***TP N°1***

1. ***Création d’un menu***

On souhaite créer un menu **Fichier**

Saisie

Affichage

Quitter

* Dans la boîte à outils « Standard », sélectionner le composant « Main Menu » et le placer dans Form1.
* Cliquer sur ce menu, et remplir les différentes cases « Caption ».
* Double Cliquer sur la rubrique « Quitter ». On accède alors à la fonction qui sera exécutée lors d’un clique sur cette rubrique. Compléter cette fonction comme ci-dessous.

void \_\_fastcall TForm1::Quitter1Click(TObject \*Sender)

{ **Form1-> Close();** }

* Exécuter le programme et vérifier.
* Cliquer dans Form1 et modifier sa couleur, via l’inspecteur d’objets, (choisir le blanc, clWhite).
* Vérifier en exécutant.
* On souhaite maintenant interdire, pour l’instant, l’accès à la rubrique « Affichage » du menu.
* Cliquer sur cette rubrique, modifier la propriété « Enabled » en « false » et vérifier en exécutant le programme.
* Ajouter maintenant au menu une rubrique « Couleur » :

## **Fichier Couleur**

Saisie Blanc

Affichage Gris

Quitter

* Puis en double cliquant sur la rubrique « Blanc », compléter la fonction comme ci-dessous :

***void \_\_fastcall TForm1::Blanc1Click(TObject \*Sender)***

***{ Form1->Color = clWhite; }***

* Enfin en double cliquant sur la rubrique « Gris », compléter la fonction comme ci-dessous :

***void \_\_fastcall TForm1::Gris1Click(TObject \*Sender)***

***{Form1->Color = clGray ; }***

1. Création d’une boîte de dialogue

* Ajouter maintenant au menu une rubrique « Outils » :

**Fichier Couleur Outils**

Saisie Blanc A propos de …

Affichage Gris

Quitter

* Poser sur la fenêtre Form1 une « Label » (dans les composants « Standard »), avec les propriétés suivantes :
* Alignment : taCenter
* AutoSize : false
* Caption : TP Builder Octobre 2018
* Color : clAqua
* Font : Times New Roman / Normal / Taille 12
* Height : 80
* Left : 150
* **Name :** Label1
* **Top :** 80
* **Transparent :** false
* **Visible :** true
* **Width :** 145
* Garder les valeurs par défaut des autres propriétés.
* Ajouter au milieu de la Label1 un « BitBtn» (dans les composants « Supplement »), avec les propriétés suivantes :
* **Name : BitBtn1**
* **Kind : bkOK**
* **Visible : true**
* Exécuter pour vérifier.
* Donner maintenant à Label1 et à Bitbtn1 la propriété Visible = false.
* Exécuter pour vérifier.
* Cliquer sur la ligne A propos de … du menu et compléter la fonction comme ci-dessous :

void \_\_fastcall TForm1::Aproposde1Click(TObject \*Sender)

{ Label1->Visible = true; BitBtn1->Visible = true; }

* Cliquer sur le bouton OK et compléter la fonction comme ci-dessous :

***void \_\_fastcall TForm1::BitBtn1Click(TObject \*Sender)***

***{ Label1->Visible = false; BitBtn1->Visible = false; }***

* Exécuter pour vérifier.

1. Création d’une boîte de saisie

* Ajouter sur Form1 une « Label » (dans les composants « Standard »), avec les propriétés suivantes :
* Alignment : taLeftJustify
* AutoSize : false
* Caption : Saisir un nombre :
* Color : clYellow
* Font : Times New Roman / Normal / Taille 10
* Height : 80
* Left : 50
* Name : Label2
* Top : 150
* Transparent : false
* Visible : true
* Width : 160
* Ajouter au milieu de la Label2 un « Edit» (ab dans dans les composants « Standard »), avec les propriétés suivantes:

Saisir un nombre :

* AutoSize : false
* Name : Edit1
* Color : clWindow
* Visible : true
* Exécuter pour vérifier.
* Donner maintenant à Label2 et à Edit1 la propriété Visible = false.
* Exécuter pour vérifier.
* Cliquer sur la rubrique « Saisie » du menu et compléter la fonction comme ci-dessous :

void \_\_fastcall TForm1::Saisie1Click(TObject \*Sender)

{ Label2->Visible = true ; Edit1->Visible = true; }

* Exécuter pour vérifier.
* Dans la boîte Edit1, double cliquer sur l’événement **OnKeyDown** et compléter la fonction comme ci-dessous :

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{float r;

if(Key == VK\_RETURN)

{r = (Edit1->Text).ToDouble();} }

* Puis, ajouter au début du source la déclaration
* *#include <vcl\dstring.h>*

C++ Builder propose une bibliothèque de manipulations de chaînes de caractères. La classe *AnsiString* y a été créée, munie des méthodes appropriées.

