(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupére toutes les valeurs saisies précédemment */

   prog->Nbr_Col1=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne1));
   prog->Nbr_Lig1=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne1));
   prog->Nbr_Col2=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne2));
   prog->Nbr_Lig2=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne2));
   prog->Dim=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog->Entry_Dim));


   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   prog->pHBox = gtk_hbox_new(FALSE, 0);


   if(prog->Nbr_Col1 != prog->Nbr_Lig2)
   {
      prog->pLabel = gtk_label_new("La multiplication de ces 2 matrices ne peut pas etre faite, les
dimensions ne sont pas correctes.");
      gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);
      prog->Button = gtk_button_new_with_label("Resaisir les dimensions");
      g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Taille_Matrice_Multiplication), prog);
```

```
        gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
        prog->Button = gtk_button_new_with_label("Revenir au menu principal!");
        g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour), prog);
        gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
        prog->Button = gtk_button_new_with_label("Quitter");
        g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowDonnees);
        gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
        gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

    }
    else
    {
       /* Grace aux valeurs, on créé 2 tables composées de boutons permettant de saisir des valeurs */

    prog->pTable1=gtk_table_new(prog->Nbr_Lig1,prog->Nbr_Col1, TRUE);
    prog->pTable2=gtk_table_new(prog->Nbr_Lig2,prog->Nbr_Col2, TRUE);
    prog->pLabel = gtk_label_new("*");
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable2, TRUE, FALSE, 0);

    /* On utilise 2 boucles for pour créer la premiere table */

    for(prog->i=1; prog->i<=prog->Nbr_Lig1;prog->i++)
    {
       for(prog->j=1; prog->j<=prog->Nbr_Col1; prog->j++)
       {
          prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
          gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
          gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j], prog->j-1,
prog->j,prog->i-1, prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
       }
    }

    /* On utilise 2 boucles for pour créer la seconde table */

    for(prog->i=1; prog->i<=prog->Nbr_Lig2;prog->i++)
    {
       for(prog->j=1; prog->j<=prog->Nbr_Col2; prog->j++)
       {
          prog->Case2[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
          gtk_spin_button_set_value(prog->Case2[prog->i][prog->j], 0);
          gtk_table_attach(GTK_TABLE(prog->pTable2), prog->Case2[prog->i][prog->j], prog->j-1,
prog->j,prog->i-1, prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
       }
    }

    /* On créé un bouton qui va appeler la fonction donnant le résultat */
```

```
   prog->Button = gtk_button_new_with_label("Multiplier!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Resultat_Multiplication), prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);}
   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée lors du clic sur "Multiplier!" */

void Resultat_Multiplication(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   /* On créé la table qui donne le résultat */

   prog->pTable_Result=gtk_table_new(prog->Nbr_Lig1,prog->Nbr_Col2, TRUE);

   /* On utilise 2 boucles for pour initialiser les valeurs à 0 */

   for(prog->i=1; prog->i<=prog->Nbr_Lig1;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col2; prog->j++)
      {
         prog->Valeur_Resultat[prog->i][prog->j]=0;
      }
   }

   /* On utilise 3 boucles for pour multiplier les valeurs des lignes et colonnes des matrices et on
additionne ces valeurs entre elle pour obtenir la valeur d'une case.
   On repete l'opération pour chaque case. */

   for(prog->i=1; prog->i<=prog->Nbr_Lig1;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col2; prog->j++)
      {
         for (prog->k=1; prog->k<=prog->Dim; prog->k++)
```

```
            {
               prog->Valeur1[prog->i][prog-
>k]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->k]));
               prog->Valeur2[prog->k][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case2[prog->k][prog->j]));
               prog->Valeur_Resultat[prog->i][prog->j]=(prog->Valeur1[prog->i][prog->k]*prog-
>Valeur2[prog->k][prog->j])+prog->Valeur_Resultat[prog->i][prog->j];
            }
            prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
            prog->pLabel = gtk_label_new(prog->Text);
            gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel, prog->j-1, prog-
>j,prog->i-1, prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
            g_free(prog->Text);
         }
      }
      gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);

      /* Création d'une box contenant 3 boutons */

      prog->pHBox=gtk_hbox_new(FALSE, 0);
      gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

      /* Création des boutons */

      prog->Button = gtk_button_new_with_label("Enregistrer dans un fichier .txt"); //Appelle la
fonction permettant d'enregistrer dans un fichier.
      g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_dialogue_fichier_multiplication),prog);
      gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

      prog->Button = gtk_button_new_with_label("Retourner au menu principal");// Appelle la
fonction Retour
      g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
      gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

      prog->Button = gtk_button_new_with_label("Quitter");// Permet de quitter le programme en
ouvrant une boite de dialoque
      g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
      gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
      gtk_widget_show_all(prog->pWindowResultat);
}

/* Fonction appelée lors du clic sur "Enregistrer dans un fichier .txt" */

void Boite_dialogue_fichier_multiplication(GtkWidget *widget, gpointer data)
{
      Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */
```

