Créer un éditeur de texte enrichi

Auteur : Christophe PIQUER Publié le 23/06/2024

Le tutoriel proposé est la traduction de « Create a Rich Text Editor for Yourself this Christmas! » publié par **Adnan Shameem** pour Noël 2013.
Après cette traduction, quelques solutions sont proposées pour enrichir l'éditeur de texte réalisé.

Sommaire

I Introduction

La première partie est consacrée à la traduction du tutoriel. Il date un peu puisqu'il a été publié en décembre 2013. Cela n'a pas vraiment d'importance, le principe et la manière de procéder restent les mêmes.

Dans cette traduction, je me suis exonéré de quelques phrases ou commentaires de l'auteur qui n'apportent rien à l'élaboration de l'éditeur de texte. J'ai conservé les dénominations en anglais des différents composants et j'ai gardé le même principe pour réaliser la deuxième partie.

Cette dernière est dédiée à des propositions pour améliorer l'éditeur de texte réalisé en suivant le tutoriel proposé par **Adnan Shameem**.

A mon avis, en termes de bonne pratique, il faudrait renommer la totalité des composants et mieux commenter le code, contrairement à ce qu'a fait l'auteur. Aussi, je me suis permis d'ajouter quelques commentaires dans le code initial.

II Partie 1

Offrez-vous un joli petit traitement de texte que vous pourrez utiliser ! Vous pouvez le développer davantage. Il prend aisément en charge **Unicode** et vous permet d'apprendre à utiliser **Toolbar**, **RichMemo**, **ImageList**, la gestion des sauvegardes de fichiers et bien plus encore.

Plain text (le texte brut) désigne tout texte qui n'a pas de style ou de formatage particulier. Tout le texte du début à la fin est du même style. C'est juste un texte simple et rien d'autre. Par exemple, vous pouvez créer du texte brut dans Windows Notepad, Notepad++, GEdit, etc.

Rich text (le texte enrichi), quant à lui, est doté d'un formatage spécial ajouté en arrière-plan. Par exemple, si vous créez des documents dans MS Word ou Libre Office, vous pouvez y ajouter des styles, comme gras, souligné, etc. Sous le capot, ces informations de style sont enregistrées avec le texte. Ce n'est donc pas seulement le texte. Il doit y avoir un peu de style sauvegardé avec. Son style est complexe.

Aujourd'hui, nous allons créer un simple éditeur de texte enrichi.

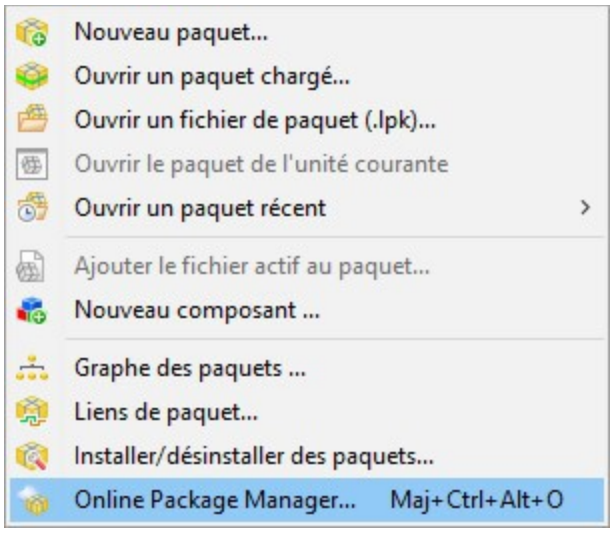
Lazarus dispose d'un bon composant pour l'édition de texte enrichi : TRichMemo. Si vous pensez à toutes les complexités liées à la gestion de tous les caractères et de leurs styles, laissez-moi vous assurer que c'est plus facile que vous ne le pensez. Ne vous inquiétez pas, ce sera un jeu d'enfant de créer cet éditeur de texte enrichi. Continuez simplement à suivre mes indications.

II.A Installer RichMemo

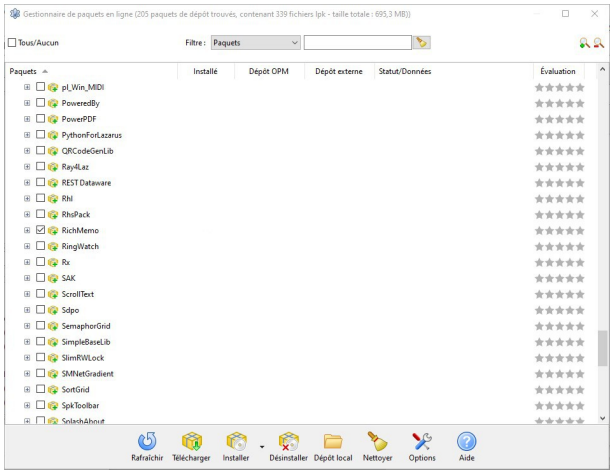
Le composant **TRichMemo** n'est pas préinstallé avec Lazarus. Alors d'abord, téléchargez le paquet du composant **RichMemo**.

*(La suite concernant le téléchargement et l'installation de **RichMemo** n'est pas traduite volontairement car elle est obsolète.)*

Pour installer le composant, rendez-vous dans Lazarus ici :



Puis cochez **RichMemo** et cliquez sur **Installer** :



Le composant **TRichMemo** est installé et visible au niveau de l'onglet **Common Controls**.

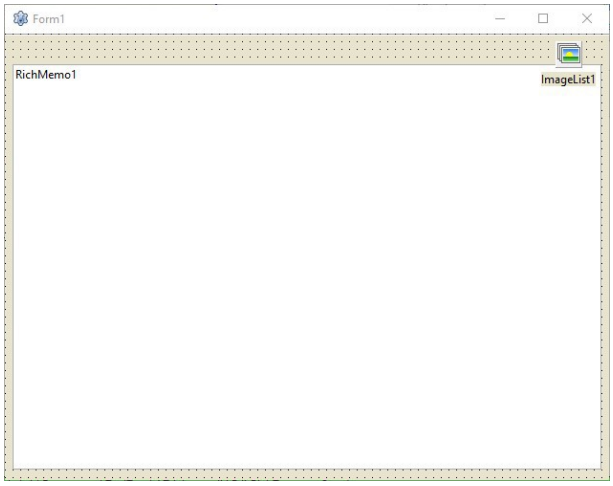
Et enfin compilez pour finaliser l'installation.

II.B Commencer par les bases

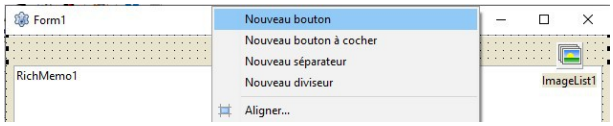
Alors allez-y et créez un nouveau projet d'application (**Projet -> Nouveau projet -> Application -> OK**). Redimensionnez le formulaire à une taille appropriée pour faire de la place aux composants.

Maintenant la barre d'outils. Posez un **TToolBar** sur le formulaire (à partir de l'onglet Contrôles communs). Redimensionnez-le. Mettez également un **TImageList**, n'importe où dans le formulaire (à partir de l'onglet **Common Controls**). Peu importe où vous le placez, car il disparaîtra au moment de l'exécution. Cette liste d'images contiendra les icônes de la barre d'outils.

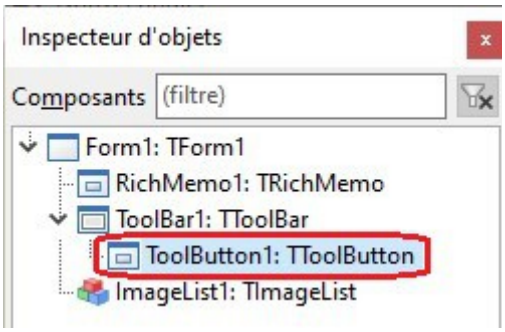
Voici notre formulaire :



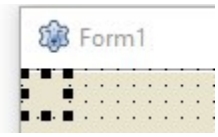
Nous allons maintenant ajouter des éléments à la barre d'outils. Cliquez avec le bouton droit sur le composant **Toolbar1** puis sélectionnez **Nouveau bouton**.