Edit1->Text est de type *AnsiString.*

*ToDouble* est une méthode qui transforme une chaîne de type *AnsiString* en un nombre réel.

Attention, aucune précaution n’a été prise, ici , en cas d’erreur de saisie (ceci sera étudié à la fin du TP).

Si on effectue une erreur de saisie (par exemple en tapant une lettre), il y aura traitement d’une exception par WINDOWS. Concrètement, pour nous, ceci se traduit, pour l’instant, par un « plantage ».

L’événement **OnKeyDown** détecte l’appui sur une touche du clavier. On valide ici la saisie lorsque l'utilisateur appuie sur la touche « ENTREE ». Celle-ci n’est pas prise en compte dans la saisie.

Exécuter pour vérifier (avec et sans erreur de saisie).

1. Traitement et affichage d’un résultat

* Modifier la fonction **Edit1KeyDown** comme ci-dessous :

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{float r;

if(Key == VK\_RETURN)

{r = (Edit1->Text).ToDouble();

Affichage1->Enabled = true;

Saisie1->Enabled = false;}}

* Exécuter pour vérifier.
* Déclarer une variable globale comme suite : *float resultat ;*
* Puis modifier la fonction **Edit1KeyDown** comme ci-dessous :

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{ float r;

if(Key == VK\_RETURN)

{r = (Edit1->Text).ToDouble();

Affichage1->Enabled = true;

Saisie1->Enabled = false;

resultat = r\*2 - 54;}}

* Construire le projet pour vérifier qu’il n’y a pas d’erreur (Menu Projet 🡪 Construire).
* Aller chercher dans la palette standard le composant « ListBox » (boîte liste)
* Le poser sur la fenêtre principale, vers la droite, avec les propriétés suivantes :
* Visible = false
* Color = clYellow
* Height = 65
* Left = 280
* Top = 24
* Width = 120
* Name : ListBox1
* Modifier la fonction **Saisie1Click** comme ci-dessous

void \_\_fastcall TForm1::Saisie1Click(TObject \*Sender)

{Label2->Visible = true;

Edit1->Visible = true;

ListBox1->Visible = false;}

* Cliquer sur la rubrique « Affichage » du menu. Ecrire la fonction de gestion de l’événement OnClick comme ci-dessous :

void \_\_fastcall TForm1::Affichage1Click(TObject \*Sender)

{AnsiString chaine;

ListBox1->Visible = true;

Edit1->Visible = false;

Label2->Visible = false;

chaine= FloatToStrF(resultat,AnsiString::sffGeneral,7,10);

ListBox1->Clear();

ListBox1->Items->Add(chaine);

Affichage1->Enabled = false;

Saisie1->Enabled = true;}

* La fonction FloatToStrF permet de convertir un nombre réel en une chaîne de caractères. Voir l’aide pour plus d’informations.
* Exécuter pour vérifier.

1. Affichage d’une série de valeurs

* Déclarer une variable globale **float tab[3]***,* supprimer la variable **résultat**.
* Modifier les fonctions **Affichage1Click**  et **Edit1KeyDown** comme ci-dessous :

void \_\_fastcall TForm1::Affichage1Click(TObject \*Sender)

{int i;

AnsiString chaine;

ListBox1->Visible = true;

Edit1->Visible = false;

Label2->Visible = false;

ListBox1->Clear();

ListBox1->Items->Add("Voici le résultat:");

for(i=0;i<3;i++)

{chaine= FloatToStrF(tab[i],AnsiString::sffGeneral,7,10);

ListBox1->Items->Add(chaine);}

Affichage1->Enabled = false;

Saisie1->Enabled = true;}

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{ float r;

if(Key == VK\_RETURN)

{ r = (Edit1->Text).ToDouble();

tab[0] = r\*2 - 54;

tab[1] = r\*3 + 100;

tab[2] = r\*2.5 - 8.5;

Saisie1->Enabled = false;

Affichage1->Enabled = true;} }

* Exécuter pour vérifier.