```
   /* Création d'une nouvelle fenetre */

   prog->pWindowEnregistrer = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowEnregistrer), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowEnregistrer), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowEnregistrer), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création d'une nouvelle box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowEnregistrer), prog->pVBox);

   /* Création d'éléments permettant de saisir le nom du fichier */

   prog->pLabel= gtk_label_new("Entrer le nom de votre fichier");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

   prog->Entry= gtk_entry_new();
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Entry, TRUE, FALSE, 0);

   /* Création d'un bouton appelant la fonction qui va ouvrir/créer le fichier saisi et rentrer les
valeurs dans le fichier */

   prog->Button= gtk_button_new_with_label("Ok!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Enregistrer_Resultat_Multiplication),prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowEnregistrer);
}

/* Fonction enregistrant une matrice résultant d'une multiplication dans un fichier */

void Enregistrer_Resultat_Multiplication(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupére le texte, on ajoute ".txt" et on ouvre/créé le fichier correspondant */

   prog->Text = gtk_entry_get_text(prog->Entry);
   prog->Text = g_strdup_printf("%s.txt", prog->Text);
   prog->fichier = fopen(prog->Text, "w+");

   /* On rentre chaque valeur grâce à 2 boucles for */

   for(prog->i=1; prog->i<=prog->Nbr_Lig1;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col2; prog->j++)
```

```
        {
           prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
           fprintf(prog->fichier, "%.2f ", prog->Valeur_Resultat[prog->i][prog->j]);
        }
        fprintf(prog->fichier, "\n");
    }

    /* On ferme le fichier et la fenetre et le fichier est enregistré ! */

    fclose(prog->fichier);
    gtk_widget_destroy(prog->pWindowEnregistrer);
    gtk_main();
}

/* Fonction appelée depuis le menu principal permettant de saisir la taille de la matrice dont on va
calculer la trace */

void Taille_Matrice_Trace(GtkWidget *widget, gpointer data)
{
    Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

    /* Création d'une fenetre */

    prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille des matrices");
    gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
    g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

    /* Création de la box contenant les differents elements */

    prog->pVBox = gtk_vbox_new(FALSE, 0);
    gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

    /* Création des zones de textes et des textes et insertion dans la box */

    prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

    prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


    prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

    prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
```

```
0);

   /* Bouton appelant la fonction permettant de saisir les valeurs de la matrice */

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Trace),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" */

void Trace(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupére les valeurs et on créé une table avec celles-ci */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création de la table grâce aux valeurs récupérées */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);

   /* On utilise 2 boucles for pour créer la table */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
```

```
            gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
            gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
        }
    }

    /* Création du bouton qui va permettre d'afficher le résultat */

    prog->Button = gtk_button_new_with_label("Calculer la Trace!");
    g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Resultat_Trace),
prog);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
    gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée lors du clic de "Calculer la Trace!" */

void Resultat_Trace(GtkWidget *widget, gpointer data)
{
    Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

    /* Création d'une nouvelle fenetre */

    prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
    gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
    g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

    /* Création d'une nouvelle box */
    prog->pVBox = gtk_vbox_new(FALSE, 0);
    gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

    /* On utilise une boucle for pour calculer la somme des cases de la diagonale */

    for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
    {
        for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
        {
            if(prog->i == prog->j)
            {
                prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
                prog->Valeur_Result=prog->Valeur1[prog->i][prog->j]+prog->Valeur_Result;
            }
        }
    }

    /* Affichage de la valeur de la trace */
```

```
   prog->Text = g_strdup_printf("La valeur de la trace pour cette matrice est %.2f", prog-
>Valeur_Result);
   prog->pLabel = gtk_label_new(prog->Text);
   g_free(prog->Text);

   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

   /* Création d'une box contenant 2 boutons */

   prog->pHBox=gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   /* Création des boutons */

   prog->Button = gtk_button_new_with_label("Retourner au menu principal"); // permet de
retourner au menu principal
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Quitter"); // Bouton permettant de quitter le
programme
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowResultat);
}

/* Fonction appelée depuis le menu */

void Taille_Matrice_Enregistrer(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille de la matrice");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */
```

```
   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   /* Bouton appelant la fonction permmettant de saisir la matrice à enregistrer */

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Saisir_Enregistrer_Matrice),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction précedente */

void Saisir_Enregistrer_Matrice(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Récupération des valeurs permettant de créer la table */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */
```

```
   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création d'une box horizontale pour les boutons */

   prog->pHBox = gtk_hbox_new(FALSE, 0);

   /* Création de la table grâce aux valeurs */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);

   /* On utilise 2 boucles for pour créer le bon nombre de cases */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      }
   }

   /* Création des différents boutons */

   prog->Button = gtk_button_new_with_label("Enregistrer!"); // Appelle la fonction enregistrant la
matrice
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_Dialogue_Enregistrer_Matrice), prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Retourner au menu principal"); //Permet de
retourner a la fenetre principale
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Quitter"); //Permet de quitter le programme
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée suite au clic sur "Enregistrer!" */
```

