Lilian BUZER - Dept. Informatique

Prise en main de l'IDE Visual C++ 2010

I - Introduction

Présentation

Un Environnement de Développement Intégré (EDI) ou IDE en anglais (Integrated Development Environment) est un logiciel regroupant au minimum les fonctionnalités suivantes : un éditeur de texte, un compilateur, des outils d'aide à la programmation et un débogueur. Un EDI fournit aussi des outils et des bibliothèques propres permettant de créer des interfaces graphiques IHM (Interface Homme-Machine en français) ou GUI (Graphical User Interface en anglais). Nous allons utiliser l'IDE Visual C++ 2010 en vous présentant successivement le menu, les différentes fenêtres de travail, la fenêtre d'édition, les barres d'outils et l'interface de débogage.

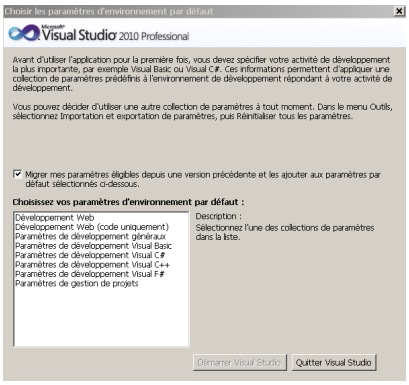
Historique

Depuis Visual 6.0 datant de 1998, Microsoft avait du mal à implanter un nouvel IDE connaissant un vif succès. Les versions qui lui succédèrent : Visual 2002 (7.0) et Visual 2003 dot net (7.1) ne générèrent pas un grand engouement. En effet, les environnements de développements des principaux concurrents (Borland, Eclipse, Linux,…) étaient depuis fort longtemps plus avancés et plus faciles à utiliser que les produits Microsoft. Ainsi, Visual C++ 6.0 malgré son côté spartiate est resté un standard pendant de nombreuses années. Fin octobre 2005, Visual Studio .Net 2005 (8.0) arrivait parmi nous et ce fut enfin un vif succès. Les développeurs le décrivirent comme un environnement orienté vers la « productivité » permettant d'accomplir de nombreuses tâches avec facilité et rapidité. Une version allégée « Visual 2005 Express » fût éditée gratuitement. Assez restreinte, cette version contient les principaux composants utiles que nous verrons en cours. Cependant, nous vous mettons en garde car les projets créés sous la version complète seront incompatibles avec la version Express. Ensuite, lui succéda, Visual 2008 (9.0) sans important changement de l'IDE et la version actuelle Visual 2010 (10.0) qui abandonna la traditionnelle couleur d'interface -gris souris- pour un mélange entre bleu roi et kaki, on n'arrête pas le progrès…

Lancez Visual C++

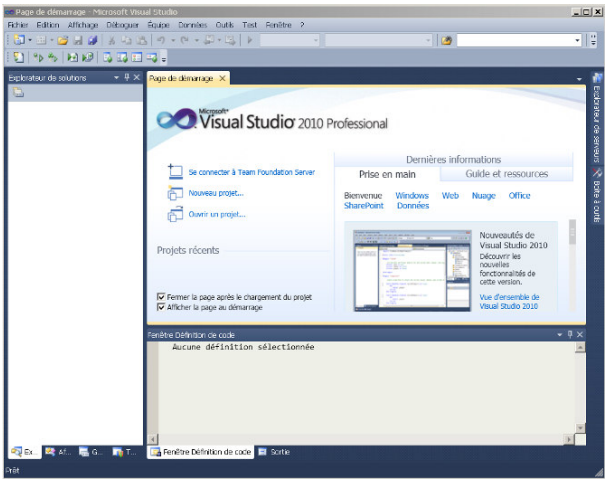
Le programme, si vous ne l'avez pas encore mis dans vos raccourcis du bureau peut se lancer à partir de Démarrer > Programmes > Microsoft Visual Studio

2010 > Microsoft Visual Studio 2010. La fenêtre de lancement s'ouvre. Si vous ne vous êtes jamais connecté sur la station actuelle, le démarrage peut prendre du temps car il faut créer votre profil ou initialisé le premier lancement du logiciel. Après cela, vous arrivez à une fenêtre de sélection où vous choisissez de travailler en C++.

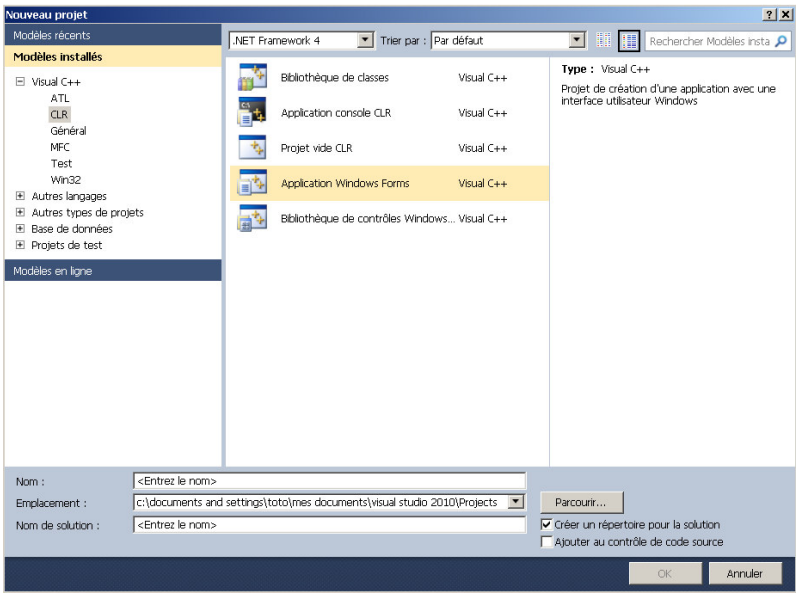


Créer son premier projet

Vous vous retrouvez face à une interface incluant diverses fenêtres, interface un peu déroutante au début :