1. Fichiers

* Ajouter au menu dans la rubrique « Fichier », une sous-rubrique « Ouvrir » et une sous-rubrique « Enregistrer »
* La sous-rubrique « Ouvrir », possède la propriété Enabled = true.
* La sous-rubrique « Enregistrer », possède la propriété Enabled = false.
* Modifier les fonctions **Affichage1Click**  et **Edit1KeyDown** comme ci-dessous :

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{ float r;

if(Key == VK\_RETURN)

{ r = (Edit1->Text).ToDouble();

Affichage1->Enabled = true;

Enregistrer->Enabled = true;

Saisie1->Enabled = false;

Ouvrir1->Enabled = false;

tab[0] = r\*2 - 54; tab[1] = r\*3 + 100; tab[2] = r\*2.5 - 8.5;}}

void \_\_fastcall TForm1::Affichage1Click(TObject \*Sender)

{ int i; AnsiString chaine;

ListBox1->Visible = true;

Edit1->Visible = false;

Label2->Visible = false;

ListBox1->Clear();

ListBox1->Items->Add("Voici le résultat:");

for(i=0;i<3;i++)

{ chaine= FloatToStrF(tab[i],AnsiString::sffGeneral,7,10);

ListBox1->Items->Add(chaine); }

Affichage1->Enabled = false;

Saisie1->Enabled = true;

Enregistrer1->Enabled = true;}

* Exécuter pour vérifier.
* Aller chercher dans la palette Dialogues le composant « SaveDialog » (boîte de sauvegarde)
* Le poser sur la fenêtre principale, avec les propriétés suivantes :
* FileName : tp1.dat (nom apparaissant par défaut)
* InitialDir : c:\Program Files\Borland\Cbuilder3\Projects\Sei
* Filter : \*.dat
* Name : SaveDialog1.
* Cliquer sur la rubrique « Enregistrer » du menu et rédiger la fonction Enregistrer1Click comme ci-dessous.

void \_\_fastcall TForm1::Enregistrer1Click(TObject \*Sender)

{

bool res;

res = SaveDialog1->Execute();

}

* Exécuter pour vérifier.
* C’est au programme de sauvegarder réellement le fichier. La variable « res » vaut***false*** si l’utilisateur a annulé l’opération, ***true*** s’il a réellement sélectionné un fichier.
* Ajouter le fichier ***stdio.h*** dans la lister des ***#include***
* Modifier fonction **Enregistrer1Click** comme ci-dessous :

void \_\_fastcall TForm1::EnregistrerClick(TObject \*Sender)

{ FILE \*f;

bool res;

res = SaveDialog1->Execute();

if(res==true)

{ f = fopen((SaveDialog1->FileName).c\_str(), "wb");

fwrite(tab,4,3,f);

fclose(f) ;}}

* On utilise ici les fonctions de traitement de fichier classiques du TURBO C. La fonction membre ***c\_str()*** de la classe ***AnsiString*** permet de convertir une « AnsiString » en tableau de caractères classique, nécessaire ici à la fonction ***fopen***.
* Exécuter pour vérifier, en choisissant le nom de fichier proposé par défaut, puis un autre. Vérifier la présence de ces fichiers dans l’explorateur de WINDOWS.
* Aller chercher dans la palette Dialogues le composant « OpenDialog »
* Le poser sur la fenêtre principale, avec les propriétés suivantes :
* FileName : tp1.dat (nom apparaissant par défaut)
* InitialDir : c:\Program Files\Borland\Cbuilder3\Projects\Sei
* Filter : \*.dat
* Name : OpenDialog1.
* Cliquer sur la rubrique « Ouvrir » du menu et rédiger la fonction **Ouvrir1Click** comme ci-dessous.

void \_\_fastcall TForm1::Ouvrir1Click(TObject \*Sender)

{FILE \*f;

bool res;

res = OpenDialog1->Execute();

if(res==true)

{f = fopen((OpenDialog1->FileName).c\_str(), "rb");

fread(tab,4,3,f);

fclose(f);

Affichage1->Enabled = true;}}

* C’est au programme d’ouvrir réellement le fichier. La variable « res » vaut***false*** si l’utilisateur a annulé l’opération, ***true*** s’il a réellement sélectionné un fichier.
* Exécuter pour vérifier.

