Eléments de programmation en Delphi

Plan de la leçon

1. Les deux types d'informations traitées par un ordinateur
2. De l'humain à la machine
3. Principe du travail avec Delphi Pascal
4. Structure d'un " projet " en Delphi Pascal
5. L'interface de Delphi 7
6. Notion dobjet et de programmation objet
7. Les propriétés des objets de Delphi
8. Les propriétés « événements » des objets
9. Modifier les propriétés des objets
10. Applications dans léditeur de textes
11. Exercices

Tu dois devenir capable de

Savoir

1. Citer et commenter les deux types d'informations que peut traiter un ordinateur ;
2. Expliquer la structure d'une application Delphi en montrant le rôle de chacun des fichiers qui la constituent ;
3. Expliquer la nature du travail à réaliser avec Delphi : réalisation d'interfaces utilisateur et du code Pascal.
4. Expliquer la nécessité des langages d'ordinateur, à mi-chemin entre le langage machine et le langage humain ;
5. Expliquer le rôle de l'éditeur de textes, du compilateur et de l'interpréteur dans le cadre de la programmation des ordinateurs ;
6. Situer les différents éléments de l'interface du programme Delphi ;
7. Expliquer la notion d'objet dans le cadre de la programmation et montrer comment des " objets " de la vie courante ressemblent à des " objets " de la programmation ;
8. Illustrer la notion d'" événement " dans le cadre de la programmation par objets ;

Savoir faire

1. Enregistrer un projet Delphi avec la (ou les) fiche(s) qu'il utilise ;
2. Compiler et exécuter un programme Delphi ;
3. Visualiser et modifier les propriétés d'un objet à l'aide de l'inspecteur d'objets de Delphi ;
4. Modifier la valeur d'une propriété lors de l'exécution d'un programme Delphi et en réponse à un événement.

Les deux types d'informations traitées par un ordinateur

Cette partie du cours aura pour but de nous familiariser avec les concepts principaux de la programmation des ordinateurs.

Nous y apprendrons les rudiments de la programmation dans le cadre du langage **Pascal Objet** proposé par **Delphi**.

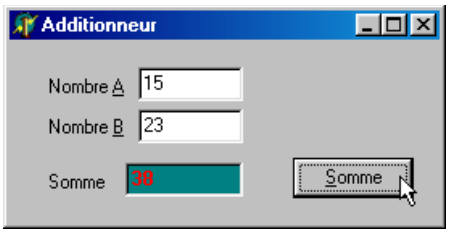
Ce que nous aurons acquis en programmation **Delphi** pourra ensuite être réinvesti dans la programmation en d'autres langages.

Qu'est-ce qu'un programme d'ordinateur?

Un programme d'ordinateur est formé d'un ensemble d'instructions dont l'exécution permet d'atteindre un objectif.

Glups... N'auriez rien en français, dans le même genre?

Disons que l'on pourrait un peu illustrer sur la base d'un exemple. En cliquant sur l'image ci-dessous, tu peux télécharger un petit programme tout simple appelé "L'additionneur". Comme il s'agit d'un exécutable, il faut bien vérifier que l'antivirus de ton ordinateur soit à jour avant de télécharger.



Quand on clique sur le bouton « Somme », l'additionneur calcule le total du Nombre A et du Nombre B.

Pour ce faire, il est muni d'une liste d'instructions du type suivant :

1. recherche la valeur indiquée dans la zone « Nombre A »
2. recherche la valeur indiquée dans la zone « Nombre B »
3. calcule leur somme et
4. indique cette somme dans la zone marquée « Somme ».

Les deux types d'informations traitées par l'ordinateur

Dans le cadre de l'additionneur et des programmes d'ordinateur, en général, celui-ci traite des informations de deux sortes :

1. des informations variables (les différents nombres qui interviennent)
2. des instructions qui lui ont été fournies

En général, les informations variables peuvent être essentiellement du type numérique, de type texte ou de type « valeur logique » (vrai ou faux).

Les instructions doivent lui être fournies dans un langage compréhensible par le processeur. Ce langage est toujours extrêmement simple, et très différent du langage courant.

Toute la difficulté de la programmation des ordinateurs repose donc dans l'art d'établir les instructions des programmes.

Les deux grands types d'informations traitées par un ordinateur sont:
○ Les grandeurs numériques et les textes

○ Les informations variables et les instructions
○ Les informations de type numérique et les informations de type "valeur logique"

Un programme est formé de:
○ Un ensemble d'instructions qui permet d'atteindre un but
○ Un ensemble d'instructions et d'informations variables
○ Un ensemble d'informations variables de type numérique, texte ou logique

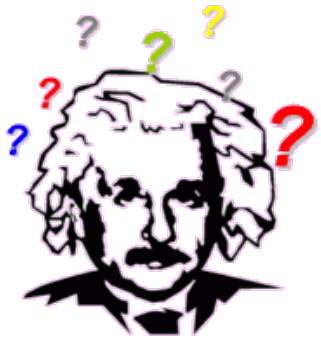
Dans l'additionneur, les informations Nombre A et Nombre B sont:
○ Des instructions
○ Des variables
○ Des valeurs logiques

De l'humain à la machine

On a indiqué, à la page précédente, que la difficulté de la programmation des ordinateurs consiste à leur donner des listes d'instructions qu'ils peuvent comprendre.

Cela pose tout d'abord un problème technique.

Le langage des ordinateurs



Le langage machine

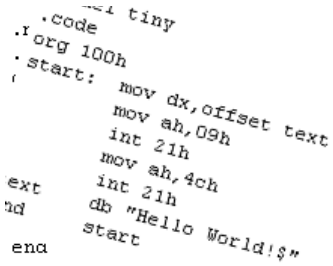
Les seules informations que puisse stocker un ordinateur sont des séries de 0 et de 1. Les données qu'il peut traiter doivent donc être codées sous cette forme.

De même, les instructions destinées aux ordinateurs sont formées d'ensembles de 0 et de 1, compréhensibles par le processeur. Un programme d'ordinateur se présente donc comme sur l'illustration ci-dessous :

```
00110110 11010101 10100011 00111001 1100110001 10101100
00100101 01011010 01110110 11011010 1101101110 10101101
```

Ce que chacun trouvera difficile à comprendre. Un ordinateur, par contre, s'y retrouve très bien.
Les premiers ordinateurs se programmaient dans ce langage : le **langage binaire** encore appelé **langage machine**.