```
void Boite_Dialogue_Enregistrer_Matrice(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowEnregistrer = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowEnregistrer), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowEnregistrer), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowEnregistrer), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowEnregistrer), prog->pVBox);

   /* Création des éléments permettant de saisir le nom du fichier */

   prog->pLabel= gtk_label_new("Entrer le nom de votre fichier");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

   prog->Entry= gtk_entry_new();
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Entry, TRUE, FALSE, 0);

   /* Création du bouton appelant la fonction qui va stocker la matrice dans un fichier */

   prog->Button= gtk_button_new_with_label("Ok!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Enregistrer_Matrice),prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox),prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowEnregistrer);
}

/* Fonction appelée lors du clic sur "Ok!" */

void Enregistrer_Matrice(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Récupération du texte, ajout de .txt, ouverture du fichier */

   prog->Text = gtk_entry_get_text(prog->Entry);
   prog->Text = g_strdup_printf("%s.txt", prog->Text);
   prog->fichier = fopen(prog->Text, "w+");

   /* On utilise 2 boucles for pour stocker chaque case dans le bon ordre */
```

```
   for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {
         prog->Valeur_Resultat[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
         prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
         fprintf(prog->fichier, "%.2f ", prog->Valeur_Resultat[prog->i][prog->j]);
      }
      fprintf(prog->fichier, "\n");
   }

   /* On ferme le fichier et la fenetre. Le fichier est enregistré ! */

   fclose(prog->fichier);
   gtk_widget_destroy(prog->pWindowEnregistrer);
   gtk_main();
}

/* Fonction calculant une matrice à la puissance n. appelée depuis le menu */

void Taille_Matrice_Puissance(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille de la  matrice");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);
   gtk_widget_show_all(prog->pWindowTaille);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des éléments permettant de saisir les dimensions de la matrice et la puissance */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);
```

```
   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   prog->Label_Puissance = gtk_label_new("Entrer la puissance a calculer");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Label_Puissance, TRUE, FALSE, 0);

   prog->Label_Puissance = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Label_Puissance, TRUE, FALSE, 0);

   /* Création du bouton qui va permettre de saisir la matrice */

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Puissance),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction précédente */

void Puissance(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Récupération des valeurs */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));
   prog->Puissance=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Label_Puissance));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
```

```
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création de la table grâce aux valeurs récupérées */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);

   /* Utilisation de 2 boucles for pour créer le bon nombre de cases */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j], prog->j-1,
prog->j,prog->i-1, prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      }
   }

   /* Création du bouton permettant d'appeler la fonction qui va calculer et afficher le résultat */

   prog->Button = gtk_button_new_with_label("Multiplier!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Resultat_Puissance),
prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée suite au clic sur "Multiplier!" de la fonction précédente */

void Resultat_Puissance(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   /* Création de la table résultat */
```

```
   prog->pTable_Result=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);

   /* Utilisation de 2 boucles for pour initialiser les différentes variables nécessaires */

    for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {
         prog->Valeur_Resultat[prog->i][prog->j]=0;
         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
         prog->Valeur2[prog->i][prog->j]=prog->Valeur1[prog->i][prog->j];
         prog->Valeur_Tempo[prog->i][prog->j]=0;
      }
   }

   /* Utilisation de 4 boucles for. La premiere pour faire n fois la multiplication, la deuxieme et la
troisieme pour se placer sur chaque case et la troisiéme
   pour multiplier les lignes et colonnes */

   for(prog->l=1; prog->l<prog->Puissance; prog->l++)
   {
      for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
      {
         for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
         {
            for (prog->k=1; prog->k<=prog->Nbr_Lig; prog->k++)
            {
               prog->Valeur_Resultat[prog->i][prog->j]=(prog->Valeur1[prog->i][prog->k]*prog-
>Valeur2[prog->k][prog->j])+prog->Valeur_Tempo[prog->i][prog->j];
               prog->Valeur_Tempo[prog->i][prog->j]= prog->Valeur_Resultat[prog->i][prog->j];

            }
            prog->Valeur_Tempo[prog->i][prog->j]=0;
         }
      }
      for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)// Cette boucle sert à avoir les bonnes
valeurs pour les multiplications suivantes (n > 2)
      {
         for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
         {
            prog->Valeur2[prog->i][prog->j]=prog->Valeur_Resultat[prog->i][prog->j];
         }
      }
   }

   /* Utilisation de 2 boucles for pour afficher le résultat */

    for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
```

```
    {
       for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
       {
          prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->i][prog->j]);
          prog->pLabel = gtk_label_new(prog->Text);
          gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel, prog->j-1, prog-
>j,prog->i-1, prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
          g_free(prog->Text);
       }
    }
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);

    /* Création d'une box contenant 3 boutons */

    prog->pHBox=gtk_hbox_new(FALSE, 0);
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

