Initiation Delphi – 1/2

* Initiation à Delphi :

Il y a peu de temps, un magazine distribuait la version 1 de Delphi à prix réduit. L'idée était séduisante, car pour une somme modique, on pouvait disposer d'un logiciel de création performant et s'initier à la création de programmes. Malheureusement, la documentation était succincte. Il est vrai que la documentation complète sous acrobat reader était disponible, mais avait de quoi rebuter la plupart des vrais débutants qui veulent s'essayer à la programmation "pour voir". Le but de cet article est de familiariser tout un chacun avec ces notions basiques et vous aider à concevoir vos premiers programmes.

Le langage Delphi est l'un des langages de programmation parmi les plus connus. Il est l'héritier du fameux "turbo-pascal" de Borland (qui s'appelle aujourd'hui Inprise). En fait, il s'agit d'une version "orientée objet" de ce langage. Ce système de développement est souple, polyvalent, et permet au débutant, comme à l'amateur éclairé de développer rapidement des applications (ce que l'on appelle communément des "programmes"). La procédure d'installation étant parfaitement décrite dans la prise en mains du magazine, nous entrerons directement dans le vif du sujet.

I) La première fois

Lorsque vous avez démarré Delphi, votre écran s'est configuré de la manière suivante :

1) En haut de l'écran, un bandeau bleu, avec au-dessous des intitulés, et en dessous, des icônes. C'est là sur une bande précise à gauche que sont placées les commandes dont nous aurons besoin, les icônes figurant au-dessous étant une simple facilité : il est en effet plus facile de cliquer sur un bouton, que d'ouvrir un menu déroulant et de sélectionner une option. De plus l'aspect visuel des icônes permet une utilisation intuitive. Sur la bande de droite, figurent les "objets" qui nous serviront à établir notre programme, et que nous étudierons plus loin.

2) Sur le côté gauche, un panneau qui présente sur le bas 2 onglets : propriétés et événements : c'est "l'éditeur d'objet". Comme son nom l'indique, ce panneau nous servira à définir l'aspect des éléments (composants) que nous utiliserons dans notre programme, et les interactions les liant.

3) Plutôt sur le coté droit, et tenant la plus grande place, un panneau gris marqué "Form1". C'est là , sur cette feuille virtuelle de papier, que nous créerons notre application, en visualisant directement le résultat de notre travail. L'essentiel de notre travail consistera à "coller" dessus des "composants", comme on le fait avec des post-it.

Il est maintenant nécessaire de vous expliquer ce que signifie programmation "orientée objet". Aux temps héroïques, le programmeur décrivait dans un langage précis et suivant des règles de syntaxe précises, un enchaînement de traitements. Pour que son programme soit esthétiquement réussi, et donne un aspect "professionnel", il passait presque autant de temps à peaufiner la présentation qu'a écrire ses routines de calcul. Ne souriez pas, car, à cette époque, tout le monde bidouillant dans son coin des programmes, il fallait se sortir du lot. Il est évident qu'un programme esthétique ne fonctionnera pas mieux qu'un programme esthétiquement raté, mais il fera plus fini, plus professionnel, et surtout plus convivial. Ce travail était fastidieux et répétitif. C'est à partir de ce constat que furent développés les "objets".

II) Qu'est-ce qu'un objet ?

Si l'on se réfère à sa définition, un objet a un aspect précis et durable, et accomplit une fonction précise. Une paire de ciseaux a toujours le même aspect qu'elle soit en plastique rouge ou en acier inoxydable, de grande taille ou de petite taille. Son aspect ne se modifie pas (sauf si on la passe dans un four électromagnétique à induction à plus de 600°, mais ce n'est pas l'objet de cet article), et elle sert à couper. En informatique, nous avons

souvent à afficher du texte. Nous utiliserons pour ce faire, un objet "étiquette" (ou "label" en shakespearien moyen). Nous choisirons sa couleur, la couleur du fond, nous choisirons également ses dimensions qui seront durables une fois fixées. Ensuite, elle sert à afficher le texte choisi par le programmeur. Nous avons donc un "outil configurable" dans notre trousse à outils : c'est un objet. Nous les appellerons "composants" par la suite.

Les composants sont regroupés sous la barre de menus, à droite et différents onglets sont accessibles. Positionnons-nous sur l'icône "label". Il y a une aide succincte qui vous renseignera sur le nom du bouton si vous laissez le curseur souris immobile dessus pendant quelques secondes. Cliquez ou double-cliquez dessus. Si vous cliquez, il ne se passera rien : il vous faudra cliquer n'importe où sur la partie grise de la feuille (intitulée "form1") pour voir votre composant apparaître. Par contre, si vous double-cliquez, il se positionnera de lui-même, au centre de la feuille.