Pour des raisons de facilité, on utilise généralement une variante dans laquelle les instructions sont codées en hexadécimal (notation en base 16).



L'assembleur et les langages évolués

On a ensuite trouvé plus facile de composer les programmes dans des langages plus simples à comprendre pour l'être humain. On charge ensuite un ordinateur de traduire le texte composé par le programmeur vers le langage machine.

C'est ainsi que sont nés le langage « assembleur », encore très proche du code machine, puis les langages plus évolués comme

Fortran, Cobol, Basic, Pascal, C et bien d'autres.

L'avantage de ces langages pour le programmeur est que le texte des programmes est plus facile à comprendre pour un humain.

En contre-partie, il est nécessaire qu'un ordinateur intervienne pour les traduire en langage machine.
Exemples : un programme qui écrit « Salut tout le monde » à l'écran

---

En langage C

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
printf("Salut tout le
monde!\n");
return EXIT_SUCCESS;
}
```

En langage Pascal

```
Program salut ;
Begin
Writeln ('Salut tout le monde') ;
End.
```

D'autres exemples de langages informatique sont illustrés dans l'encyclopédie Wikipedia. Vérifie, pour quelques langages, qu'il s'agit bien d'un moyen terme entre le langage humain et un langage de machine (vois les rubriques Smalltalk, Perl, C++, Cobol,...).

Tu dois maintenant être capable de citer les langages informatiques évoqués ci-dessus et d'expliquer l'intérêt de chacun d'eux. Vérifie que c'est le cas, sans regarder l'écran.

Les langages informatiques "évolués" sont utilisés pour :
○ simplifier le travail des programmeurs
○ simplifier le travail des ordinateurs
○ rendre compréhensible le travail des ordinateurs

Le langage directement compréhensible par les ordinateurs est :
○ le code binaire
○ les langages comme Pascal ou Basic
○ l'assembleur

Les étapes de la traduction en langage machine

L'éditeur de code

Les textes qui constituent les programmes d'ordinateur pourraient être écrits à l'aide de n'importe quel programme de traitement de textes.

Cependant, on préfèrera le plus souvent utiliser des logiciels spécialisés dans la rédaction de ces textes. En plus des fonctions traditionnelles de traitement de textes, on y disposera d'un certain nombre d'outils qui facilitent le travail (autocomplétion du code,...).

L'éditeur de code est donc un programme de traitement de textes un peu spécialisé.

Le compilateur

Les textes composés en assembleur ou dans ces langages évolués doivent être traduits en langage machine. Ce travail est réalisé par un programme spécialisé appelé compilateur.

Le compilateur examine les instructions écrites par le programmeur et les transforment en langage binaire, compréhensible par le processeur.

Il existe un grand nombre de compilateurs. Chacun est spécialisé dans le traitement de l'un ou l'autre langage évolué.

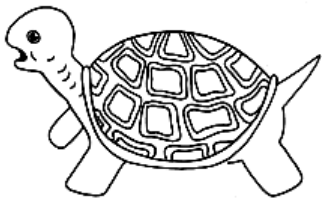
Un texte écrit dans un langage doit être compilé à l'aide d'un compilateur approprié à ce langage précis. Un texte écrit en langage C doit être compilé par un compilateur C.

Le logiciel Delphi inclut, entre-autres, un module de composition de texte en langage Pascal (l'éditeur) et un compilateur Pascal. Ce cours aurait pu également utiliser le langage C (plus difficile à aborder) ou le langage Basic (beaucoup moins structuré), par exemple.

Remarque : les interpréteurs

Dans certains langages, le code source n'est pas préalablement traduit en langage machine par un compilateur.

Dans ce cas, la transformation en langage machine se fait au moment de l'exécution du programme : un interpréteur traduit le programme, ligne par ligne.



Quand une ligne du programme doit être exécutée un grand nombre de fois, l'interpréteur la traduit autant de fois qu'elle est exécutée. Il en résulte une perte de temps et donc une plus grande lenteur des langages interprétés (comme certaines versions de Basic) par rapport aux langages compilés.

Tu dois maintenant être capable de citer les noms des trois logiciels évoqués ci-dessus et d'indiquer leur rôle. Vérifie que c'est le cas, sans regarder l'écran.

Le rôle de l'éditeur de code est de:
○ transformer le langage évolué en code binaire
○écrire le texte des programmes d'ordinateurs