    /* Création des boutons */

    prog->Button = gtk_button_new_with_label("Enregistrer dans un fichier .txt"); // Appelle la
fonction permetant d'enregistrer la matrice. On utilise la meme que pour l'addition
    g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_dialogue_fichier_addition_soustraction),prog);
    gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

    prog->Button = gtk_button_new_with_label("Retourner au menu principal"); // Appelle la
fonction permettant de revenir au menu principal
    g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
    gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

    prog->Button = gtk_button_new_with_label("Quitter"); // Appelle une boite de dialoque
permettant de quitter le programme */
    g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
    gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);
    gtk_widget_show_all(prog->pWindowResultat);
}

/* Fonction appelée depuis le menu pour calculer le déterminant */

void Taille_Matrice_Determinant(GtkWidget *widget, gpointer data)
{
     Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

    /* Création d'une fenetre */

    prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille de la matrice");
    gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
```

```
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Saisir_Matrice_Determinant),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction */

void Saisir_Matrice_Determinant(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupére les valeurs */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

```
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création de la table grâce aux valeurs récupérées */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);

   /* Utilisation de 2 boucles for pour créer le bon nombre de cases */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      }
   }

   /* Création du bouton appelant la fonction permettant de calculer le déterminant et de l'afficher */

   prog->Button = gtk_button_new_with_label("Calculer!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Calcul_Determinant),
prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée suite au clic sur "Calculer!" de la fonction précédente */

void Calcul_Determinant(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
```

```
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   /* On utilise la méthode du pivot de Gauss pour trouver le déterminant. */

   prog->deter=1;

   for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {

         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));


      }
   }

   for(prog->i = 1; prog->i <= prog->Nbr_Lig; prog->i++) // Recherche du plus grand terme de la
colonne i en valeur absolue.
   {
      prog-> max = 0;

      prog->k = prog->i;
      for (prog->j = prog->i; prog->j <= prog->Nbr_Col; prog->j++)
      {
         if (fabs(prog->Valeur1[prog->i][prog->j]) > prog->max)
         {
            prog-> max = fabs(prog->Valeur1[prog->i][prog->j]);
             prog->k = prog->j;
         }
      }

      if (prog->k != prog->i) // On échange les lignes pour que l'élément le plus grand (le pivot) soit
sur la diagonale
      {
         for(prog->j=1;prog->j<=prog->Nbr_Col;prog->j++)
         {prog->echange = prog->Valeur1[prog->j][prog->i];
         prog->Valeur1[prog->j][prog->i] = prog->Valeur1[prog->j][prog->k];
         prog->Valeur1[prog->j][prog->k] = prog->echange;}
         prog->deter *= -1; // Échange de lignes : le déterminant est changé en son opposé

      }
```

```
        /* on ajoute à chaque ligne un multiple de la ligne pivot pour annuler les termes sous le pivot,
ce qui ne modifie pas le déterminant */

        for (prog->j = prog->i+1; prog->j <= prog->Nbr_Col; prog->j++) //
        {
           prog->multiplicateur = -prog->Valeur1[prog->i][prog->j]/prog->Valeur1[prog->i][prog->i];
           for (prog->ii = prog->i; prog->ii <= prog->Nbr_Col; prog->ii++)
           {
              prog->Valeur1[prog->ii][prog->j] += prog->Valeur1[prog->ii][prog->i]*prog-
>multiplicateur;
           }

        }

        /* La matrice est triangulaire. Son déterminant est le produit des termes de la diagonale */

        prog->deter *= prog->Valeur1[prog->i][prog->i];
     }

     prog->Text = g_strdup_printf("La valeur du determinant pour cette matrice est %.2f", prog-
>deter);
     prog->pLabel = gtk_label_new(prog->Text);
     g_free(prog->Text);

     gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);

     /* Création d'une box contenant 2 boutons */

     prog->pHBox=gtk_hbox_new(FALSE, 0);
     gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

     /* Création des boutons */

     prog->Button = gtk_button_new_with_label("Retourner au menu principal"); // Appelle la
fonction permettant de revenir au menu principal
     g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
     gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

     prog->Button = gtk_button_new_with_label("Quitter"); // Affiche une boite de dialogue
permettant de quitter l'application */
     g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
     gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

     gtk_widget_show_all(prog->pWindowResultat);

}

/* Fonction appelée depuis le menu principal */
```

```
void Taille_Matrice_Inverse(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille de la matrice");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);