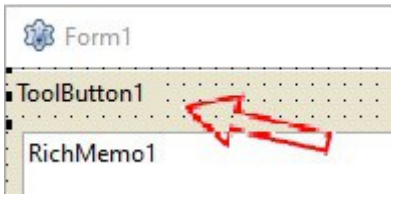
Vous remarquerez qu'un nouveau **TToolButton** apparaît sur l'**inspecteur d'objets**.



Le **TToolbar** et le **TToolButton** sont deux objets distincts. Vous verrez une petite zone dans le coin supérieur gauche de **TToolbar**.

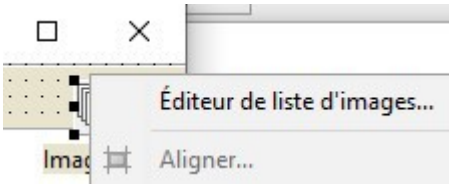


Si vous cliquez dessus, vous pouvez sélectionner le bouton de la barre d'outils. D'un autre côté, vous pouvez cliquer sur la zone pour sélectionner l'objet **TToolbar**. Pour rendre les boutons plus visibles, continuez et sélectionnez l'objet **TToolbar**. Définissez ensuite sa propriété **ShowCaptions** sur **True**. Vous pouvez maintenant facilement voir les **légendes** des boutons de la barre d'outils.



II.C Préparer la liste d'images

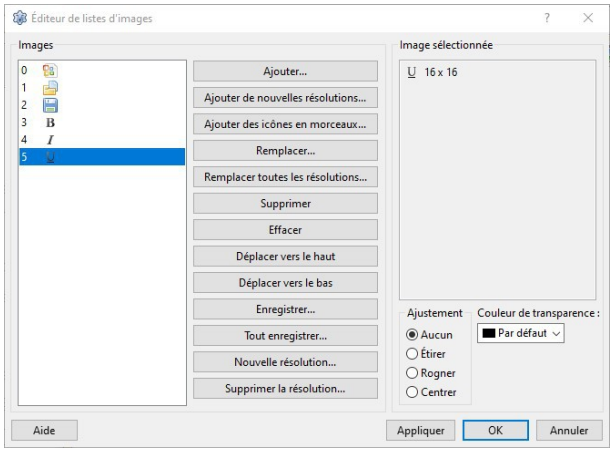
Nous allons maintenant préparer notre liste d'images. Le petit composant qui conservera les icônes de notre barre d'outils. **Faites un clic droit** dessus et sélectionnez **Éditeur de liste d'images…** (ou vous pouvez également **double-cliquer** dessus).



Ajoutez maintenant les images pour la barre d'outils de la manière suivante. J'ai utilisé les icônes Slik de FamFamFam.com. Vous pouvez également les télécharger (cliquez sur l'icône choisie et enregistrez-les). Voici les index et les noms des fichiers des icônes :

0. page_white_office.png
1. dossier_page.png
2. disk.png
3. text_bold.png
4. text_italic.png
5. text_underline.png

Votre boîte de dialogue **ImageList Editor** devrait ressembler à ceci :



Sélectionnez maintenant le composant **Toolbar1** et mettez sa propriété **Images** sur **ImageList1**. Cela nous permettra de définir les images de **ImageList** via l'index des icônes. Vous l'aurez en un instant.

(Voir les annexes, les icônes nécessaires pour tout le tutoriel peuvent y être directement copiées. Je vous ai également mis à disposition des icônes réalisées avec un logiciel tiers.)

II.D Construire la barre d'outils

Préparez maintenant les éléments de votre barre d'outils comme ceci :

1. **btnNew : TToolButton**

Cliquez avec le bouton droit sur la barre d'outils, créez un nouveau bouton de barre d'outils et nommez le **btnNew**.

Définissez maintenant ses propriétés comme suit :

- **Caption = New**
- **ImageIndex = 0** (Vous pouvez voir les icônes dans le menu déroulant de cette propriété. Il est très facile de comprendre quelle icône vous sélectionnez.)

Vous verrez probablement les icônes être coupées. C'est normal. C'est parce que la taille des boutons de la barre d'outils est très petite. Définissez donc la propriété **ButtonHeight** de **Toolbar1** sur **36**. Définissez également **AutoSize** sur **True**, ce qui définira automatiquement la hauteur de la barre d'outils.

2. **btnOpen : TToolButton**

- **Caption = Open**

- **ImageIndex = 1**

3. **btnSave : TToolButton**
   - **Caption = Save**
   - **ImageIndex = 2**

4. Cliquez avec le bouton droit sur la barre d'outils et cliquez sur **Nouveau séparateur**. Cela mettra une certaine distance entre les boutons.

5. Cliquez avec le bouton droit sur la barre d'outils et sélectionnez **Nouveau bouton à cocher** :
   - **btnBold : TToolButton**
   - **Caption = Bold**
   - **ImageIndex = 3**

Nouveau bouton à cocher créez un **TToolButton** mais définissez son style sur **tbsCheck**. Vous pouvez créer un nouveau bouton et modifier sa propriété **Style** manuellement. C'est le même.

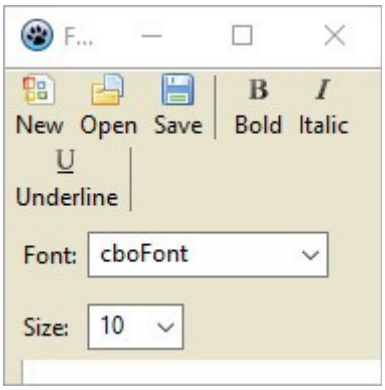
6. Encore une fois, **Nouveau bouton à cocher** :
   - **btnItalic: TToolButton**
   - **Caption = Italic**
   - **ImageIndex = 4**

7. **Nouveau bouton à cocher** :
   - **btnUnderline: TToolButton**
   - **Caption = Underline**
   - **ImageIndex = 5**

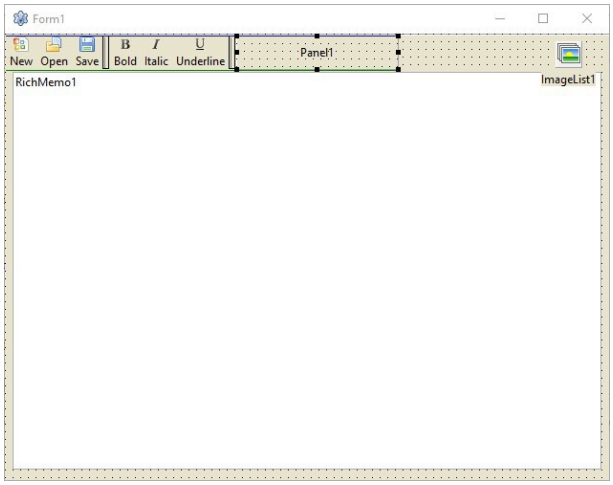
8. Clic droit sur la barre d'outils et cliquez sur **Nouveau séparateur**.

9. Nous devons maintenant créer une liste déroulante de sélecteur de police. Puisque **TCombobox** ne peut pas être créé avec un clic droit sur la barre d'outils, nous allons le

créer manuellement. Et nous inclurons également une étiquette indiquant « **Font** : ». Nous pouvons joliment rassembler l'étiquette et la liste déroulante dans un panneau (**Tpanel**) afin de pouvoir conserver un peu d'espace. Si nous créons ces deux éléments sans le panneau, il ne peut y avoir aucun espace entre les composants. De plus, lorsque vous redimensionnez le formulaire et que les boutons de la barre d'outils doivent être affichés sur deux lignes ou plus, l'ensemble du panneau reste groupé :



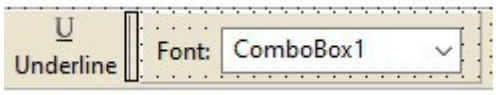
Donc, créez d'abord un **TPanel** à l'intérieur de la barre d'outils (lors du dessin, **commencez à dessiner depuis l'intérieur de la barre d'outils** pour créer le panneau en tant qu'enfant de la barre d'outils). Sa hauteur sera automatiquement définie par **ButtonHeight** de la barre d'outils.