○ traduire le code binaire en langage compréhensible

Détermine l'ordre chronologique dans lequel les trois processus suivants interviennent
○ L'édition du texte, la compilation, l'exécution
○ La compilation, l'édition du texte, l'exécution
○ L'exécution, l'édition du texte, la compilation
○ La compilation, l'interprétation, l'exécution

Le rôle du compilateur est de:
○ transformer le langage évolué en code binaire

○écrire le texte des programmes d'ordinateurs
○ traduire le code binaire en langage compréhensible

La différence entre un interpréteur et un compilateur est que:
○ Le compilateur est spécialisé dans un langage alors qu'un interpréteur peut traiter tous les langages évolués
○ L'interpréteur traduit une ligne de code à la fois, cependant que le compilateur traduit tout le programme en une fois
○ Le compilateur traduit une ligne de code à la fois, cependant que l'interpréteur traduit tout le programme en une fois
○ Il n'y a pas de différence entre un compilateur et un interpréteur

Principe du travail avec Delphi Pascal

Interface utilisateur

Lorsque nous utilisons un ordinateur, le logiciel avec lequel nous travaillons nous propose généralement d'interagir avec lui dans des fenêtres contenant différents éléments classiques : des zones d'édition, des cases à cocher, des boutons à cliquer,...

L'ensemble des éléments cités plus haut figure alors dans ce que l'on appelle une fenêtre, une boîte de dialogue ou plus généralement une **interface utilisateur** :l'endroit où l'humain communique avec la machine, lui fournit des données et reçoit les résultats des traitements par l'ordinateur.

Le travail de l'ordinateur se fait en sous-sol

Lorsque l'ordinateur reçoit l'instruction de traiter les données fournies par l'utilisateur, il exécute une série plus ou moins longue d'instructions afin de produire les résultats souhaités.

Mais j'aimerais bien voir l'ordinateur travailler, moi. Le travail des autres m'a toujours fasciné.

Malheureusement, il est impossible de voir ce travail se réaliser. Tout se passe dans les circuits électroniques du processeur. Quelques mouvements (complexes) d'électrons, tout au plus...

En gros, il se confirme que l'essentiel est invisible pour les yeux !

Quelle culture!

Le travail du concepteur du logiciel

Lorsque l'on conçoit un logiciel, il y a donc au moins deux tâches à réaliser :

1. concevoir une interface utilisateur claire et pratique

2. imaginer les instructions nécessaires à l'ordinateur pour produire les résultats espérés.

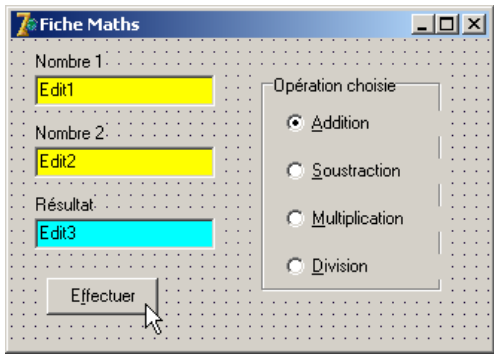
Lorsque nous travaillerons à concevoir des logiciels avec Delphi Pascal, nous aurons donc deux types d'activités :

1. concevoir des **interfaces utilisateurs** encore appelées des **fiches** dans le vocabulaire de Delphi
2. **imaginer les instructions** qui constitueront les programmes.

La conception des fiches

L'interface utilisateur contient un certain nombre de " contrôles " (cases à cocher, zones d'édition, listes déroulantes,...) qui permettent :

1. à l'utilisateur de fournir des informations à l'ordinateur (les entrées) et
2. à l'ordinateur de fournir les résultats de ses traitements (les sorties).



Lors de la conception des fiches, nous allons donc disposer les " contrôles " nécessaires sur les fiches.

Dans le vocabulaire de Delphi, les " contrôles " sont appelés des **composants**.

Dans l'exemple ci-dessus, une fiche en cours d'élaboration contient des **composants** de différents types : zones d'édition, boutons d'options, étiquettes, bouton d'action,...

La rédaction des programmes

A un signal donné (un clic sur un bouton, la frappe d'une touche au clavier,...), l'ordinateur devra exécuter une série plus ou moins complexe d'instructions.

Ces instructions seront décrites en langage Pascal dans l'éditeur de code de Delphi.

La fenêtre dans laquelle l'utilisateur entre en contact avec l'ordinateur est appelée
○ Interface utilisateur
○ Interface ordinateur
○ Compilateur
○ Fiche

Dans le vocabulaire de Delphi Pascal, une fiche est
○ Une interface utilisateur en construction
○ Un " contrôle " à placer dans une interface utilisateur
○ Un programme d'ordinateur

Dans le vocabulaire de Delphi Pascal, un composant est
○ Un programme d'ordinateur
○ Un "contrôle" à placer sur une fiche