   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Saisir_Matrice_Inverse),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction précédente */

void Saisir_Matrice_Inverse(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */
```

```
   /* On récupére les valeurs */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);

   /* Création de la table grâce aux valeurs récupérées */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable1, TRUE, FALSE, 0);

   /* Utilisation de 2 boucles for pour créer le bon nombre de cases */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      }
   }

   /* Création du bouton appelant la fonction permettant de calculer l'inverse de la matrice et de
l'afficher */

   prog->Button = gtk_button_new_with_label("Calculer!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Calcul_Inverse),
prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée lors du clic sur "Calculer!" de la fonction précédente */
```

```
void Calcul_Inverse(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

    prog->pTable_Result = gtk_table_new(prog->Nbr_Lig,1, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);

   /* On utilise la méthode du pivot de Gauss pour trouver le déterminant. */

   prog->deter=1;

   for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {

         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));


      }
   }

   for(prog->i = 1; prog->i <= prog->Nbr_Lig; prog->i++) // Recherche du plus grand terme de la
colonne i en valeur absolue.
   {
     prog-> max = 0;

      prog->k = prog->i;
      for (prog->j = prog->i; prog->j <= prog->Nbr_Col; prog->j++)
      {
         if (fabs(prog->Valeur1[prog->i][prog->j]) > prog->max)
         {
            prog-> max = fabs(prog->Valeur1[prog->i][prog->j]);
             prog->k = prog->j;
```

```
        }
      }

      if (prog->k != prog->i) // On échange les lignes pour que l'élément le plus grand (le pivot) soit
sur la diagonale
      {
        for(prog->j=1;prog->j<=prog->Nbr_Col;prog->j++)
        {prog->echange = prog->Valeur1[prog->j][prog->i];
        prog->Valeur1[prog->j][prog->i] = prog->Valeur1[prog->j][prog->k];
        prog->Valeur1[prog->j][prog->k] = prog->echange;}
        prog->deter *= -1; // Échange de lignes : le déterminant est changé en son opposé

      }

      /* on ajoute à chaque ligne un multiple de la ligne pivot pour annuler les termes sous le pivot,
ce qui ne modifie pas le déterminant */

      for (prog->j = prog->i+1; prog->j <= prog->Nbr_Col; prog->j++) //
      {
        prog->multiplicateur = -prog->Valeur1[prog->i][prog->j]/prog->Valeur1[prog->i][prog->i];
        for (prog->ii = prog->i; prog->ii <= prog->Nbr_Col; prog->ii++)
        {
          prog->Valeur1[prog->ii][prog->j] += prog->Valeur1[prog->ii][prog->i]*prog-
>multiplicateur;
        }
      }

      /* La matrice est triangulaire. Son déterminant est le produit des termes de la diagonale */

      prog->deter *= prog->Valeur1[prog->i][prog->i];
    }
    if (prog->deter==0){prog->pLabel = gtk_label_new("La matrice n'est pas inversible car le
déterminant est égal à 0");
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);}

    /* On utilise la méthode de l'élimination de Jordan pour trouver les solutions. */

    else
    {for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
    {

      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {
        prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
        if (prog->i == prog->j)
        {prog->Valeur_Resultat[prog->i][prog->j]=1;}
        else {prog->Valeur_Resultat[prog->i][prog->j]=0;}
      }
    }
```

```
   for(prog->i = 1; prog->i <= prog->Nbr_Lig; prog->i++)
   {
      prog->echange=prog->Valeur1[prog->i][prog->i];

      for (prog->j = 1; prog->j <= prog->Nbr_Lig; prog->j++)
      {
         prog->Valeur1[prog->j][prog->i]=prog->Valeur1[prog->j][prog->i]/prog->echange;
         prog->Valeur_Resultat[prog->j][prog->i]=prog->Valeur_Resultat[prog->j][prog->i]/prog-
>echange;
      }



         for (prog->ii = 1; prog->ii <= prog->Nbr_Col; prog->ii++)
         {
            if (prog->ii != prog->i)
            {
               prog->multiplicateur=-prog->Valeur1[prog->i][prog->ii];

               for(prog->k = 1; prog->k<=prog->Nbr_Col;prog->k++)
               {
                  prog->Valeur1[prog->k][prog->ii] += prog->Valeur1[prog->k][prog->i]*prog-
>multiplicateur;
                  prog->Valeur_Resultat[prog->k][prog->ii] += prog->Valeur_Resultat[prog->k][prog-
>i]*prog->multiplicateur;
               }

            }
         }
      }}

   if (prog->deter != 0)
   {for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {for(prog->j=1; prog->j<=prog->Nbr_Col;prog->j++)
   {
      prog->Text = g_strdup_printf("%.2f", prog->Valeur_Resultat[prog->j][prog->i]);
      prog->pLabel = gtk_label_new(prog->Text);
      gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel, prog->j-1, prog->j,prog-
>i-1,prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      g_free(prog->Text);
   }}}

   /* Création d'une box contenant 2 boutons */

   prog->pHBox=gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   /* Création des boutons */
```

```
   prog->Button = gtk_button_new_with_label("Enregistrer dans un fichier .txt"); //Permet
d'appeler la fonction qui enregistre dans un fichier