Effacez **Caption**. Créez un **TLabel** à l'intérieur et saisissez dans **Caption** « **Font:** ».

Créez maintenant un **TComboBox** à l'intérieur. Nommez le **cboFont**. Vous verrez maintenant que

vous pouvez librement conserver un espace entre ces deux éléments et que vous pouvez même les centrer verticalement, ce qui aurait été impossible sans le panneau. Gardez la liste déroulante telle qu'elle est. Nous allons charger les polices avec le code d'aide plus tard.



Définissez **BevelOuter** sur **bvNone** du **panneau** pour supprimer la bordure.

10. Créez un autre **TPanel** dans la barre d'outils pour **Font Size**. Effacez **Caption** et mettez **BevelOuter** sur **bvNone**. Créez un **TLabel** à l'intérieur du panneau. Saisissez dans **Caption** « Size : ». Créez une **TcomboBox** et nommez le **cboFontSize** et dans **Items** saisissez :

    8

    10

    12

    14

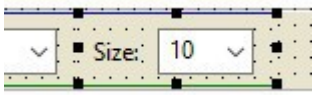
    16

    18

    26

    36

    72



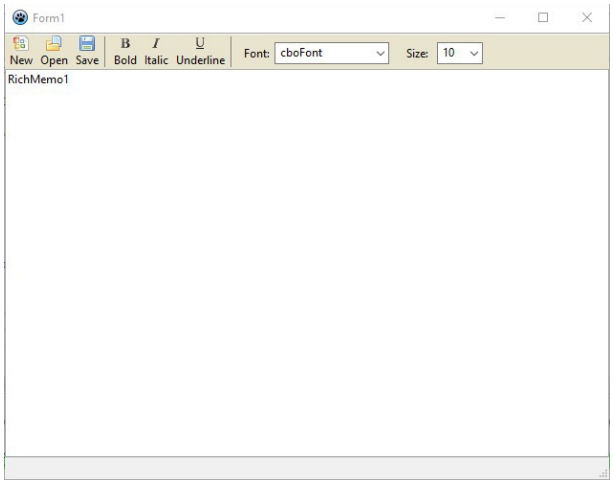
Vous pouvez éventuellement mettre **ItemIndex** à 1, pour définir la valeur par défaut.

La barre d'outils est terminée ! C'était la plus grande complexité de ce projet ! Vous êtes désormais prêt à affronter la suite du projet.

II.E Un premier aperçu

Maintenant, pour lui donner un aspect plus professionnel, posez un **TStatusBar** sur le formulaire (à partir de l'onglet **Common Controls**).Le RichMemo1 n'est pas dans une position exacte. Sélectionnez **RichMemo1** et définissez sa propriété **Align** sur **alClient**.

Allez-y et testez-le (**Exécuter->Exécuter ou F9**).



Cela a certainement l'air bien mais ce n'est pas fonctionnel. Alors mettons-nous au code !

II.F Le code

Nous allons maintenant donner vie à notre magnifique nouveau traitement de texte, avec du code.

II.F.1 Gestion de fichiers

Tout d'abord, le bouton **New**. Double-cliquez dessus et saisissez :

```
procedure TForm1.btnNewClick(Sender: TObject); // Nouveau document
begin
  RichMemo1.Clear;
  Saved:= False;
  Filename:= 'Untitled.rtf';
  Filepath:= '';
  Caption:= Filename;
end;
```

Ajoutez la variable suivante sous la **première clause var** :

```
 Filename: String = 'Untitled';
 Filepath: String;
 Saved: Boolean;
```

Ainsi, nous disons au **RichMemo** d'être effacé et nous définissons le booléen **Saved** sur **False**, ce

qui signifie que le nouveau document n'est pas enregistré. Nous pourrons donc plus tard utiliser cette valeur pour vérifier si le document est enregistré ou non. Peut-être qu'une boîte de dialogue « **Enregistrer ?** » serait bien, non ?

Chaque fois que l'utilisateur modifie le document, le booléen enregistré doit être défini sur **False**. Ainsi, lors de l'événement **OnChange** de **RichMemo1**, nous devrions le définir sur **False**. Alors, double-cliquez sur **RichMemo1** et entrez :

```
procedure TForm1.RichMemo1Change(Sender: TObject);
begin
  Saved:=False;
end;
```

Maintenant, c'est le bouton **Open**. Pour qu'une boîte de dialogue ouverte apparaisse, nous avons besoin de **TOpenDialog**. Créez-en un (à partir de l'onglet **Dialogs**).

Dans la propriété **DefaultExt** saisissez **.rtf** et **Filter** comme ceci : **RTF Files (*.rtf)|*.rtf**

Maintenant, double-cliquez sur **btnOpen** et entrez :

```
procedure TForm1.btnOpenClick(Sender: TObject); // Ouvrir un document
var
  fs : TFileStream;
begin
  if OpenDialog1.Execute then
    begin
      fs:= nil;
      try
        // Utf8ToAnsi est requis pour windows
        fs:= TFileStream.Create(Utf8ToAnsi(OpenDialog1.FileName),
fmOpenRead or fmShareDenyNone);
        RichMemo1.LoadRichText(fs);
        Saved:= True; // Puisque nous ouvrons un fichier enregistré
        Filename:= ExtractFileName(OpenDialog1.FileName);
        Filepath:= ExtractFilePath(OpenDialog1.FileName);
        Caption:= Filename;
      except
      end;
      fs.Free;
    end;
end;
```

Pour le bouton **Save**, créez un **TSaveDialog** (à partir de l'onglet **Dialogs**). Saisissez **.rtf** dans la propriété **DefaultExt** et dans **Filter** : **RTF Files (*.rtf)|*.rtf**

```
procedure TForm1.btnSaveClick(Sender: TObject); // Enregistrer le
document
var
  fs: TFileStream;
begin
```

```
  if SaveDialog1.Execute then
    begin
    fs:= nil;
    try
      fs:= TFileStream.Create(Utf8ToAnsi(SaveDialog1.FileName),
fmCreate);
      RichMemo1.SaveRichText(fs);
      Saved:= True;
      Filename:= ExtractFileName(SaveDialog1.FileName);
      Filepath:= ExtractFilePath(SaveDialog1.FileName);
      Caption:= Filename;
    except
    end;
    fs.Free;
    end;
end;
```

Ajoutez maintenant la procédure suivante à l'événement **OnCloseQuery** du composant **Form** (Sélectionnez **Form**, puis **Inspecteur d'objets -> Événements -> OnCloseQuery ->** […]) : [Si vous rencontrez des difficultés pour sélectionner **Form1**, sélectionnez-le dans l'**Inspecteur d'objets**.]

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose:
Boolean); // Procédure pour fermeture
var
  Response: Integer;
begin
  if not Saved then
    begin
      Response:= MessageDlg('Enregistrer ?', 'Souhaitez-vous
enregistrer ?', mtConfirmation, mbYesNoCancel,0);
      if Response = mrYes then
        begin
          btnSaveClick(Sender); // Enregistrer
          CanClose:= True; // et fermer
        end
      else
      if Response = mrNo then
        begin
          CanClose:= True; // Fermer sans enregistrer
        end
      else
      begin
        CanClose:= False; // Annuler la fermeture
      end;
    end;
end;
```

II.F.2 Autres boutons de la barre d'outils

Maintenant que la gestion de base des fichiers est terminée, nous pouvons nous concentrer sur les

parties les plus amusantes. Nous avons la liste des polices. Nous devons charger les polices automatiquement dans la **Combobox**. Ainsi, dans la procédure événementielle **OnCreate** du composant **Form**, nous entrons :

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  cboFont.Items.Assign(Screen.Fonts); // Chargement des polices de
texte
  cboFont.ItemIndex:= 0; // Première police sélectionnée
end;
```

Nous allons maintenant créer une procédure pour mettre à jour la barre d'outils en fonction de l'endroit où se trouve le curseur. Mais d'abord, ajoutez la variable à la **première clause var de l'unité** :

```
var
  ...
  SelFontFormat: TFontParams;
```

Les informations de formatage seront stockées dans cette variable. Pour changer le formatage, **(1)** nous devrons d'abord obtenir le formatage dans une variable **TFontParams**, **(2)** puis le modifier et **(3)** l'appliquer à la sélection. Par exemple, pour faire une sélection en gras, nous devrons obtenir le formatage au point de départ de la sélection, le mettre en gras, puis définir toute la sélection avec le nouveau formatage. Ce formatage peut être utilisé dans de nombreuses procédures (italique, souligné, etc.), nous l'avons donc ajouté dans la clause **var**.