○ Une interface utilisateur

Structure d'un " projet " en Delphi Pascal

Interfaces utilisateur et application Delphi Pascal

En général, dès qu'un logiciel devient un peu complexe, il présente plusieurs interfaces utilisateur. Ainsi, un logiciel de traitement de textes offrira des interfaces utilisateur pour :

- mettre en forme les paragraphes (retraits, espacements, tabulations,...)
- mettre en forme les caractères (couleur, police de caractère, taille,...)
- imprimer les documents (choix des pages à imprimer, orientation des pages, nombre d'exemplaires,...)
- enregistrer les documents (nom de fichier, format,...)
- ...

Un logiciel complexe réalisé avec Delphi Pascal pourrait donc demander de concevoir plusieurs **fiches**.

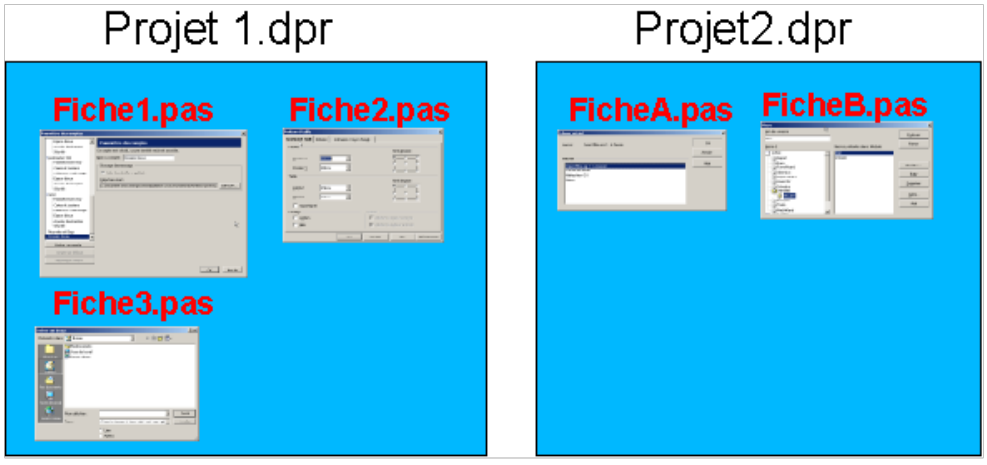
Comment enregistrer son travail

Une application réalisée avec Delphi Pascal est encore appelée un **projet**. Chaque **projet** peut comporter un nombre indéfini de fiches, selon les besoins.

Lorsque l'on veut enregistrer son travail, il faut donc sauvegarder :

- le **projet**, dans un fichier qui porte l'extension **.dpr** (qui rappelle qu'il s'agit d'un **pr**ojet en **D**elphi ;
- chaque fiche et les instructions liées aux composants qu'elle contient dans un fichier **.pas** (qui rappelle qu'il s'agit de langage **Pascal**).

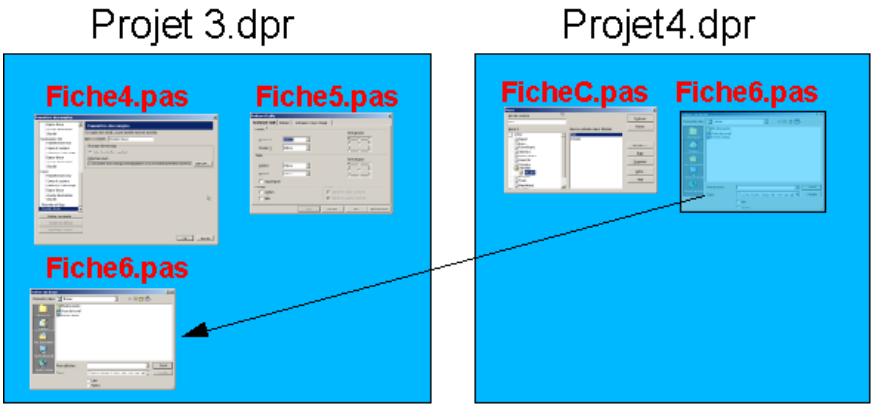
L'illustration ci-dessous présente deux projets. Le premier contient 3 fiches ; le deuxième en contient seulement deux.



Dites, ils ne se sont pas trop foulés, les concepteurs de Delphi. Ils n'auraient pas pu penser à enregistrer toutes les fiches ensemble dans le projet ?
Si je compte bien, il faut 4 enregistrements différents rien que pour l'application **projet 1.dpr**.

Tout à fait exact : quatre enregistrements sont nécessaires : le projet et chacune des trois fiches.
Mais nous allons voir l'avantage que présente cette méthode maintenant.

Dans l'exemple suivant, le programmeur du projet 4 s'est rendu compte qu'il a déjà élaboré une fiche qui répond à ses besoins dans le projet 3.



Plutôt que de recommencer tout le travail, il pourra indiquer qu'il souhaite simplement reprendre la fiche 6 de l'autre projet.

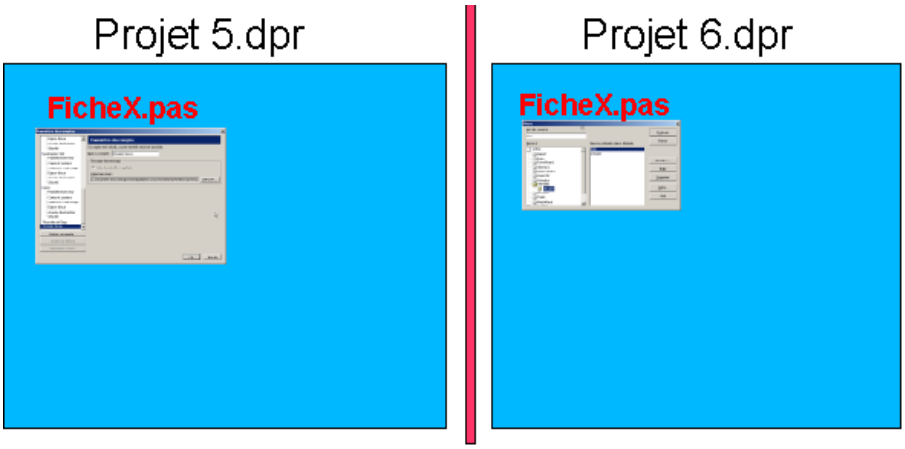
Super, cette méthode de recyclage. Elle convient tout à fait à mon naturel paresseux. J'espère que je pourrai souvent l'utiliser.

Pas de chance, dans le cadre de ce cours, les applications compteront rarement plus d'une seule fiche.

Attention, danger

Le " recyclage " des fiches conçues précédemment est évidemment un gros avantage lorsque l'on conçoit de nombreux projets de grande ampleur. Mais il présente un piège, si l'on n'y prend pas garde.

Dans l'exemple ci-dessous, deux projets différents contiennent chacun une seule fiche nommée **FicheX.pas**.



Ces deux projets ne peuvent pas être enregistrés dans le même répertoire.
Pour quelle raison ?
○ Parce que deux projets, c'est vraiment trop gros pour un seul répertoire.
○ Deux fichiers situés dans le même répertoire porteraient le même nom.
○ Parce que le nombre de fichiers possibles dans un répertoire est trop limité.

Combien de fiches peut compter un projet Delphi, au maximum?
○ Aucune
○ Une seule
○ Maximum 4
○ Il n'y a pas de limite

Derniers conseils

Avant de démarrer, essaie de bien comprendre l'intérêt des différents conseils qui suivent. N'oublie pas d'en profiter durant ton travail.

- Chaque exercice proposé dans le cours fait l'objet d'une nouvelle application.
- Chaque application est enregistrée dans un fichier **.dpr** qui porte un nom évocateur (et non **projetx.dpr** comme le propose Delphi).
- Chaque fiche est enregistrée dans un fichier **.pas** qui porte un nom évocateur (en nom **UnitX.pas**).
- Chaque projet est enregistré dans un répertoire qui permet de retrouver facilement tel exercice présenté à telle page du cours.

Et donc, l'exercice 3 de la page 8 de la leçon 7, je l'enregistre dans un répertoire appelé Chap7p8ex3, par exemple ?

Voilà un nom qui n'est guère joli, mais qui a le mérite d'être efficace !

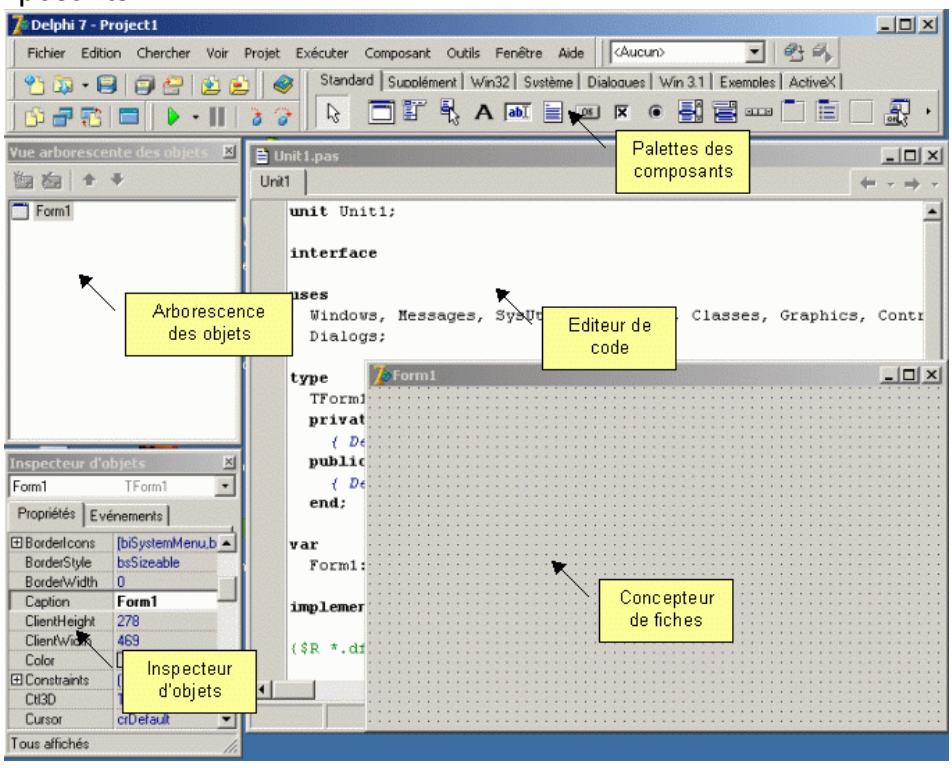
Utiliser l'interface de Delphi et exécuter un programme

Avant d'utiliser le logiciel **Delphi**, voyons comment se présente l'écran principal et tentons d'indiquer les différents éléments qui le composent.