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Boite_dialogue_fichier_addition_soustraction),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Retourner au menu principal"); // Appelle la
fonction permettant de revenir au menu principal
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Quitter"); // Affiche une boite de dialogue
permettant de quitter l'application */
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowResultat);

}

/* Fonction appelée depuis le menu principal */

void Taille_Matrice_Systeme(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data; /*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création d'une fenetre */

   prog->pWindowTaille = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowTaille), "Taille de la matrice");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowTaille), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowTaille), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box contenant les differents elements */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowTaille), prog->pVBox);

   /* Création des zones de textes et des textes et insertion dans la box */

   prog->Nombre_Colonne = gtk_label_new("Entrer le nombre de colonnes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Colonne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Colonne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Colonne, TRUE, FALSE,
0);
```

```
   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de lignes de la matrice");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Nombre_Ligne = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Nombre_Ligne, TRUE, FALSE,
0);

   prog->Nombre_Ligne = gtk_label_new("Entrer le nombre de coordonnees du vecteur");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Nombre_Ligne, TRUE, FALSE, 0);

   prog->Entry_Dim = gtk_spin_button_new_with_range(1,300,1);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Entry_Dim, TRUE, FALSE, 0);

   prog->Button = gtk_button_new_with_label("Ok!");
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);
   g_signal_connect(G_OBJECT(prog->Button), "clicked",
G_CALLBACK(Saisir_Matrice_Systeme),prog);

   gtk_widget_show_all(prog->pWindowTaille);
}

/* Fonction appelée lors du clic sur "Ok!" de la fonction précédente */

void Saisir_Matrice_Systeme(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* On récupére les valeurs */

   prog->Nbr_Col=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Colonne));
   prog->Nbr_Lig=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog-
>Entry_Nombre_Ligne));
   prog->Dim=gtk_spin_button_get_value_as_int(GTK_SPIN_BUTTON(prog->Entry_Dim));

   /* Création de la fenetre */

   prog->pWindowDonnees = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowDonnees), "Saisie des donnees");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowDonnees), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowDonnees), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowDonnees), prog->pVBox);
```

```
   prog->pHBox = gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);

   /* Création de la table grâce aux valeurs récupérées */

   prog->pTable1=gtk_table_new(prog->Nbr_Lig,prog->Nbr_Col, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->pTable1, TRUE, FALSE, 0);

   prog->pLabel=gtk_label_new("*");
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->pLabel, TRUE, FALSE, 0);

   prog->pBox = gtk_vbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->pBox, TRUE, FALSE, 0);

   prog->pTable_Result = gtk_table_new(prog->Nbr_Lig,1, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->pTable_Result, TRUE, FALSE, 0);

   for (prog->l=1;prog->l<=prog->Nbr_Lig;prog->l++)
   {
      prog->Text = g_strdup_printf("coeff%d \n", prog->l);
      prog->pLabel = gtk_label_new(prog->Text);
      g_free(prog->Text);
      gtk_box_pack_start(GTK_BOX(prog->pBox), prog->pLabel, TRUE, FALSE, 0);
   }

   prog->pTable2=gtk_table_new(prog->Nbr_Lig,1, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->pTable2, TRUE, FALSE, 0);

   /* Utilisation de 2 boucles for pour créer le bon nombre de cases */

   for(prog->i=1; prog->i<=prog->Nbr_Col;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Lig; prog->j++)
      {
         prog->Case1[prog->i][prog->j]= gtk_spin_button_new_with_range(-300,300,0.00000001);
         gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
         gtk_table_attach(GTK_TABLE(prog->pTable1), prog->Case1[prog->i][prog->j],prog->i-1,
prog->i, prog->j-1, prog->j,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      }
   }
   for(prog->i=1; prog->i<=prog->Dim;prog->i++)
   {
      prog->Case2[1][prog->i]= gtk_spin_button_new_with_range(-300,300,0.00000001);
      gtk_spin_button_set_value(prog->Case1[prog->i][prog->j], 0);
      gtk_table_attach(GTK_TABLE(prog->pTable2), prog->Case2[1][prog->i],0, 1, prog->i-1,
prog->i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
   }

   /* Création du bouton appelant la fonction permettant de trouver les solutions si elles existent et
de les afficher */
```

```
   prog->Button = gtk_button_new_with_label("Calculer!");
   g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Calcul_Systeme),
prog);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->Button, TRUE, FALSE, 0);

   gtk_widget_show_all(prog->pWindowDonnees);
}

/* Fonction appelée lors du clic sur "Calculer!" de la fonction précédente */

void Calcul_Systeme(GtkWidget *widget, gpointer data)