1. ***Amélioration de la zone de saisie***

* Un composant a le « focus d’entrée », lorsqu’il est actif dès la prochaine action sur le clavier.
* Modifier la fonction comme Saisie1Click ci-dessous :

void \_\_fastcall TForm1::Saisie1Click(TObject \*Sender)

{ Label2->Visible = true;

Edit1->Visible = true;

ListBox1->Visible = false;

Edit1->SetFocus(); // donne le focus d’entrée à la zone d’édition}

* Exécuter pour vérifier.
* Modifier la fonction **Edit1KeyDown** comme ci-dessous :

void \_\_fastcall TForm1::Edit1KeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{

float r;

if(Key == VK\_RETURN)

{

try

{

r = (Edit1->Text).ToDouble();

Affichage1->Enabled = true;

Enregistrer->Enabled = true;

Saisie1->Enabled = false;

Ouvrir1->Enabled = false;

tab[0] = r\*2 - 54;

tab[1] = r\*3 + 100;

tab[2] = r\*2.5 - 8.5;

}

catch(...)

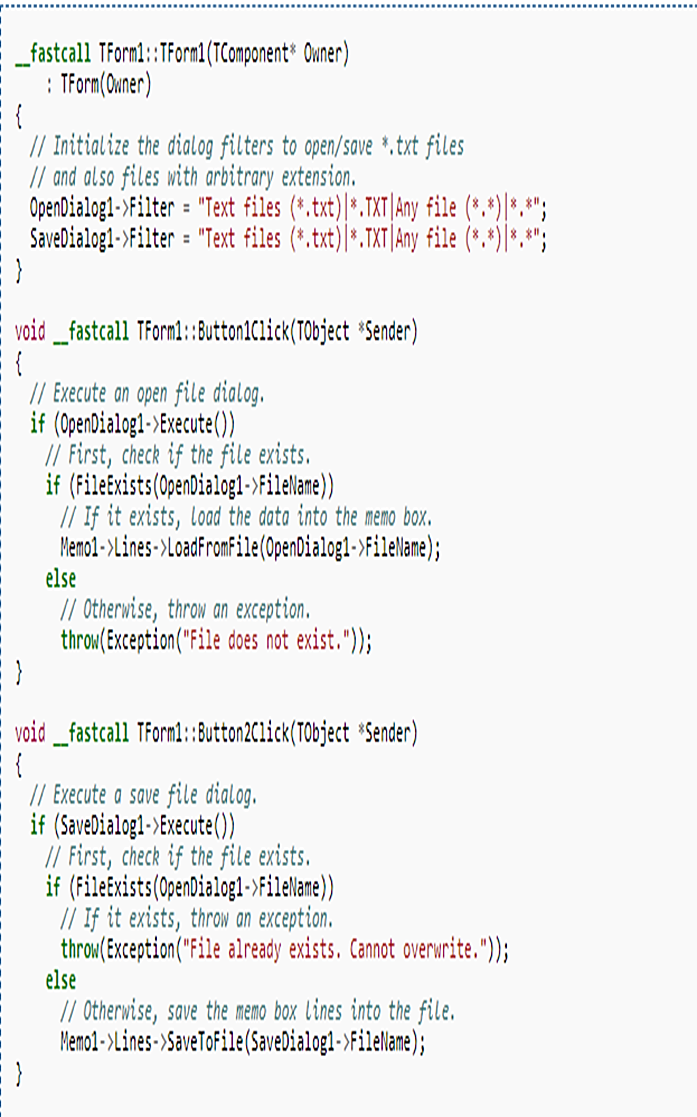
{ ShowMessage("Ce n'est pas un reel"); }} }

* On traite ici l’exception générée par WINDOWS lors d’une erreur de saisie. Si tout se passe bien, le bloc « try » est exécuté, sinon, le bloc « catch » est exécuté.
* On découvre ici, au passage la fonction **ShowMessage** qui permet d’afficher très facilement un message soumis à validation sur l’écran.
* Exécuter sous Builder pour vérifier, en effectuant une erreur de saisie. WINDOWS génère une erreur. Cliquer sur OK et continuer l’exécution.
* Lancer maintenant directement l’exécutable Sei\_Pro.exe depuis l’explorateur de WINDOWS, en effectuant une erreur de saisie.
* Ca marche !