Structure de l'interface

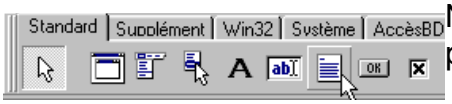
Au lancement de **Delphi**, on est confronté à une interface du type illustré ci-dessous. On y distingue :

1. la palette des composants
2. le concepteur de fiches
3. l'éditeur de code (généralement caché derrière la fenêtre du concepteur de fiches)
4. l'explorateur des objets
5. l'inspecteur d'objets



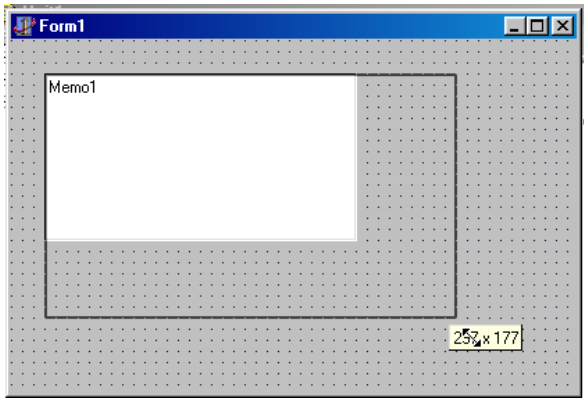
- Si ce n'est déjà fait, démarre le logiciel **Delphi**.
- Inspecte les différents éléments de l'interface.
- Déplace la fenêtre du concepteur de fiches pour apercevoir la fenêtre de l'éditeur de code.

Utilisation de l'interface utilisateur de Delphi



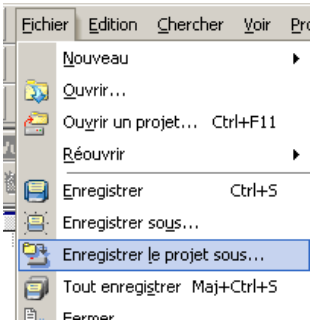
Nous allons, en quelques instants, concevoir un micro-programme de traitement de textes.

- Dans la palette des composants, sélectionne, si nécessaire l'onglet **Standard**.
- Clique sur le composant **Memo**
- Clique ensuite n'importe où dans le concepteur de fiche : un composant **Memo** y est déposé.
- Redimensionne le **Memo** pour qu'il occupe la plus grand partie de la fiche.



Nous avons déjà fini d'utiliser l'éditeur de Delphi : la partie du logiciel qui sert à concevoir les programmes.

Enregistrer le projet en cours



Il est temps, maintenant, d'enregistrer le travail dans ton répertoire personnel. L'enregistrement va se faire en deux temps.

Dans un premier temps, tu vas enregistrer la fiche contenant le composant **Memo** que tu viens de composer.

Dans un deuxième temps, tu vas enregistrer l'ensemble du projet de traitement de textes.

- Prépare l'enregistrement en créant un nouveau répertoire nommé **ch1p5** dans ton répertoire personnel.
- Dans le menu **Fichier**, sélectionne la commande **Enregistrer le projet sous...**
- Dans la première boîte de dialogue d'enregistrement, sélectionne ton répertoire personnel puis dans **ch1p5**.
- Remplace le nom par défaut **Unit1.pas** par le nom **tdt.pas**
- Clique sur **Enregistrer** pour... enregistrer la fiche
- Dans la deuxième boîte de dialogue d'enregistrement, remplace le nom par défaut **Projectx.dpr** par le nom **tdtxt.dpr**.

Le nom du projet doit être différent du nom des unités qu'il contient.

Il est très important d'enregistrer tout projet avant l'étape suivante qui est la compilation. Lors de cette dernière étape, le compilateur va tenter d'écrire le programme exécutable sur le disque dans le répertoire par défaut qui, pour des raisons de sécurité, se trouve... dans un répertoire protégé contre l'écriture.

S'il ne peut pas écrire le programme, le compilateur va signaler l'erreur et interrompre le processus de traduction.

Nous allons finalement demander au compilateur d'entrer en jeu et de transformer le code **Pascal** qui a été écrit automatiquement en **langage machine**.

Compiler le projet

- Dans le menu Projet, sélectionne la commande Compiler tdtxt.

Après quelques instants de travail, le logiciel nous redonne la main. La compilation a eu lieu : le code machine se trouve

- dans la mémoire de l'ordinateur ;
- dans ton répertoire personnel, sous la forme d'un fichier exécutable (.exe).

Notre programme peut maintenant être exécuté.

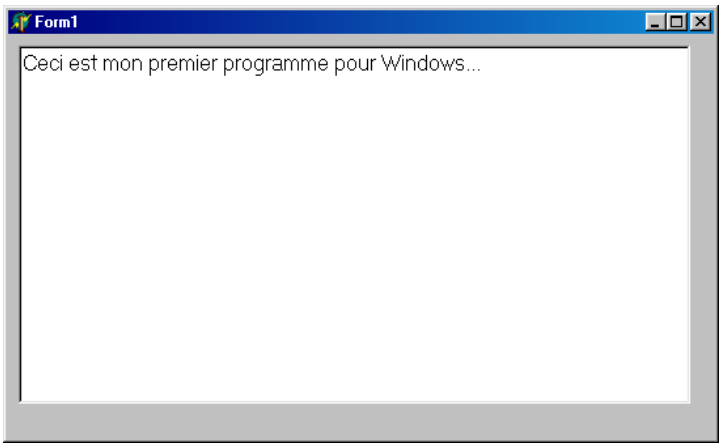
Vous êtes sûr' Je ne l'ai rien vu faire, moi !

Tout à fait sûr. Simplement, le processus est très rapide et silencieux... tant que tout se passe bien.
Nous allons d'ailleurs pouvoir exécuter le programme.

Exécuter le programme

- Dans le menu Exécuter , sélectionne la commande Exécuter ou clique sur l'outil  ou frappe la touche F9.

Le programme s'exécute. Nous pouvons écrire comme bon nous semble dans la fenêtre de notre micro-traitement de textes.



- Pour interrompre, clique sur  ou ALT + F4

Notion d'objet et de programmation objet

Notion d'objet

Dans le cadre de la programmation par objets, nous allons travailler avec des entités du même type que la **Fiche** ou le **Memo** utilisés dans le micro-programme de traitement de textes précédent.

Ces objets sont munis de deux types de caractéristiques principales : ils disposent

1. D'un certain nombre de **propriétés** et
2. D'un certain nombre de **méthodes**, des actions que peut réaliser l'objet

Analogie