Puis la procédure :

```
procedure TForm1.PrepareToolbar(); // Procédure de préparation de la
barre d'outils
begin
 cboFont.Caption:= SelFontFormat.Name;
 cboFontSize.Caption:= inttostr(SelFontFormat.Size);

 if (fsBold in SelFontFormat.Style = true) then
   btnBold.Down:= True // Mise en gras du texte
 else
   btnBold.Down:= False;

  if (fsItalic in SelFontFormat.Style = true) then
    btnItalic.Down:= True // Mise en italique du texte
  else
    btnItalic.Down:= False;

  if (fsUnderline in SelFontFormat.Style = true) then
    btnUnderline.Down:= True // Soulignement du texte
  else
    btnUnderline.Down:= False;
```

```
end;
```

Copier-coller la procédure ci-dessus sous la clause d'implémentation. Une déclaration à terme sera ajoutée sous les déclarations de **Form1**.

Ajoutez la procédure suivante à l'événement **OnMouseDown** de **RichMemo1** (sélectionnez **RichMemo1**, puis **Inspecteur d'objets -> Événements -> OnMouseDown -> […]**) :

```
procedure TForm1.RichMemo1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  RichMemo1.GetTextAttributes(RichMemo1.SelStart, SelFontFormat);
  PrepareToolbar;
end;
```

Lors du changement de position du curseur, nous récupérons/mettons à jour le formatage dans **SelFontFormat**.

Si vous ouvrez maintenant un **RTF** et cliquez par endroits, vous verrez que la barre d'outils sera mise à jour (le nom de la police, gras, italique, souligné, etc. change). Ainsi, il peut lire le style. Nous allons maintenant préparer les styles.

- **BOUTON BOLD** (gras)

  Double-cliquez sur **btnBold** et saisissez :

```
procedure TForm1.btnBoldClick(Sender: TObject); // Texte en gras
begin
  if (fsBold in SelFontFormat.Style = False) then
    SelFontFormat.Style:=SelFontFormat.Style + [fsBold]
  else
    SelFontFormat.Style:=SelFontFormat.Style - [fsBold];
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength, SelFontFormat);
end;
```

Nous avons déjà **(1)** obtenu le formatage dans **SelFontFormat**, (2) nous modifions le style de formatage et y ajoutons **fsBold** (ou le repassons à normal s'il est déjà en gras), **(3)** puis l'appliquons à la sélection.

Maintenant, exécutez-le (**F9 ou Exécuter->Exécuter**). Sélectionnez un texte et appuyez sur **Bold**. Voilà votre premier bouton de formatage fonctionnel ! Je vois déjà un petit sourire sur votre visage.

- **BOUTON ITALIC** (italique)

  Maintenant, double-cliquez sur **btnItalic** et entrez :

```
procedure TForm1.btnItalicClick(Sender: TObject); // Texte en italique
begin
  if (fsItalic in SelFontFormat.Style = False) then
    SelFontFormat.Style:=SelFontFormat.Style + [fsItalic]
  else
    SelFontFormat.Style:=SelFontFormat.Style - [fsItalic];
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength, SelFontFormat);
end;
```

- **BOUTON UNDERLINE** (souligné)

  Maintenant, double-cliquez sur **btnUnderline** et entrez :

```
procedure TForm1.btnUnderlineClick(Sender: TObject); // Texte souligné
begin
  if (fsUnderline in SelFontFormat.Style = False) then
    SelFontFormat.Style:= SelFontFormat.Style + [fsUnderline]
  else
    SelFontFormat.Style:= SelFontFormat.Style - [fsUnderline];
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength,SelFontFormat);
end;
```

- **FONT LIST** (liste des polices)

  Maintenant, dans la procédure événementielle **OnSelect** de **cboFont**, entrez ceci :

```
procedure TForm1.cboFontSelect(Sender: TObject); // Sélection police de
texte
begin
  SelFontFormat.Name:= cboFont.Text;
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength,SelFontFormat);
  RichMemo1.SetFocus; // Focus sur RichMemo
end;
```

- **FONT SIZE** (taille de police)

  Pour modifier la taille de la police, ajoutez la procédure suivante sur l'événement **OnSelect** de **cboFontSize** :

```
procedure TForm1.cboFontSizeSelect(Sender: TObject); // Sélection
taille police de texte
begin
  SelFontFormat.Size:= StrToInt(cboFontSize.Text);
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength, SelFontFormat);
end;
```

II.F.3 Retouche

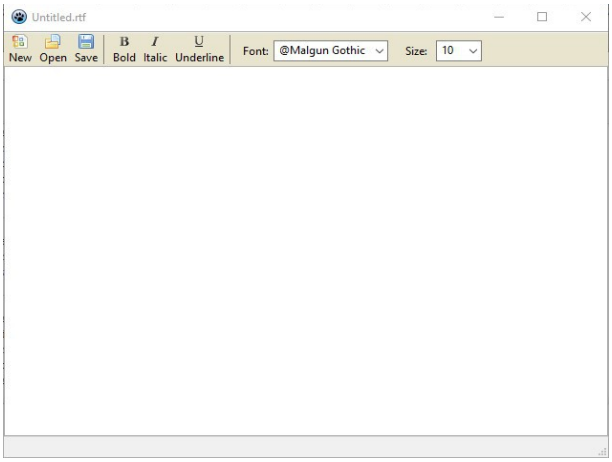
Ajoutez la ligne suivante à la fin de la procédure OnCreate de Form1 :

```
btnNewClick(Sender); // Ouverture sur un nouveau document
```

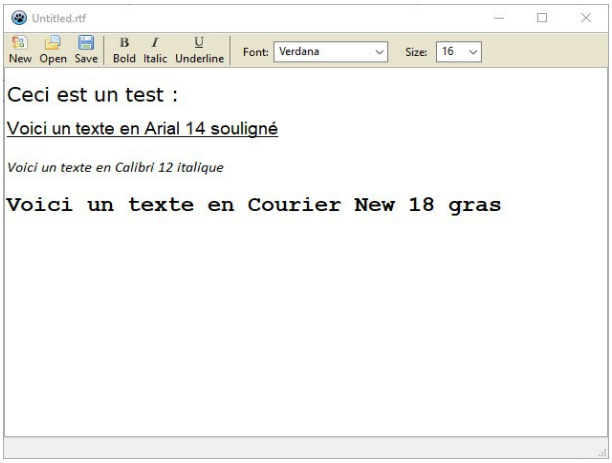
Cela fera comme si l'utilisateur avait cliqué sur **btnNew** au démarrage du formulaire. Il créera donc un nouveau document, définira correctement la légende du formulaire et certaines variables (nom de fichier, enregistré, etc.).

En option, vous pouvez ajouter des menus si vous souhaitez lui donner un aspect plus professionnel. Vous devez dessiner un **TMainMenu**, double-cliquer dessus et ajouter des menus. Lorsque vous sélectionnez un élément, l'inspecteur d'objets vous permet de modifier ses propriétés, d'y ajouter un événement de clic, etc.

Maintenant, exécutez-le (**F9 ou Exécuter->Exécuter**).



Tapez, ouvrez des documents et testez.



II.F.4 Autres améliorations

Améliorer n'importe quel code source est l'un des meilleurs moyens de mettre en pratique ses compétences en programmation. Vous pouvez améliorer le projet de plusieurs manières :

- **Commandes de menus** :

  Vous pouvez créer des menus vous-même.