{
   Structure *prog=(Structure *)data;/*Ligne qui permet de réutiliser toute les données de la
structure principale */

   /* Création de la fenetre */

   prog->pWindowResultat = gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title(GTK_WINDOW(prog->pWindowResultat), "Resultat");
   gtk_window_set_default_size(GTK_WINDOW(prog->pWindowResultat), 320, 200);
   g_signal_connect(G_OBJECT(prog->pWindowResultat), "destroy",
G_CALLBACK(gtk_main_quit), NULL);

   /* Création de la box */

   prog->pVBox = gtk_vbox_new(FALSE, 0);
   gtk_container_add(GTK_CONTAINER(prog->pWindowResultat), prog->pVBox);

   prog->pTable_Result = gtk_table_new(prog->Nbr_Lig,1, TRUE);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pTable_Result, TRUE, FALSE, 0);

   /* On utilise la méthode du pivot de Gauss pour trouver le déterminant. */

   prog->deter=1;

   for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {

         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));


      }
   }

   for(prog->i = 1; prog->i <= prog->Nbr_Lig; prog->i++) // Recherche du plus grand terme de la
colonne i en valeur absolue.
```

```
    {
       prog-> max = 0;

       prog->k = prog->i;
       for (prog->j = prog->i; prog->j <= prog->Nbr_Col; prog->j++)
       {
          if (fabs(prog->Valeur1[prog->i][prog->j]) > prog->max)
          {
             prog-> max = fabs(prog->Valeur1[prog->i][prog->j]);
              prog->k = prog->j;
          }
       }

       if (prog->k != prog->i) // On échange les lignes pour que l'élément le plus grand (le pivot) soit
sur la diagonale
       {
          for(prog->j=1;prog->j<=prog->Nbr_Col;prog->j++)
          {prog->echange = prog->Valeur1[prog->j][prog->i];
          prog->Valeur1[prog->j][prog->i] = prog->Valeur1[prog->j][prog->k];
          prog->Valeur1[prog->j][prog->k] = prog->echange;}
          prog->deter *= -1; // Échange de lignes : le déterminant est changé en son opposé

       }

       /* on ajoute à chaque ligne un multiple de la ligne pivot pour annuler les termes sous le pivot,
ce qui ne modifie pas le déterminant */

       for (prog->j = prog->i+1; prog->j <= prog->Nbr_Col; prog->j++) //
       {
          prog->multiplicateur = -prog->Valeur1[prog->i][prog->j]/prog->Valeur1[prog->i][prog->i];
          for (prog->ii = prog->i; prog->ii <= prog->Nbr_Col; prog->ii++)
          {
             prog->Valeur1[prog->ii][prog->j] += prog->Valeur1[prog->ii][prog->i]*prog-
>multiplicateur;
          }

       }

       /* La matrice est triangulaire. Son déterminant est le produit des termes de la diagonale */

       prog->deter *= prog->Valeur1[prog->i][prog->i];
    }

    if (prog->deter==0){prog->pLabel = gtk_label_new("Il n'y a pas de solutions a ce systeme");
    gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pLabel, TRUE, FALSE, 0);}

    /* On utilise la méthode de l'élimination de Jordan pour trouver les solutions. */
    else
    {for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
    {
```

```
      prog->Valeur2[1][prog->i]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog-
>Case2[1][prog->i]));
      for(prog->j=1; prog->j<=prog->Nbr_Col; prog->j++)
      {
         prog->Valeur1[prog->i][prog-
>j]=gtk_spin_button_get_value_as_float(GTK_SPIN_BUTTON(prog->Case1[prog->i][prog->j]));
      }
   }

   for(prog->i = 1; prog->i <= prog->Nbr_Lig; prog->i++)
   {
      prog->echange=prog->Valeur1[prog->i][prog->i];

      for (prog->j = 1; prog->j <= prog->Nbr_Lig; prog->j++)
      {
         prog->Valeur1[prog->j][prog->i]=prog->Valeur1[prog->j][prog->i]/prog->echange;

      }
         prog->Valeur2[1][prog->i]=prog->Valeur2[1][prog->i]/prog->echange;


         for (prog->ii = 1; prog->ii <= prog->Nbr_Col; prog->ii++)
         {
            if (prog->ii != prog->i)
            {
               prog->multiplicateur=-prog->Valeur1[prog->i][prog->ii];

               for(prog->k = 1; prog->k<=prog->Nbr_Col;prog->k++)
               {
                  prog->Valeur1[prog->k][prog->ii] += prog->Valeur1[prog->k][prog->i]*prog-
>multiplicateur;
               }
            prog->Valeur2[1][prog->ii] += prog->Valeur2[1][prog->i]*prog->multiplicateur;
            }
         }
      }}
   if (prog->deter != 0)
   {for(prog->i=1; prog->i<=prog->Nbr_Lig;prog->i++)
   {
      prog->Text = g_strdup_printf("%.2f\n", prog->Valeur2[1][prog->i]);
      prog->pLabel = gtk_label_new(prog->Text);
      gtk_table_attach(GTK_TABLE(prog->pTable_Result), prog->pLabel, 0, 1,prog->i-1,prog-
>i,GTK_EXPAND | GTK_FILL, GTK_EXPAND| GTK_FILL, 0, 0);
      g_free(prog->Text);
   }}

   /* Création d'une box contenant 2 boutons */

   prog->pHBox=gtk_hbox_new(FALSE, 0);
   gtk_box_pack_start(GTK_BOX(prog->pVBox), prog->pHBox, TRUE, FALSE, 0);
```

```
    /* Création des boutons */

    prog->Button = gtk_button_new_with_label("Retourner au menu principal"); // Appelle la
fonction permettant de revenir au menu principal
    g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(Retour),prog);
    gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

    prog->Button = gtk_button_new_with_label("Quitter"); // Affiche une boite de dialogue
permettant de quitter l'application */
    g_signal_connect(G_OBJECT(prog->Button), "clicked", G_CALLBACK(OnQuitter),
(GtkWidget*) prog->pWindowResultat);
    gtk_box_pack_start(GTK_BOX(prog->pHBox), prog->Button, TRUE, FALSE, 0);

    gtk_widget_show_all(prog->pWindowResultat);
}
```