Une automobile peut être comparée à un objet de la programmation. Effectivement, elle dispose d'un certain nombre de

1. **Propriétés** : sa couleur, sa cylindrée, sa puissance moteur, le nombre de places, le nombre d'airbags,...
2. **Méthodes** : démarrer le moteur, freiner, ouvrir une portière, dégivrer la lunette arrière, klaxonner,...

Exercices

1. Montre qu'un lecteur de cassettes vidéo est comparable à un objet de la programmation.

2. Même question pour un radio-réveil.

Les propriétés des objets de Delphi

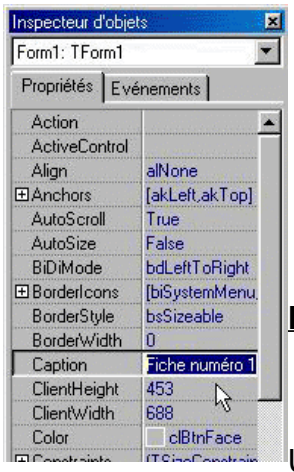
Lors de la programmation en Delphi, nous allons manipuler des objets qui disposent également

- de **propriétés** et
- de **méthodes**

Commençons par évoquer les propriétés.

Propriétés de l'objet Fiche dans l'inspecteur d'objets

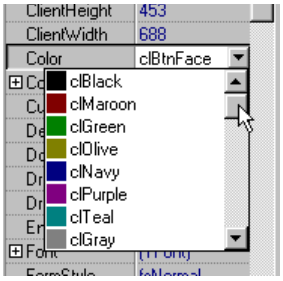
La propriété Caption de la fiche



- Si nécessaire, démarre Delphi ou commence une nouvelle application: **Fichier** > **Nouvelle application**
- Dans l'inspecteur d'objets, repère le nom de l'objet en cours : **Form1** et recherche la propriété **Caption** ;
- Modifie le contenu de la propriété **Caption** en y écrivant **Fiche numéro 1**
- Observe, pendant la frappe, le titre de la fiche.

La propriété Color de la fiche

- Repère la propriété **Color** et clique dans la zone indiquant la couleur



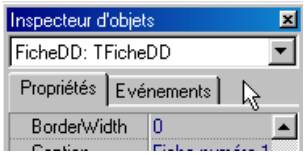
Une liste déroulante des différentes couleurs disponibles apparaît

- Sélectionne la couleur bleue référencée par la notation **clBlue** (signifiant, bien sûr, couleur bleue)

La couleur de la fiche se modifie instantanément.

La propriété Name de la fiche

Chaque objet porte un nom qui permet de l'identifier durant le déroulement du programme qui le contient.



- Recherche la propriété **Name** de la fiche et indiques-y la valeur **Fiche** suivie de tes initiales (sans espace entre **Fiche** et tes initiales) ; par exemple, Dominique Dupont indiquera **FicheDD**.

Aucun effet visuel ne se manifeste ici : la propriété est pourtant bien fixée, mais tout se passe au niveau de la mémoire de l'ordinateur ;

- Examine maintenant l'éditeur de code (déplace éventuellement la fiche) : le logiciel a tenu compte du nouveau nom ;
- Examine le haut de la fenêtre de l'inspecteur d'objets et vérifie que le nouveau nom est également pris en compte.

Autres propriétés à examiner

D'autres propriétés intéressantes de la fiche peuvent encore être examinées et testées. Passe quelques minutes à expérimenter sur les propriétés suivantes :

- Left
- Top
- Width
- Height
- Cursor

Conclusion

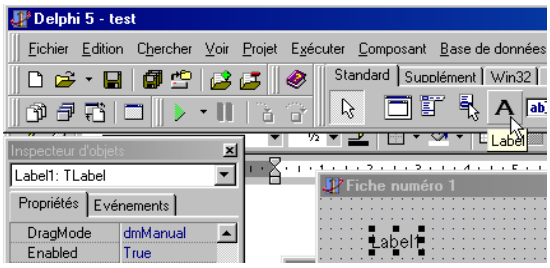
Delphi est un outil de programmation visuelle. Dans ce concept, la mise au point des programmes se fait dans une interface ordinateur/programmeur qui facilite, au maximum, la visualisation des effets souhaités à l'écran dès la conception.

En programmation non visuelle, il faut préciser manuellement chacune des propriétés de l'objet.
Certaines propriétés ne peuvent cependant être répercutées visuellement (Ex : le nom de la fiche) ou ne sont répercutées qu'à l'exécution (Ex : la forme du curseur).

Delphi fait partie de la même famille de logiciels que Visual Basic ou C++ Builder, par exemple.

Applications

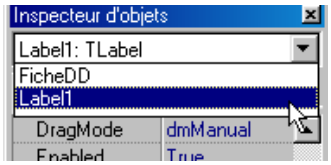
Propriétés de l'objet Label dans l'inspecteur d'objets



- Dans la palette standard, sélectionne l'objet **Label** à l'aide d'un simple clic ;
- Déplace la souris vers la fiche ;
- Fais un nouveau clic à l'endroit où tu

souhaites déposer l'objet **Label**.

- Vérifie la présence de deux objets, dans cette application, par l'intermédiaire de l'inspecteur d'objets : la fiche et le **Label** ;
- Pour activer la **Fiche** dans l'inspecteur d'objets, clique n'importe où dans la fiche ;
- Pour activer le **Label** dans l'inspecteur d'objets, clique sur le **Label** sur la fiche.

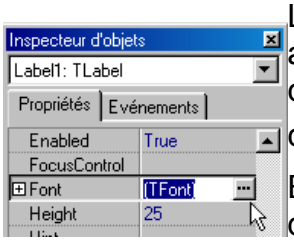


Tu peux aussi sélectionner les objets dans la fenêtre de l'arborescence des objets.

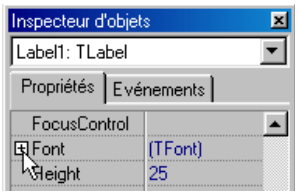
L'objet **Label** dispose d'un autre jeu de propriétés que la **Fiche**, bien que certaines propriétés soient communes.

Passe quelques minutes à expérimenter sur les propriétés suivantes :

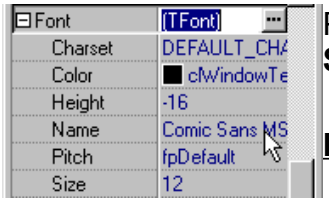
- Caption
- Color
- Width
- Height
- Cursor
- Name