- **Défaire - refaire** :

  Il existe une fonction d'annulation et de restauration intégrée que vous pouvez utiliser. Ajoutez des boutons dans la barre d'outils et faites-les glisser pour les réorganiser si nécessaire.

- **Couleur du texte :**

  Vous pouvez facilement changer la couleur via **SelFontFormat.Color**, tout comme les boutons de formatage.

- **Barré** en utilisant **fsStrikeOut**, tout comme nous avons utilisé **fsBold**.

- Fonction **glisser-déposer** de fichiers.

- **Menu contextuel** permettant de couper, copier, coller, supprimer, etc.

- Vous pouvez utiliser **CutToClipboard**, **CopyToClipboard** et **PasteFromClipboard** de **TRichMemo**.

II.F.5 Problèmes connus

Il s'agit simplement d'une implémentation de base de **RichMemo**. Il y a quelques problèmes. Les boutons de changement de formatage ne fonctionnent pas bien lorsque plusieurs styles sont appliqués à la sélection. Par exemple, si votre sélection contient du texte en gras et en italique et que vous cliquez sur souligné, le style gras et italique sera réinitialisé et la sélection sera soulignée. Vous pouvez utiliser **RichMemo1.GetStyleRange** pour obtenir chaque style et le modifier en conséquence. Vous pouvez consulter l'exemple de projet inclus avec le package de composants. Il dispose d'un bouton « Next Style Range » qui détecte chaque style différent. Le code est celui-ci :

```
procedure TForm1.Button6Click(Sender: TObject);
var
  ofs,len: Integer;
begin
  RichMemo1.GetStyleRange( RichMemo1.SelStart, ofs, len );
  if (ofs = RichMEmo1.SelStart) and (len = RichMemo1.SelLength)then
    begin
      ofs := ofs + len;
      RichMemo1.GetStyleRange( ofs, ofs, len );
    end;
```

```
  RichMemo1.SelStart := ofs;
  RichMemo1.SelLength := len;
end;
```

Quand vous fermez le programme avec un document non enregistré, cliquez sur **Yes**, puis dans la boîte de dialogue appuyez sur **Annuler**, le programme se ferme et le document est perdu !

III Partie 2

Cette partie est consacrée à présenter et mettre en œuvre quelques améliorations.

III.A Correction du code de sélection de taille de la police

En effet, lorsqu'on sélectionne le texte pour lequel on effectue un changement de police, celui-ci reste en surbrillance. Pour corriger, il faut procéder comme suit :

```
procedure TForm1.cboFontSizeSelect(Sender: TObject); // Sélection
taille police de texte
begin
  SelFontFormat.Size:= StrToInt(cboFontSize.Text);
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength, SelFontFormat);
  RichMemo1.SetFocus; // A rajouter pour correction bug (sinon texte
reste en surbrillance)
end;
```

III.B Barrer le texte

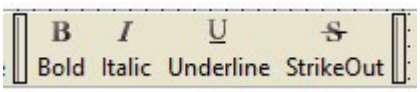
Nous procédons de la même manière que pour le bouton pour mettre en gras le texte.

Récupérez l'icône pour le bouton : ici

Créez un nouveau **bouton à cocher** :

- **Caption : StrikeOut**
- **Name : btnStrikeOut**

Glissez le bouton à côté de celui pour souligner et appliquez l'icône.



Au niveau de la procédure **PrepareToolbar** rajoutez ceci :

```
  if (fsStrikeOut in SelFontFormat.Style = true) then
    btnStrikeOut.Down:=True // Barrer le texte
  else
    btnStrikeOut.Down:=False;
```

Double-cliquez sur le bouton **StrikeOut** et saisissez :

```
procedure TForm1.btnStrikeOutClick(Sender: TObject); // Texte barré
begin
  if (fsUnderline in SelFontFormat.Style = False) then
    SelFontFormat.Style:=SelFontFormat.Style + [fsStrikeOut]
  else
    SelFontFormat.Style:=SelFontFormat.Style - [fsStrikeOut];
  RichMemo1.SetTextAttributes(RichMemo1.SelStart,
  RichMemo1.SelLength,SelFontFormat);
end;
```

III.C Aligner le texte

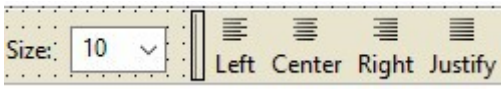
Faites un clic droit sur la barre d'outils et cliquez sur **Nouveau séparateur**. Déposer le séparateur après le sélecteur de la taille de police.

Créez **quatre boutons à cocher** après le séparateur.

- Bouton pour **aligner le texte à gauche** :
  - **Caption : Left**
  - **Name : btnLeft**
- Bouton pour **centrer le texte :**
  - **Caption : Center**
  - **Name : btnCenter**
- Bouton pour **aligner le texte à droite** :
  - **Caption : Right**
  - **Name : btnRight**
- Bouton pour **justifier le texte :**
  - **Caption : Justify**
  - **Name : btnJustify**

Récupérez les icônes adéquates et chargez-les dans **ImageList1**, puis appliquez-les aux boutons.

Voilà comment cela doit se présenter :



Passons au code :

- Dans la procédure **PrepareToolbar**, ajoutez à la suite du code saisi précédemment :

```
  if RichMemo1.SelText <>'' then
    btnLeft.Down:= True // Alignement du texte sélectionné à gauche
  else
    btnLeft.Down:= False;

  if RichMemo1.SelText <>'' then
    btnCenter.Down:= True // Alignement du texte sélectionné au centre
  else
    btnCenter.Down:= False;

  if RichMemo1.SelText <>'' then
    btnRight.Down:= True // Alignement du texte sélectionné à droite
  else
    btnRight.Down:= False;

  if RichMemo1.SelText <>'' then
    btnJustify.Down:= True // Justification du texte sélectionné
  else
    btnJustify.Down:= False;
```

- Double-cliquez sur le bouton **Left** et saisissez :

```
procedure TForm1.btnLeftClick(Sender: TObject); // Texte à gauche
begin
  If RichMemo1.GetParaAlignment(RichMemo1.SelStart) <> RichMemo.paLeft
then

RichMemo1.SetParaAlignment(RichMemo1.SelStart,RichMemo1.SelLength,RichM
emo.paLeft);
end;
```

- Double-cliquez sur le bouton **Center** et saisissez :

```
procedure TForm1.btnCenterClick(Sender: TObject); // Texte centré
begin
  If RichMemo1.GetParaAlignment(RichMemo1.SelStart) <>
RichMemo.paCenter then

RichMemo1.SetParaAlignment(RichMemo1.SelStart,RichMemo1.SelLength,RichM
emo.paCenter);
```

```
end;
```

- Double-cliquez sur le bouton **Right** et saisissez :

```
procedure TForm1.btnRightClick(Sender: TObject); // Texte à droite
begin
  If RichMemo1.GetParaAlignment(RichMemo1.SelStart) <> RichMemo.paRight
then

RichMemo1.SetParaAlignment(RichMemo1.SelStart,RichMemo1.SelLength,RichM
emo.paRight);
end;
```

- Double-cliquez sur le bouton **Justify** et saisissez :

```
procedure TForm1.btnJustifyClick(Sender: TObject); // Texte justifié
begin
  If RichMemo1.GetParaAlignment(RichMemo1.SelStart) <>
RichMemo.paJustify then

RichMemo1.SetParaAlignment(RichMemo1.SelStart,RichMemo1.SelLength,RichM
emo.paJustify);
end;
```

Pour utiliser ces boutons : compilez, écrivez quelques mots et sélectionnez les en les mettant en surbrillance, cliquez sur le bouton de votre choix, le texte prend alors l'alignement choisi.

III.D Changer la couleur du texte

Il faut créer un bouton comme nous avons procédé pour les trois premiers boutons et déposer un composant **TColorDialog** sur **Form1**.