III) En savons-nous assez pour créer notre première application ?

Oui, avec la P.O.O. (Programmation Orientée Objet) nous sommes prêts pour tenter de concevoir nous-mêmes notre premier programme. Notre première application, à défaut d'être extraordinaire, aura le mérite de nous familiariser avec quelques mécanismes très simples qu'il nous faut connaître, avant de nous lancer dans la grande aventure logicielle.

Notre premier programme visera tout simplement à transformer une température exprimée en degrés Celsius, en degrés Farenheit.(Pour cela, nous pouvons nous dispenser de l'étape de préparation du programme, car ce n'est pas l'objet de cet article, mais pour un vrai programmeur, l'étape de la programmation est la dernière étape de l'analyse qu'il a faite du problème, et de sa solution.)

Notre seul algorithme de programmation (méthode de résolution du problème) tient dans la formule suivante :

DegréFarenheit = ((DegréCelsius *9)/5)+32

Dans cet exemple, DegréFarenheit et DegréCelsius sont des variables, comme le sont x et y en mathématiques. La programmation consiste à apprendre au programme ce qu'il faut faire avec les variables (lui dire quel est l'enchaînement des calculs à réaliser, comme dans une formule mathématique ex : pour calculer une droite : y=2*x+1...). Toutefois, pour que le programme ait une utilité, il faut faire ce que l'on appelle l'application numérique (oui, je sais, c'est loin tout ça...) c'est à dire effectuer le calcul avec les chiffres réels. Pour cela, il faut affecter la valeur de la température à convertir à la variable DegréCelsius, ce qui nous donnera grâce à la formule écrite, un résultat numérique (une valeur) pour DegréFarenheit.

Nous allons donc créer :

1 label pour indiquer "saisie de la température en d¡ Celsius".
1 label pour indiquer "valeur convertie en d¡ Farenheit".
1 editbox pour saisir la valeur en degrés Celsius.
1 editbox pour afficher la valeur convertie en degrés Farenheit.
1 bouton "quitter" pour quitter (quelle surprise...) l'application.

Sélectionnez chacun des ces éléments et positionnez les sur la feuille. Leur emplacement n'a aucune importance, mais essayez de faire en sorte que le résultat ne soit pas incohérent (par exemple en distribuant les composants aux quatre coins de la feuille). Nous verrons ultérieurement comment Delphi peut se charger d'un positionnement harmonieux. Vous remarquerez que Delphi nomme chaque composant que vous allez placer sur la feuille (editbox1, editbox2, button1,...). Sachez que vous pouvez les renommer pour plus de clarté ("saisie_d°" étant plus parlant qu' "editbox1"), mais nous ne le feront pas ici. Les noms de chaque composant apparaissent en haut de l'éditeur d'objet, (le petit triangle vers le bas vous permet de sélectionner un composant et d'afficher ses propriétés).

IV) Click et double-click sont dans un bateau...

Si vous cliquez (ne double-cliquez pas tout de suite, ou alors lisez le paragraphe qui concerne le double-click) sur l'un des objets dans la feuille, vous verrez ses contours apparaître en noir, avec des carrés aux angles et au milieu de chaque segment : ce sont les poignées de dimensionnement, qui permettent de modifier la taille du composant.

Mais plus utile, l'éditeur d'objet donne tous les paramètres dudit composant. Vous avez ainsi accès à tous les paramètres. Vous pouvez ainsi vous amuser, en temps réel, à faire toutes les modifications possibles et imaginables. Changez la couleur du fond de la boite, changez la couleur du texte, tout vous est permis. Toutefois, il convient de noter que le nom de la boite figure dans la propriété caption. Le texte affiché dans la boite, figure dans la propriété texte.

Vous noterez que certains paramètres sont suivis de 3 points dans une boite grise à l'extrême droite de la ligne de paramètre; ce sont ceux qui déclenchent une autre boite de dialogue, où l'on vous demandera de préciser vos actions. Dans la plupart des cas, le fait de double-cliquer dans la colonne de droitevous permettra d'avoir un menu plus convivial. Loin de vous faire une liste exhaustive, je pense que le mieux, c'est de manipuler : avec l'habitude, vous connaîtrez les possibilités de chaque objet, et vous n'y réfléchirez même plus.