La propriété **Font** est constituée de plusieurs sous-items. On peut les atteindre en cliquant dans la zone de définition de la propriété puis en cliquant sur le bouton ... qui y apparaît. La boîte de dialogue classique de police de caractères apparaît alors.

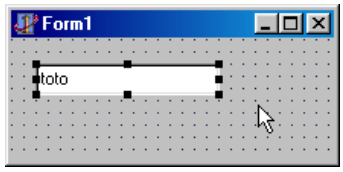


En cliquant sur le signe **+** à gauche du nom de la propriété, on ouvre plusieurs lignes supplémentaires dans l'inspecteur d'objet. Chacune des lignes permet de régler certains aspects de la propriété **Font**.



Parmi les items qui apparaissent, la ligne **Style** peut également être ouverte de la même façon.

Propriétés de l'objet TEdit dans l'inspecteur d'objets



- Sur une fiche vierge, dépose un composant **TEdit**.
- Dans l'inspecteur d'objets, modifie sa propriété **Text** de manière à ce que la mention **Edit1** soit remplacée par ton prénom
- Modifie également la fonte de la police de caractères et sa couleur selon ton goût.

Autres objets

- Etablis une fiche semblable à celle de l'exemple ci-contre .
- Modifie les propriétés des différents objets présentés de manière à obtenir la plus grande ressemblance

possible. Le pointeur de souris a la forme de sablier uniquement lorsqu'il passe sur le composant **Memo**.
- Examine quelques autres propriétés de ces objets.

Les propriétés « événements » des objets

Dans la programmation pour Windows, l'ensemble des logiciels en fonctionnement est géré par des événements : telle touche a été frappée au clavier, un clic du bouton droit de la souris a eu lieu, tel élément a été activé, on a fait glisser la souris,...

Delphi gère les événements comme des propriétés des objets.

Les objets de **Delphi** sont également capables de répondre aux événements.

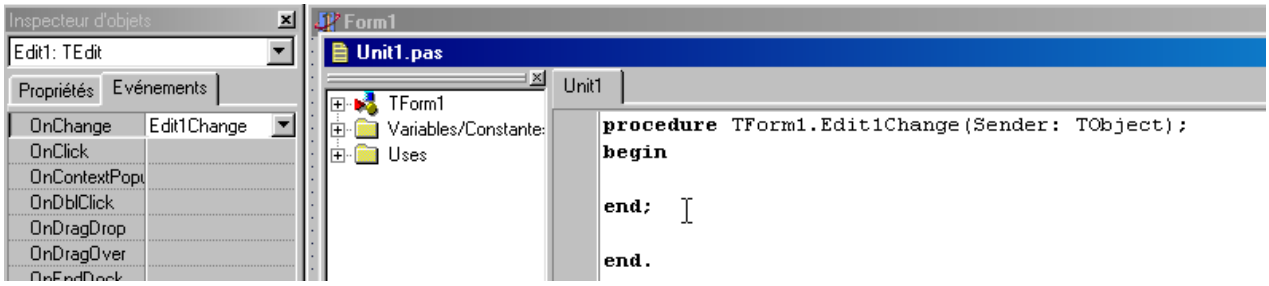
Pour chaque objet, on trouve dans l'inspecteur d'objets l'ensemble des événements auxquels il est capable de répondre.

L'illustration ci-contre présente l'inspecteur d'objets d'un composant de type **Tedit**. Parmi les événements auxquels il est sensible, on voit :

- **OnChange** : quand le contenu du composant change ;
- **OnClick** : quand on y fait un clic de souris ;
- **OnDblClick** : quand on y fait un double clic ;
- **OnEnter** : quand on entre dans le composant (début d'édition)
- **OnKeyDown** : quand on frappe une touche dans le composant ;
- ...

Pour indiquer une action à exécuter quand tel événement a eu lieu, double clic dans la zone correspondant à l'événement.

Automatiquement, l'éditeur de code s'ouvre au bon endroit : il suffira ensuite de compléter le texte de la procédure correspondante.



Nous apprendrons à utiliser l'éditeur de code sur une page ultérieure.

Pour ouvrir l'éditeur de code à partir de la fenêtre de l'inspecteur d'objet, il faut
○ Faire un simple clic dans une propriété de l'objet visé
○Faire un simple clic dans un événement de l'objet visé
○ Faire un double clic dans un événement de l'objet visé
○ Frapper la touche F9

Dites, j'ai malgré tout un petit soucis, parce que cela fait un peu " charabia ", toutes ces expressions comme

" onDragDrop " et autres.

Il est vrai que la connaissance d'un peu d'anglais n'est pas nuisible, ici. Cependant, pour avoir toutes informations utiles, il suffit de faire un simple clic dans la zone correspondant à l'événement et de frapper la touche F1 pour déclencher l'aide intégrée du logiciel.

Ouah ! C'est magique !!

Modifier les propriétés des objets

La qualification des objets

Jusqu'à présent, nous avons modifié les propriétés des objets en changeant leurs valeurs dans l'inspecteur d'objet.

Cela peut aussi se faire dans le déroulement d'un programme.

Pour indiquer un objet et une de ses propriétés, on fait suivre le nom de l'objet de la propriété que l'on souhaite désigner ; on sépare le nom de l'objet de sa propriété par un « . ».

On écrira, par exemple :

OBJET	.	PROPRIETE
Label1	.	Color
Fiche1	.	Top

L'affectation d'une valeur à une propriété

En programmation, l'affectation est l'action de donner une valeur à une propriété ou à une variable.

Le symbole qui sert à désigner l'affectation en langage Pascal est « := ». Ce symbole varie en fonction du langage dans lequel on travaille : en langage C et en Basic, le symbole est « = ».

Donc, pour changer la valeur d'une propriété sous Delphi, on écrira, par exemple :