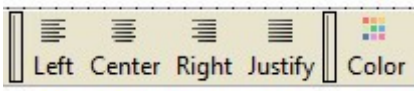
Le bouton :

- **Caption : Color**
- **Name : btnColor**

Insérez un séparateur après le bouton pour justifier le texte.

Récupérez l'icône adéquate et chargez-la dans **ImageList1**, puis appliquez-la au bouton.

Voilà comment cela doit se présenter :



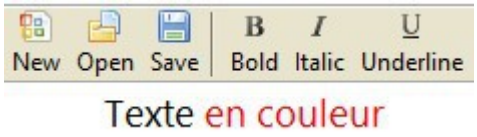
Double-cliquez sur le bouton **Color** et saisissez :

```
procedure TForm1.btnColorClick(Sender: TObject); // Couleur du texte
begin
  if ColorDialog1.Execute then

RichMemo1.SetRangeColor(RichMemo1.SelStart,RichMemo1.SelLength,ColorDia
log1.Color);
  RichMemo1.SetFocus; // Focus sur RichMemo
end;
```

Pour utiliser ce bouton : compilez, écrivez un texte et sélectionnez une partie de texte en la mettant en surbrillance, cliquez sur le bouton, puis sur la couleur dans **ColorDialog1**, elle prend alors la couleur choisie.

Voilà ce que ça peut donner :



III.E Surligner le texte

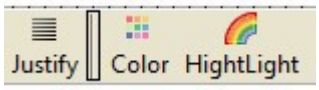
Tout d'abord, ajoutez **RichEdit** dans les clauses **uses**. Cela permet d'obtenir d'autres fonctionnalités pour le composant **RichMemo1**.

Comme précédemment, créez un bouton :

- **Caption : HightLight**
- **Name : btnHightLight**

Récupérez l'icône adéquate et chargez-la dans **ImageList1**, puis appliquez-la au bouton.

Placer le bouton après le bouton créé précédemment comme ceci :



Il faut ensuite créer la procédure **RMHightLight** pour surligner le texte saisi.

D'abord, déclarez la variable **CHARFORMAT2** la procédure **RMHightLight** sous **private** :

```
  private
    Format: CHARFORMAT2;
    source: String;
    procedure PrepareToolbar();
```

```
    procedure RMHightLight(RichMemo: TRichMemo; BackColor: TColor);
```

Voici la procédure **RMHightLight** :

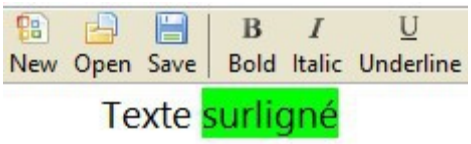
```
procedure TForm1.RMHightLight(RichMemo: TRichMemo; BackColor:
TColor); // Surligner la partie de texte sélectionné
begin
  FillChar(Format, SizeOf(Format), 0);
  with Format do
  begin
    cbSize := SizeOf(Format);
    dwMask := CFM_BACKCOLOR;
    crBackColor := BackColor;
    RichMemo.Perform(EM_SETCHARFORMAT, SCF_SELECTION,
longint(@Format));
  end;
  RichMemo.SelLength := 0;
end;
```

Double-cliquez sur le bouton **HightLight** et saisissez :

```
procedure TForm1.btnHightLightClick(Sender: TObject); // Texte surligné
begin
  if ColorDialog1.Execute then
    RMHightLight(RichMemo1,ColorDialog1.Color);
  RichMemo1.SetFocus; // Focus sur RichMemo
end;
```

Pour utiliser ce bouton : compilez, écrivez un texte et sélectionnez une partie de texte en la mettant en surbrillance, cliquez sur le bouton, puis sur la couleur dans **ColorDialog1**, elle est alors surlignée avec la couleur choisie.

Voilà ce que ça peut donner :



III.F Copier-couper-coller

Pour pouvoir utiliser le presse-papiers, il faut mettre **Clipbrd** dans les clauses **uses**.

Mettez un séparateur après le bouton **HigthLight**, puis à la suite créez dans la barre d'outils trois boutons de la même façon que vous avez l'avez fait pour les précédents.

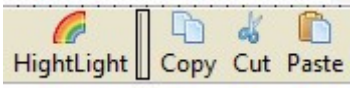
- Bouton **copier** :
    - **Caption : Copy**

- **Name : btnCopy**

- Bouton **couper** :
  - **Caption : Cut**
  - **Name : btnCut**
- Bouton **coller** :
  - **Caption : Paste**
  - **Name : btnPaste**

Récupérez les icônes adéquates et chargez-les dans **ImageList1**, puis appliquez-les aux boutons.

Voilà comment cela doit se présenter :



D'abord déclarez la variable **source** et les procédures **Copier**, **Couper** et **Coller** sous **Private** :

```
  private
    source: String;
    procedure PrepareToolbar();
    procedure RMHightLight(RichMemo: TRichMemo; BackColor: TColor);
    procedure Copier(destination: TRichMemo);
    procedure Couper(destination: TRichMemo);
    procedure Coller(destination: TRichMemo);
```

Ensuite, créez les procédures comme suit :

– la procédure pour **copier** :

```
procedure TForm1.Copier(destination: TRichMemo); // Procédure pour
copier
begin
  // Vérifier si le texte est sélectionné dans une source autre que le
TMemo
  if source <> '' then
    begin
      // Copier le texte sélectionné
      Clipboard.AsText := source;
      destination.CopyToClipboard;
    end
  else
  // Vérifier si le texte est sélectionné dans le TMemo
  if RichMemo1.SelLength > 0 then
    begin
      // Copier le texte sélectionné
```

```
      Clipboard.AsText := RichMemo1.SelText;
      destination.CopyToClipboard;
    end
  else
    // Vérifier si c'est une image sélectionnée
    if Clipboard.HasPictureFormat then
      // Copier l'image sélectionnée
      destination.CopyToClipboard;
end;
```

– la procédure pour **couper** :

```
procedure TForm1.Couper(destination: TRichMemo); // Procédure pour
couper
begin
  // Vérifier si le texte est sélectionné dans le TMemo
  if RichMemo1.SelLength > 0 then
    begin
      // Copier le texte sélectionné et l'effacer
      Clipboard.AsText := RichMemo1.SelText;
      destination.CopyToClipboard;
      RichMemo1.SelText := '';
    end
  else
  // Vérifier si c'est une image sélectionnée dans le TMemo
  if Clipboard.HasPictureFormat then
    begin
      // Copier l'image sélectionnée et l'effacer
      destination.CopyToClipboard;
      RichMemo1.ClearSelection;
    end;
end;
```

– la procédure pour **coller** :

```
procedure TForm1.Coller(destination: TRichMemo); // Procédure pour
coller
begin
  // Coller le texte du presse-papiers dans le TMemo
  if Clipboard.AsText <>'' then
    destination.PasteFromClipboard
  else
  // Coller l'image du presse-papiers dans le TMemo
  if Clipboard.HasPictureFormat then
    destination.PasteFromClipboard;
end;
```

Appliquez les procédures aux boutons :

– Double-cliquez sur le bouton **Copy** et saisissez :

```
procedure TForm1.btnCopyClick(Sender: TObject); // Copier
begin
  Copier(RichMemo1);
end;
```

– Double-cliquez sur le bouton **Cut** et saisissez :

```
procedure TForm1.btnCutClick(Sender: TObject); // Couper
begin
  Couper(RichMemo1);
end;
```

– Double-cliquez sur le bouton **Paste** et saisissez :

```
procedure TForm1.btnPasteClick(Sender: TObject); // Coller
begin
  Coller(RichMemo1);
end;
```

Les procédures permettent de copier-coller non-seulement du texte mais aussi des images. Le texte copié peut venir d'une source externe mais aussi de mots saisis dans **RichMemo1**. Il en va de même pour les images. Vous remarquerez que pour le copier-coller du texte de **RichMemo1** à **RichMemo1**, il a été nécessaire d'insérer dans la procédure un bout de code spécifique pour que ça fonctionne. Quant aux images collées, il est possible d'en modifier les dimensions.