Il est à noter que certains paramètres dans la colonne de gauche, sont précédés du signe + . Dans ce cas, il s'agit d'un menu "gigogne" : si vous cliquez la case comportant le +, il se transforme en -, et vous affiche une nouvelle série de sous-paramètres de l'option considérée. Un autre click sur le - et vous revoilà avec un +, mais ces sous-menu ont disparu...

Vous vous êtes bien amusé et maintenant vous avez une boite rose avec le texte "saisie de la température en d° Celsius" en vert, et une boite mauve avec le texte "valeur convertie en d° Farenheit". En face 2 boites désespérément vides. C'est par l'une de ces boite que nous allons saisir la valeur de la température à convertir, et afficher, par l'autre, le résultat de la conversion. Mais laissez-moi vous dire que du point de vue des couleurs, vous avez vraiment de drôles de goûts.

V) Il reste double-click

Si vous avez double-cliqué sur le composant, vous avez vu une page apparaître, entièrement écrite. C'est en fait le véritable programme que vous êtes en train d'écrire. Ce que l'on appelait "le listing" (nostalgie, quand tu nous tiens...) à l'époque héroïque de la programmation. Il faut bien reconnaître qu'il est nettement plus facile d'écrire le programme de la façon dont nous l'avons fait, que de devoir saisir tout ce texte à la main. Si vous parcourez ce texte par les ascenseurs, vous pourrez identifier 4 zones. Si nous remplissons correctement ces 4 zones (d'ailleurs, Delphi va nous y aider), en y indiquant les interactions entre les composants (les "événements"), nous aurons fini notre premier programme. Voilà, ce n'est pas plus compliqué que cela. Vous avez certainement remarqué que le bas de l'éditeur d'objet contenait 2 onglets : le premier, "propriétés", vous le connaissez, car nous venons de l'utiliser, le second s'appelle "événements". Il contient les événements susceptibles de se produire pour le composant sur lequel nous avons cliqué.

Positionnez-vous dans le composant button1 (1 seul click) et placez-vous dans l'éditeur d'objets, (on attaque les travaux pratiques). Dans propriétés/caption saisissez "Quitter". Ensuite rendez-vous, en cliquant sur l'onglet correspondant, dans la partie "événements". Choisissez dans la liste "on click" et double-cliquez dans la case vide sur la droite de on click. Ciel ! nous revoilà dans le listing. Mais cette fois, regardez ce qui est écrit : nous sommes dans une procédure (elle se situe dans la zone "implémentation" : c'est là que seront stockées toutes les procédures que nous utiliserons). Nous allons indiquer au programme comment il doit se comporter si l'événement on click se réalise, à savoir si l'utilisateur clique sur le bouton "Quitter". Dans la structure de commande, entre les mots-clés "begin" et "end;", nous saisissons directement au clavier :

```
close ;
```

(n'oubliez pas le point-virgule : dans cet exemple, cela n'a pas grande importance, mais si nous utilisions une procédure plus complexe, cela serait important car ce caractère sert de séparateur entre différentes lignes. Alors je pense que le mieux est de prendre l'habitude de le mettre.)

Nous allons à présent passer en phase de test de notre programme. Nous ne lui avons pas encore expliqué comment faire la conversion, nous n'aurons donc pas de résultat, mais nous allons pouvoir tester si en cliquant sur le bouton "Quitter", nous quittons réellement l'application.

Dans la barre des icônes de gauche, nous cliquons sur la flèche verte (qui est en fait un triangle pointé vers la droite). L'environnement change, notre fenêtre avec son bouton apparaît (ainsi que l'onglet où nous avons saisi notre procédure). Clickons dessus : Miracolo, eppur si muova (Pardon, ça m'a échappé) : nous quittons l'application et nous revenons sous l'EDI (Environment Developpement Interface) que nous venions de quitter.

Une astuce au passage, qui a souvent été évoquée, et qui donne un petit plus à votre application : pour faire comme les vrais pros, dans le caption du bouton "Quitter", mettez & devant Quitter, le texte ne sera pas changé, à part la lettre Q qui sera désormais soulignée, mais vous pourrez quitter l'application en faisant Ctrl+Q comme les grands. Comme quoi, en programmation de petits efforts peuvent donner de grands effets.