Partie 3 : Gestion des messages- Les boites de dialogue

Les boîtes de dialogue, c'est certain, vous connaissez ! Cependant, afin de nous assurer que nous parlons de la même chose, voici une petite description de ce qu'est une boîte de dialogue. Il s'agit d'une petite fenêtre pouvant servir à plusieurs choses :

* La figure suivante vous montre à quoi ressemblent des boîtes de dialogues « informatives ».



Partie 1 : gestion des erreurs

On souhaite créer un programme de sauvegarde d’un memo dans un fichier texte.

Cet exemple utilise une zone de mémo, deux boutons, une boîte de dialogue ouverte et une boîte de dialogue d'enregistrement sur un formulaire. Lorsque l'application est exécutée, les filtres de dialogue d'ouverture et d'enregistrement de dialogue sont initialisés afin d'ouvrir et de sauvegarder des fichiers \* .txt ou des fichiers avec toute extension.

Cliquez sur le bouton Ouvrir / Enregistrer pour exécuter la boîte de dialogue Ouvrir / Enregistrer. Une exception est levée lorsque vous essayez d'ouvrir un fichier qui n'existe pas ou d'essayer de remplacer un fichier à l'aide de la boîte de dialogue Enregistrer.

* Recopiez le code suivant
* Exécutez et Vérifiez
* Améliorer le code afin de guider l’utilisateur à éviter les erreurs (En ajoutant des boites de message).
* Composants à ajouter à la forme :
* Bouton1 : Ouvrir
* Boutton2 : Enregistrer
* Memo

Partie 2 : Gestion des messages

L’interface suivante se compose de 4 buttons chacun introduits un type de message. Vous pouvez vous inspirer de cette partie afin d’améliorer le code de gestion des erreurs dans la 1er Partie.

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{ShowMessage("C'est mon premier message");

}

//----------------------------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{Application->MessageBox("Voulez-vous vraiment supprimer ce fichier ?","Confirmation",MB\_YESNOCANCEL|MB\_ICONQUESTION);}

//-----------------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{MessageDlg("Voulez-vous continuez ?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0);

MessageDlg("Espace mémoire insufusant !",mtWarning,TMsgDlgButtons()<<mbCancel,0);}

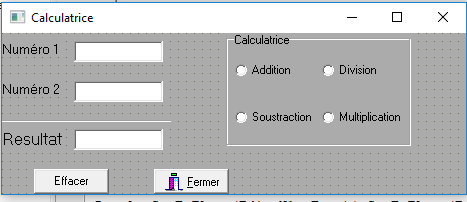
//--------------------------------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{InputBox("Adresse","Taper votre adresse","Annaba"); }

1. Calculatrice

* Dans ce que suit nous allons ecrire un tout petit programme de calculatrice qui effectue une opération à la fois.
* Identifier les types de boutons utilisés dans cette interface.
* Quelles sont les fonctions à utiliser ?
* Critiquez ergonomie de l’interface.



* A noter
* Le composant groupe de boutons radio ***(TRadioGroup)*** simplifie le regroupement de boutons radio et gère leur fonctionnement en commun. Pour ajouter des boutons radio à un groupe, modifiez la propriété ***Items*** dans l'inspecteur d'objets ; chaque chaîne de ***Items*** constitue un bouton radio qui apparaît dans le groupe en utilisant la chaîne spécifiée comme libellé. La valeur de la propriété ***ItemIndex*** détermine le bouton radio sélectionné. Affichez les boutons radio sur une ou plusieurs colonnes en définissant la valeur de la propriété ***Columns.*** Pour espacer les boutons, redimensionnez le composant groupe de boutons radio.

La place d'un contrôle dans l'ordre de tabulation ( TabOrder) peut être définie sur n'importe quelle valeur supérieure ou égale à 0. ... Un groupe de boutons radio à un seul TABSTOP au moment de l'exécution