Maintenant, compilez et essayez.

III.G Défaire-refaire

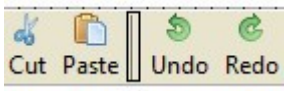
Vous pouvez mettre aussi en place la fonctionnalité **Undo-Redo**, c'est-à-dire **défaire-refaire**. Cela permet d'annuler et puis éventuellement rétablir. Par exemple je supprime une partie de texte puis j'annule la suppression et enfin je reviens sur ma décision et je supprime à nouveau.

Mettez un séparateur après le bouton **Paste**, puis à la suite créez dans la barre d'outils deux boutons de la même façon que vous avez l'avez fait pour les précédents :

- Bouton **défaire** :
  - **Caption : Undo**
  - **Name : btnUndo**
- Bouton **refaire** :
  - **Caption : Redo**
  - **Name : btnRedo**

Récupérez les icônes adéquates et chargez-les dans **ImageList1**, puis appliquez-les aux boutons.

Voilà comment cela doit se présenter :



Afin de mettre en œuvre la fonctionnalité, il faut créer deux listes pour enregistrer d'une part les actions à défaire (à annuler) et d'autre part les actions à refaire. Il faut leur associer deux procédures pour les alimenter ainsi qu'une pour défaire et une autre pour refaire.

Déclarez les listes et procédures sous **private** :

```
  Private
    ...
    UndoList: TStringList;
    RedoList: TStringList;
    ...
    procedure AddToUndoList(const Texte: string);
    procedure AddToRedoList(const Texte: string);
    procedure Undo;
    procedure Redo;
```

Procédure de création des listes :

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ...
  UndoList := TStringList.Create; // Création de la liste "Défaire"
  RedoList := TStringList.Create; // Création de la liste "Refaire"
end;
```

Procédure d'alimentation de la liste « **Défaire** » :

```
procedure TForm1.AddToUndoList(const Texte: string); // Alimentation de
la liste "Défaire"
begin
  UndoList.Add(Texte);
  btnUndo.Enabled := UndoList.Count > 0;
  btnRedo.Enabled := False;
end;
```

Procédure d'alimentation de la liste « **Refaire** » :

```
procedure TForm1.AddToRedoList(const Texte: string); // Alimentation de
la liste "Refaire"
begin
```

```
  RedoList.Add(Texte);
  btnRedo.Enabled := RedoList.Count > 0;
  btnUndo.Enabled := False;
end;
```

Procédure pour **Défaire** :

```
procedure TForm1.Undo; // Procédure "Défaire"
begin
  if UndoList.Count > 0 then
  begin
    AddToRedoList(RichMemo1.Text);
    RichMemo1.Text := UndoList[UndoList.Count - 1];
    UndoList.Delete(UndoList.Count - 1);
    btnRedo.Enabled := True;
    btnUndo.Enabled := UndoList.Count > 0;
  end;
end;
```

Procédure pour **Refaire** :

```
procedure TForm1.Redo; // Procédure "Refaire"
begin
  if RedoList.Count > 0 then
  begin
    AddToUndoList(RichMemo1.Text);
    RichMemo1.Text := RedoList[RedoList.Count - 1];
    RedoList.Delete(RedoList.Count - 1);
    btnUndo.Enabled := True;
    btnRedo.Enabled := RedoList.Count > 0;
  end;
end;
```

Pour initialiser la fonctionnalité, il est nécessaire qu'une action soit enregistrée dans la liste « **Défaire** ». Cela doit se faire sur l'événement **OnChange** de **RichMemo1** :

```
procedure TForm1.RichMemo1Change(Sender: TObject);
begin
  Saved:= False;
  AddToUndoList(RichMemo1.Text); // Initialisation de "Défaire-Refaire"
end;
```

Appliquez les procédures aux boutons :

- Double-cliquez sur le bouton **Undo** et saisissez :

```
procedure TForm1.btnUndoClick(Sender: TObject);
begin
  Undo; // Défaire
```

```
end;
```

– Double-cliquez sur le bouton **Redo** et saisissez :

```
procedure TForm1.btnRedoClick(Sender: TObject);
begin
  Redo; // Refaire
end;
```

Lors de l'enregistrement du document, il n'est plus nécessaire de conserver les listes. Il convient donc de les effacer et ainsi libérer de la mémoire. Pour cela, il faut insérer les lignes de code au niveau de la procédure d'enregistrement.

```
procedure TForm1.btnSaveClick(Sender: TObject); // Enregistrer le
document
var
  fs: TFileStream;
begin
  if SaveDialog1.Execute then
    begin
    fs:= nil;
    try
      ...
      // Vider les listes
      UndoList.Clear;
      RedoList.Clear;
    except
    end;
    fs.Free;
    end;
end;
```

Maintenant vous pouvez compiler et essayer.

ⓘ Pour utiliser la fonctionnalité **Undo-Redo**, il peut être nécessaire de cliquer plusieurs fois sur les boutons.

III.H Lien hypertexte

Il est possible d'insérer un lien hypertexte dans le texte que vous saisissez.

Pour mettre en place cette fonctionnalité, il faut commencer par mettre **Lclintf** dans les clauses **uses** du projet.

Ensuite, posez un séparateur après le bouton **Redo**, puis créez dans la barre d'outils un bouton de

la même façon que vous avez l'avez fait pour les précédents.

Le bouton :

- **Caption : Link**
- **Name : btnLink**

Récupérez l'icône adéquate et chargez-la dans **ImageList1**, puis appliquez-la au bouton.

Placer le bouton après le séparateur :

Double-cliquez sur le bouton **Link** et saisissez :

```
procedure TForm1.btnLinkClick(Sender: TObject); // Mettre en lien
hypertexte
begin
  RichMemo1.SetLink(RichMemo1.SelStart,RichMemo1.SelLength,True);
end;
```

Pour que le lien soit actif, il faut saisir dans l'événement **OnLinkAction** de **RichMemo1** ceci :

```
procedure TForm1.RichMemo1LinkAction(Sender: TObject; ALinkAction:
TLinkAction;
  const info: TLinkMouseInfo; LinkStart, LinkLen: Integer); // Action
du lien hypertexte
begin
  if RichMemo1.isLink(RichMemo1.SelStart) then
    OpenDocument(RichMemo1.SelText);
end;
```

Pour essayer, compilez et collez un lien de page Internet dans le **RichMemo1**, puis sélectionnez-le en le mettant en surbrillance et cliquez sur le bouton **Link**. Le lien est alors mis en bleu et souligné. Ensuite, pour ouvrir la page Internet choisie, mettez en surbrillance le lien et cliquez dessus. La page qui aura été préalablement refermée s'ouvre. C'est le navigateur Internet par défaut qui est utilisé.

ⓘ En utilisant cette méthode, il est aussi possible d'insérer un lien qui peut être le chemin d'un dossier ou d'un fichier.

III.I Imprimer

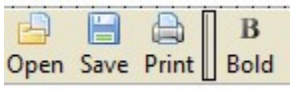
Il faut créer un bouton comme nous avons procédé pour le bouton précédent et déposer un composant **TPrintDialog** sur **Form1**.

Le bouton :

- **Caption : Print**
- **Name : btnPrint**

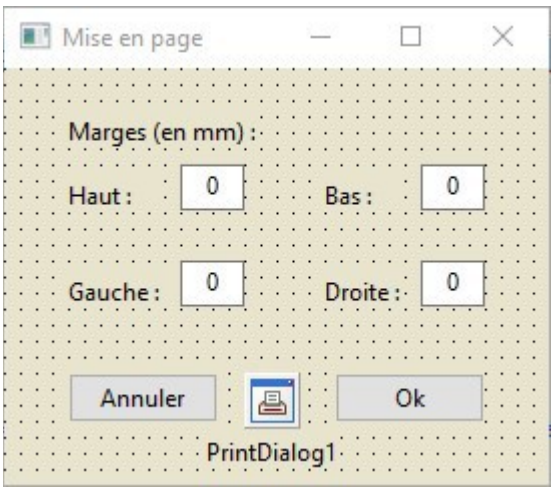
Récupérez l'icône nécessaire et chargez-la dans **ImageList1**, puis appliquez-la au bouton.

Insérez le bouton après celui pour enregistrer :



Afin de pouvoir réaliser une mise en page avant impression il faut créer une boite de dialogue qui permettra de saisir les marges du document. Pour faire cela :

- Créez un nouveau formulaire **Form2** et enregistrez la nouvelle unité **mpimp**.
- Dans les options du projet, placez **Form2** dans les **fiches disponibles**.
- Sur **Form2**, posez 5 **TLabel**, 4 **TEdit**, 1 **TPrintDialog** et 2 **TButton** pour réaliser ceci :



Pour les marges, nommez les composants **TEdit** :

Haut :

- **Name : EdH**
- **Alignment : taCenter**

Bas :

- **Name : EdB**
- **Alignment : taCenter**

Gauche :

- **Name : EdG**
- **Alignment : taCenter**

Droite :

- **Name : EdD**
- **Alignment : taCenter**

Pour les boutons :

Le bouton **Ok** :

- **Caption : Ok**
- **Name : btnOk**

Le bouton **Annuler** :

- **Caption : Annuler**
- **Name : btnAn**

Double-cliquez sur le bouton **Print** et saisissez :