Vous allez me dire : c'est bien beau tout ça, je quitte l'application, mais à quoi ça me sert si je n'y fais rien dans cette application. Je vous répondrais alors "vous avez bien raison, mais lorsque nous allons mettre au point notre programme, ce petit bouton nous sera utile pour éviter de nous ballader trop souvent dans les menus". Allons maintenant au fait : comment saisir des valeurs et les traiter ?

VI) On attaque le gros morceau

6-1 Les variables forever in my life

Si nous cliquons sur l'onglet "unit1" qui dépasse sous notre programme, nous constatons que le listing est désormais modifié. Notre procédure Button1.Click (SenderToobject()) que nous venons de saisir est apparue dans la partie implémentations. C'est là que devront se trouver toutes les procédures qu'elles soient crées par Delphi comme nous l'avons vu, ou que nous les saisissions à la main, comme nous allons le faire maintenant.

aparté-

Il va de soi qu'il faut sauvegarder régulièrement votre travail pour éviter les surprises désagréables du style panne de courant ou mains d'enfants qui pianotent quand vous avez le dos tourné (rigolez pas comme des baleines ça m'est arrivé pas mal de fois, et contrairement à ce que vous croyez, ça vous arrivera aussi). Pour ce faire cliquez sur la valise en haut, à gauche (celle qui est la plus à droite des 2. Celle qui est la plus à gauche sert à charger un programme que vous aviez préalablement sauvegardé.). Une boite de dialogue vous demandera une première fois le nom de sauvegarde. Attention, il s'agit simplement de la sauvegarde du "listing" que nous venons d'évoquer. Delphi vous propose une sauvegarde avec extension .PAS, signe que votre programme est rédigé en Pascal. Pour l'instant, à notre niveau, nous pouvons nous contenter de sauvegarder avec les options par défaut. Ensuite, Delphi récidive et vous repose la même question. Cette fois, il s'agit du nom réel de votre programme. Il est suivi de l'extension .DPR. C'est seulement à ce moment que sont sauvegardés la feuille que vous venez de créer avec ses "zoulies" couleurs, ses boites, et autres boutons. D'autres sauvegardes sont effectuées en même temps, mais cela n'offre aucun intérêt pour l'instant. Pour votre sauvegarde, choisissez un nom évocateur, évitez "toto"...

revenons à nos moutons-

Avant toute chose, rappelons que pour Delphi, il faut systématiquement déclarer les variables. Cette formule étrange veut simplement dire que Delphi a besoin qu'on lui dise quelles sont les variables qui vont être utilisées, et dans quelle catégorie les ranger. Pourquoi une telle contrainte, alors que dans la plupart des langages, on peut très bien s'en passer ?

Tout simplement, car Delphi est un langage qui se compile. Cela signifie que Delphi, pour gagner en efficacité (vitesse d'exécution, place mémoire, etc...) transforme le texte du listing en un langage spécial directement compréhensible par le microprocesseur, alors que les autres langages doivent être traduits mot-à-mot lors de leur exécution. Dephi doit donc au préalable, réserver en mémoire des emplacements pour les variables, et leur affecter une taille plus ou moins importante selon le type de donnée à y conserver.

Pour ne pas alourdir inutilement notre exposé, notre démarche se limitera à quelques types de variables :

les "string" concernent les **chaines de caractères** (les mots que vous lisez en ce moment même**...**) **sans espaces**

les "intéger" qui sont les entiers; c'est-à dire les nombres sans virgule.

les "real" qui sont les autres, ceux que l'on appelle "à virgule flottante".

pour préciser ces notions importantes, voici des exemples :

anticonstitutionnellement est de type string
toto12 est de type string
1 est de type integer
123000 est de type integer
3,14152654853 est de type real
12530,6 est de type real

Pour notre température, si nous savons que nous donnerons toujours la température en chiffres entiers (par ex 18 pour 18°C), nous pourrons utiliser des types integer. Par contre, si nous pensons qu'un jour, nous risquons d'avoir une température de la forme 20,5 (pour 20,5°C) alors nous serons obligés d'utiliser le type real pour les déclarer. Dans notre exemple, nous allons les déclarer de type real.

Pour ce faire, nous allons, en cliquant sur l'onglet "unit1", dans le listing, et, grâce aux ascenseurs, nous nous positionnons dans la partie "public" et nous saisissons au clavier :

```
DegréFarenheit, DegréCelsius : real ;
```

Nous venons d'indiquer à Delphi que les variables DegréFarenheit et DegréCelsius sont de type real et le fait de les avoir déclarées dans la section "public" fait qu'elles pourront être utilisées par n'importe quelle procédure de notre programme. (n'oubliez toujours pas le point-virgule.)