- bibiothèque contenant la déclaration des variables et des fonctions.

```
#ifndef PROJET_H_INCLUDED
#define PROJET_H_INCLUDED

void OnQuitter(GtkWidget* widget, gpointer data);

void Taille_Matrice_Addition(GtkWidget *widget, gpointer data);
void Taille_Matrice_Soustraction(GtkWidget *widget, gpointer data);
void Taille_Matrice_Multiplication(GtkWidget *widget, gpointer data);
void Taille_Matrice_Systeme(GtkWidget *widget, gpointer data);
void Taille_Matrice_Inverse(GtkWidget *widget, gpointer data);
void Taille_Matrice_Puissance(GtkWidget *widget, gpointer data);
void Taille_Matrice_Trace(GtkWidget* widget, gpointer data);
void Taille_Matrice_Determinant(GtkWidget *widget, gpointer data);
void Taille_Matrice_Enregistrer(GtkWidget *widget, gpointer data);

void Addition(GtkWidget* widget, gpointer data);
void Soustraction(GtkWidget* widget, gpointer data);
void Multiplication(GtkWidget* widget, gpointer data);
void Saisir_Matrice_Systeme(GtkWidget *widget, gpointer data);
void Saisir_Matrice_Inverse(GtkWidget *widget, gpointer data);
void Puissance(GtkWidget *widget, gpointer data);
void Trace(GtkWidget* widget, gpointer data);
void Saisir_Matrice_Determinant(GtkWidget *widget, gpointer data);
void Saisir_Enregistrer_Matrice(GtkWidget *widget, gpointer data);

void Resultat_Addition(GtkWidget *widget,gpointer data);
void Resultat_Soustraction(GtkWidget* widget, gpointer data);
void Resultat_Multiplication(GtkWidget* widget, gpointer data);
void Calcul_Systeme(GtkWidget *widget, gpointer data);
void Calcul_Inverse(GtkWidget *widget, gpointer data);
void Resultat_Puissance(GtkWidget *widget, gpointer data);
void Resultat_Trace(GtkWidget* widget, gpointer data);
void Calcul_Determinant(GtkWidget *widget, gpointer data);
void Boite_Dialogue_Enregistrer_Matrice(GtkWidget *widget, gpointer data);

void Enregistrer_Matrice(GtkWidget *widget, gpointer data);
void Enregistrer_Resultat_Addition_Soustraction(GtkWidget* widget, gpointer data);
void Enregistrer_Resultat_Multiplication(GtkWidget* widget, gpointer data);

void Boite_dialogue_fichier(GtkWidget *widget, gpointer data);
void Boite_dialogue_fichier_addition_soustraction(GtkWidget *widget, gpointer data);
void Boite_dialogue_fichier_multiplication(GtkWidget *widget, gpointer data);

void Retour(GtkWidget *widget, gpointer data);

typedef struct Structure
   {
   FILE *fichier;
```

```
GtkWidget *pWindowMain;
GtkWidget *pWindowTaille;
GtkWidget *pWindowDonnees;
GtkWidget *pWindowResultat;
GtkWidget *pWindowEnregistrer;
GtkWidget *pVBox;
GtkWidget *pHBox;
GtkWidget *pMenuBar;
GtkWidget *pMenu;
GtkWidget *pMenuItem1;
GtkWidget *pMenuItem2;
GtkWidget *pLabel;
GtkWidget *Nombre_Colonne;
GtkWidget *Nombre_Ligne;
GtkWidget *Button;
GtkWidget *Label_Puissance;
GtkWidget *Entry;
GtkWidget *Entry_Nombre_Colonne;
GtkWidget *Entry_Nombre_Colonne1;
GtkWidget *Entry_Nombre_Colonne2;
GtkWidget *Entry_Nombre_Ligne;
GtkWidget *Entry_Nombre_Ligne1;
GtkWidget *Entry_Nombre_Ligne2;
GtkWidget *Entry_Dim;
GtkWidget *pBox;
GtkWidget *pTable1;
GtkWidget *pTable2;
GtkWidget *pTable_Result;
GtkWidget *Destroy;
GtkWidget *Case1[100][100];
GtkWidget *Case2[100][100];
gfloat Valeur1[100][100];
gfloat Valeur2[100][100];
gfloat Valeur_Resultat[100][100];
gfloat Valeur_Tempo[100][100];
gfloat Valeur_Result;
GList *pList;
guint Nbr_Col;
guint Nbr_Col1;
guint Nbr_Col2;
guint Nbr_Lig;
guint Nbr_Lig1;
guint Nbr_Lig2;
guint Puissance;
gfloat max, multiplicateur;
guint ii;
gfloat echange;
gfloat deter;
guint Dim;
guint i;
```

```
    guint j;
    guint k;
    guint l;
    gchar* Text;
    gchar* sNom;
    }Structure;

#endif // PROJET_H_INCLUDED
```