```
procedure TForm1.btnPrintClick(Sender: TObject); // Imprimer
begin
  Form2:= TForm2.Create(Application);
  Form2.ShowModal;
end;
```

Un clic sur le bouton **Print** ouvre la boite de dialogue pour l'impression.

Pour pouvoir ouvrir la boite de dialogue, il faut mettre l'unité **mpimp** au niveau de **Form1** dans la clause **uses** ici :

```
implementation

{$R *.lfm}

{ TForm1 }
```

```
uses
  mpimp;
```

Il faut aussi que les clauses **uses** de **Form2** soient celles-ci :

```
interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
PrintersDlgs, RichMemo;
```

Pour assurer le lien avec Form1, il faut mettre l'unité **main** au niveau de **Form2** dans la clause **uses** ici :

```
implementation

{$R *.lfm}

{ TForm2 }

uses
  main;
```

Au niveau du formulaire d'impression, la saisie des marges doit se faire en millimètres dans les composants **TEdit**.

Lorsque le composant **TPrintDialog** est sollicité, l'utilisateur peut choisir l'imprimante physique ou virtuelle avec laquelle il désire imprimer le document.

Les marges définies, l'utilisateur lance l'impression en cliquant sur le bouton **Ok**.

```
procedure TForm2.btnOkClick(Sender: TObject); // Imprimer
Var
  print_parameters: TPrintParams;
begin
  B:= StrToInt(EdB.Text);
  D:= StrToInt(EdD.Text);
  G:= StrToInt(EdG.Text);
  H:= StrToInt(EdH.Text);
    try
    if not PrintDialog1.Execute then
      Exit;
    // Paramètres d'impression (*2,83 => conversion des points en
millimètres)
    print_parameters.Margins.Top := H*2.83;
    print_parameters.Margins.Bottom := B*2.83;
    print_parameters.Margins.Left := G*2.83;
    print_parameters.Margins.Right := D*2.83;
    Form1.RichMemo1.Print(print_parameters); // Impression
  finally
```

```
    PrintDialog1.Free;
    Form2.Close;
  end;
end;
```

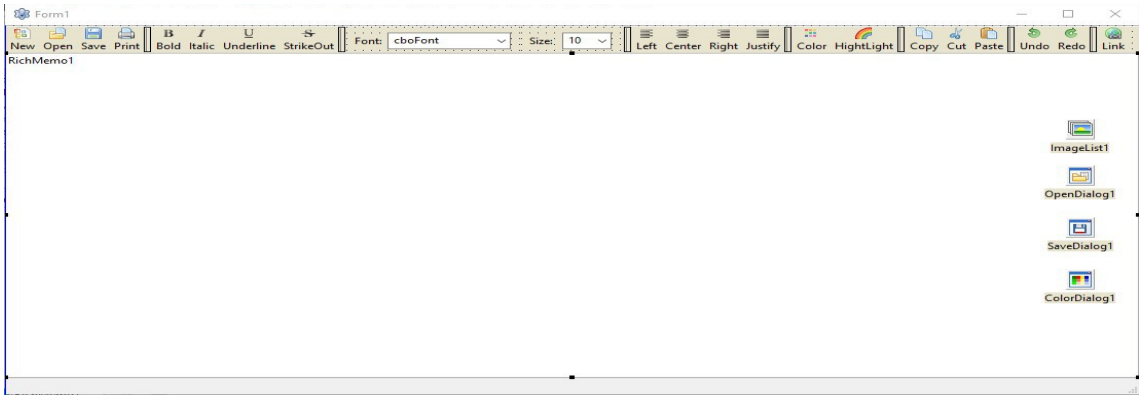
Le bouton **Annuler** permet d'annuler l'impression et de revenir sur l'éditeur de texte.

```
procedure TForm2.btnAnClick(Sender: TObject); // Annuler l'impression
begin
  Close;
end;
```

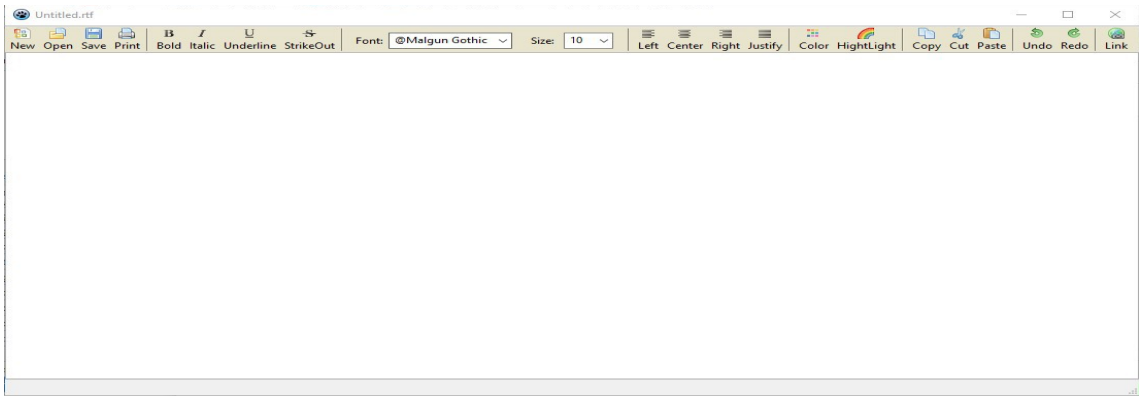
IV Conclusion

Cher lecteur et utilisateur, voilà ce à quoi vous êtes en principe parvenu :

Avant compilation



Après compilation



Ce tutoriel vous a guidé pour fabriquer un éditeur de texte. Mais, est-ce qu'aujourd'hui cela a encore du sens quand on voit le panel d'éditeurs de texte proposés tant à l'achat que gratuits. À mon avis, l'intérêt d'un tel tutoriel réside plutôt dans l'apprentissage de l'utilisation de certains composants et des procédures mises en œuvre. Ainsi, vous pourrez peut-être vous en servir dans vos prochaines créations.

Si les icônes proposées ne vous conviennent pas, il est possible de créer vos propres icônes notamment avec l'application développée en pascal et gratuite **Greenfish Icon Editor Pro** que vous pouvez télécharger ici.

V Annexes

V.A Les icônes Slik de FamFamFam.com du tutoriel


Nouveau


Ouvrir


Enregistrer


Gras


Italique


Souligné


Barré


À gauche


Au centre


À droite


Justifié


Couleur


Surligné


Copier


Couper


Coller


Défaire


Refaire


Lien


Imprimer

V.B Les icônes réalisées avec Greenfish Icon Editor Pro


Nouveau


Ouvrir


Enregistrer


Gras


Italique


Souligné


Barré


À gauche


Au centre


À droite

Justifié

Couleur

Surligné

Copier

Couper

Coller

Défaire

Refaire

Lien

Imprimer