Il ne nous reste plus désormais qu'à écrire cette fameuse équation qui transforme des degrés Celsius en degrés Farenheit.

Nota : Ici nous allons aborder un chapitre délicat. En effet, lorsque vous saisirez la valeur que vous souhaiterez convertir dans l'éditbox correspondante, vous vous rendrez compte que seules des chaînes string peuvent y être saisies. Le meilleur exemple en est que comme virgule, vous ne pourrez utiliser que celle du clavier, et non celle du pavé numérique, qui génère un . et qui caractérise un chiffre. Si vous essayez, le programme se bloquera. Il vous faudra alors faire la manipulation suivante :

1) aller dans la barre de menu à l'option "réinitialiser"
2) cliquer OK au message qui apparaîtra pour vous prévenir que vous risquez de perdre...

La solution consiste à transformer cette chaîne de type string en variable de type real. Pour ce faire, nous allons utiliser l'instruction StrToReal. Pour l'affichage de la valeur, il faudra bien entendu faire l'inverse, grâce à l'instruction RealToStr (quelle surprise !), qui, comme son nom l'indique accomplit la transformation inverse, d'une variable real en chaine string.

6-2 On voit le bout du tunnel

La formule finale sera donc :

```
DegréFarenheit := (((StrToReal(DegréCelsius))*9)/5)+32
```

Or, nous venons de voir que DegréFarenheit était de type real : il nous faut donc le transformer pour l'afficher. Nous connaissons l'instruction nécessaire, mais comment allons-nous prévenir le programme de l'afficher dans l'éditbox ?

Pour ce faire, nous allons voir la dernière chose à savoir de cette initiation. Vous vous rappelez que l'éditeur d'objet affichait une liste de propriétés particulières pour un objet donné. Si nous les considérons comme des sous-menus, il suffit d'indiquer à Delphi, quel chemin, quelle arborescence suivre, pour afficher correctement le résultat. Nous affichons dans une boite editbox, un texte quelconque grâce à la propriété text, c'est ce que nous allons indiquer à Delphi :

```
editbox2.text := RealToStr(DegréFarenheit )
```

Notez bien que l'on vient de faire la conversion inverse de la variable numérique en chaine que l'on peut afficher. (pensez aussi que la virgule est celle du clavier et non celle du pavé numérique.)

Dans le cas, ou vous auriez souhaité utiliser des entiers, il aurait fallu indiquer StrToInt et IntToStr

Ce procédé s'adapte également à toutes les autres propriétés : par exemple, nous aurions pu modifier la propriété caption en indiquant editbox2.caption , ou même Form1.caption et ainsi afficher le résultat de notre calcul dans le bandeau bleu de notre application. Vous venez de découvrir la règle générale : pour modifier l'aspect du composant, il suffit d'indiquer à Delphi : "nom du composant" . "caractéristique" = ce que vous voulez y voir...

Il va de soi que nous aurions pu tout réaliser en 1 seule opération, mais que cela aurait nui à la clarté de l'exposé.

Pour déclencher l'événement qui lancera ce calcul, il faut choisir entre 2 solution : soit l'on crée 1 bouton comme nous l'avons fait pour "quitter", mais cette fois, en l'appelant "conversion" et en saisissant au lieu de close ; les 2 formules de calcul et l'affichage, ou bien encore, tout bêtement, choisir comme événement on exit. Cela vous confirme s'il en était encore besoin, la puissance de Delphi, et sa souplesse d'utilisation.
Si vous voulez vous entraîner, je vous invite à prévoir le cas inverse pour transformer des degrés Farenheit en dergrés Celsius. Si vous êtes malins, vous n'aurez que des parties de 2 lignes à modifier...

Comme vous êtes sympas, je vous livre le secret : Le secret de l'informatique, le seul à connaître, c'est que plus vous pratiquez, plus tout cela vous paraîtra simple.

VII) Conclusion

Nous voici à la fin de cet article d'initiation. Les principales notions de Delphi ont été abordées, mais rien ne vaut l'expérimentation pour progresser. Il va de soi que cet exemple n'a de valeur qu'éducative, mais qui sait ? peut-être, à partir de cet exemple, pourrez-vous développer une application, par exemple de conversion de litres en gallons, de degrés Farenheit en Celsius ou autre chose et l'envoyer à DP Presse... C'est là tout le mal que nous vous souhaitons !

© Lindor Garfield
lgarfield@fr.europost.org