`Label1.Color := clred`	Ce qui confèrera la couleur rouge au composant appelé Label1.
`Fiche1.Top := 120`	Ce qui déplacera le coin supérieur gauche de la fiche à la coordonnée 120 depuis le haut de l'écran.

Applications dans l'éditeur de code

Nous avons vu, à la page précédente, que la syntaxe pour l'affectation d'une valeur à une propriété est du type **objet.propriété := valeur**.

Voyons ce que cela donne en pratique.

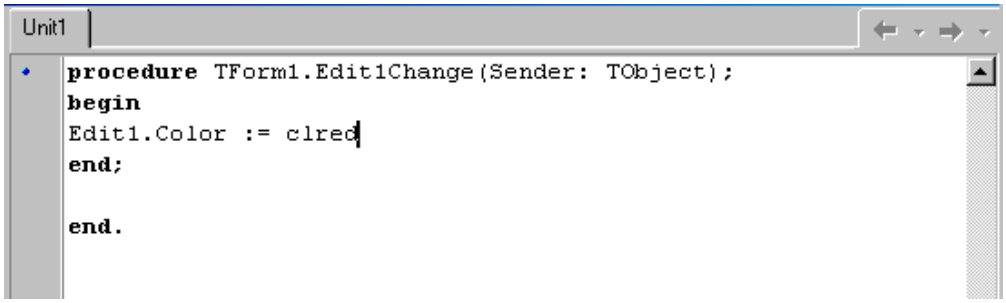
Changement de couleur d'un composant de type Tedit

- Sur une fiche, dépose un composant de type **Tedit** dont tu conserveras le nom

Edit1 ;
- Si nécessaire, active l'objet **Tedit** ;
- Active l'onglet **Evénements** de l'inspecteur d'objets ;
- Fais un double clic dans l'événement **OnChange** ;

L'éditeur de texte s'ouvre, le curseur se trouve dans la procédure **Tform1.Edit1Change**, entre les mots **begin** et **end**.

- A cet endroit, frappe l'instruction **Edit1.Color:=clred**, comme sur l'illustration ci-dessous.



- Compile et exécute ce programme.
- Tente de modifier la valeur indiquée dans le composant **TEdit** : il vire immédiatement au rouge.
- Termine le programme en cliquant sur la case de fermeture .
- Recommence en choisissant une autre couleur. La liste des couleurs disponibles est indiquée dans la propriété **Color** de l'objet **Tedit** dans l'inspecteur d'objets.

Changement de position d'un composant TButton

- Sur une fiche, dépose un composant de type **Tbutton** dont tu conserveras le nom et que tu placeras à la position 20 pixels du haut de l'écran ;
- Si nécessaire, active l'objet **TButton** ;
- Active l'onglet **Evénements** de l'inspecteur d'objets ;
- Fais un double clic dans l'événement **OnClick** ; l'éditeur de texte s'ouvre, le curseur se trouve dans la procédure **TForm1.Button1Click**, entre les mots **begin** et **end** ;
- A cet endroit, frappe l'instruction **Button1.Top := 50** ;
- Compile et exécute ce programme.

Lors d'un clic de souris sur le bouton, celui-ci se déplace à la position 50 pixels depuis le haut de la fiche.

- Fais de même avec une autre grandeur numérique du composant **TButton**.

Modification d'une valeur de type texte

- Sur une fiche, dépose un composant de type **Tbutton** dont tu conserveras le nom et la valeur de **Caption** ;
- Dans l'événement **OnClick**, indique l'instruction qui permettra de changer la valeur de **Caption** : **Button1.Caption := 'Cliqué'** ;

Les informations de type **Texte** doivent être indiquées entre des apostrophes en langage Pascal.

- Compile et exécute ce programme.

Exercices

1. Etablis une fiche dans laquelle un composant **TLabel** prend la couleur rouge si l'on clique sur un bouton et la couleur bleue si l'on clique sur un autre bouton. Enregistre le projet dans le sous-répertoire **RougeBleu** dans ton répertoire personnel.
2. Etablis une fiche dans laquelle tu placeras un composant **TEdit**

- quand le curseur entre dans le composant **TEdit**, ce dernier prend la couleur rouge ;
- quand le curseur sort de ce composant, il retrouve une couleur bleue.

Conseil : examine attentivement les événements qui pourraient être utiles. Enregistre le projet dans le sous-répertoire **RougeBleuBis** dans ton répertoire personnel.

3. Etablis une fiche dans laquelle tu placeras un **TButton** et deux zones **TEdit**. Quand on clique sur le bouton, le contenu de la première zone **TEdit** se recopie dans la deuxième zone. Enregistre le projet dans le sous-répertoire **CopieEdit** dans ton répertoire personnel.
4. Etablis une fiche dans laquelle tu placeras deux composants de type **TEdit** ; le contenu du deuxième composant doit être automatiquement identique à celui du premier composant, dès que celui-ci change. Enregistre le projet dans le sous-répertoire **CloneEdit** dans ton répertoire personnel.
5. Etablis une fiche dans laquelle tu places un composant de type **TButton** à 100 pixels du bord gauche et un **TLabel** à proximité; quand la souris se déplace au-dessus du bouton, celui-ci se déplace à la position 200 pixels du bord gauche. Quand elle se déplace au-dessus du **TLabel**, le bouton reprend sa place. Enregistre le projet dans le sous-répertoire **BoutonFarceur** dans ton répertoire personnel.