Nous allons créer un premier projet succinct nous permettant de découvrir l'environnement. Pour cela, allez dans le menu en haut à gauche et choisissez : Fichier → Nouveau → Projet. Dans la liste à gauche choisissez : Visual C++ CLR. Les autres choix correspondent à d'autres versions des bibliothèques Microsoft. Par exemple, win32 est la plus veille bibliothèque disponible. Elle correspond à la génération Windows98 et est écrite en C. Pour réactualiser cette bibliothèque pour le langage objet C++, Microsoft dût la réécrire entièrement et ce fût la naissance de la bibliothèque MFC (Microsoft Fondation Class). Dans ce contexte de migration, beaucoup de développeurs reprochèrent à la MFC d'être restée trop proche des concepts de l'API win32. De plus, comme beaucoup d'autres objets furent intégrés à la MFC dans la foulée, cela amena une des plus belles usines à gaz microsoftiennes qui porta atteinte à l'intégrité psychique de bon nombre de programmeurs. Aujourd'hui, une version plus moderne est disponible grâce à CLR. Comme vous le verrez au chapitre suivant, nous y trouverons une gestion de l'IHM intuitive et facile à utiliser même pour les débutants.



Choisissez ensuite à droite « Application Windows Forms » traduction maladroite de « Windows Forms Application ». Si vous vous baladez dans les autres options du menu de gauche (je doute que vous n'ayez pas eu cette envie), vous remarquerez qu'à chaque fois, on vous propose dans la fenêtre de droite une version de projet de type « Console » ou de type « Fenêtrée = Form ». Face à ce choix, il faudra décider dés le début ce que vous voulez car il sera impossible de changer par la suite.

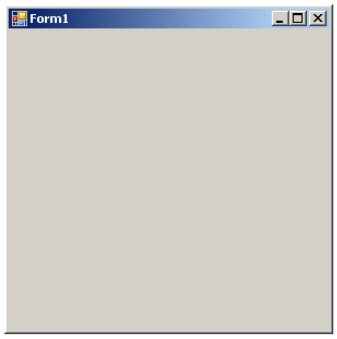
Les applications « console » correspondent aux applications en mode texte (pensez au DOS, au terminal Linux, au minitel !!). Elles n'ont pas de fenêtres avec des boutons ou d'autres interfaces visuelles, elles n'utilisent pas la souris et ne peuvent pas gérer de graphisme. Elles correspondent au programme C/C++ contenant un seul point d'entrée par la fonction main().

Les applications « fenêtrées » correspondent à ce que l'on rencontre tous les jours. Ces applications ne peuvent pas faire de sortie texte à partir d'un simple « printf ». De plus, elles ne contiennent pas de main() !!! (Pour être exact, il en existe un, mais vous n'y avez pas accès). Les points d'entrée de votre programme sont multiples et sont appelés suite à un évènement utilisateur du type : click sur un bouton, déplacement de la fenêtre, déplacement de la souris dans cette fenêtre…

Pour continuer, donnez un nom original à votre projet (test, toto, TP1…) et vérifiez que le répertoire de travail correspond à un emplacement où vous avez les droits d'écriture (c:\temp pour l'ESIEE ou le bureau windows). Evitez de choisir votre répertoire réseau (U:\moncompte), car lors des étapes de compilation (transformation du code source en exécutable) beaucoup d'accès disque sont nécessaires. Si le disque réseau est fortement sollicité à ce moment là, le temps de compilation passe facilement de quelques secondes à quelques dizaines de secondes.

Par contre, profitez de votre disque réseau pour stocker une archive de votre travail à la fin de chaque séance de TP. En début de séance, copiez la dernière archive sur le répertoire de travail local (c:\temp ou le bureau). Une fois la séance de TP terminée, supprimez les répertoires et sous-répertoires Debug et Release stockant les fichiers de compilation intermédiaires sans intérêt pour vous et surtout prenant plusieurs Mo. Il vous reste les fichiers de projets et les sources du programme, faites maintenant une nouvelle archive (zip,rar…) du répertoire de projet et stockez la sur votre disque réseau. C'est la manière la plus efficace de travailler et ainsi vous conservez de plus les versions antérieures de votre programme.

Une fois ces paramètres rentrés, cliquez sur OK. Visual va alors créer plusieurs fichiers qui vont servir de base à votre projet, le disque dur mouline et nous attendons quelques secondes. Une fois que Visual répond à nouveau, la maquette de votre projet a été finalisée. Pour tester si tout s'est bien passé, appuyez sur F5, ce qui provoque la compilation et l'exécution du programme. Vous devriez voir apparaître la fenêtre suivante :



Un bug ? Un trou noir dans le système d'exploitation ? Non, une simple fenêtre vide qui attend vos modifications.

II Le Menu

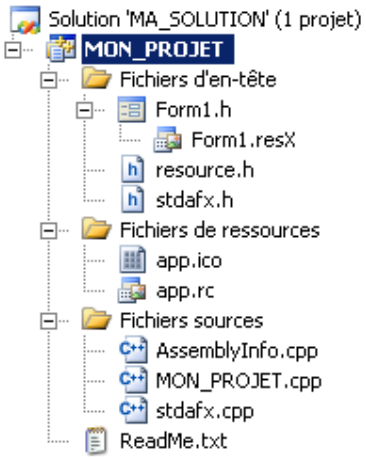
La seule chose qui ne se déplace pas dans l'interface de Visual, c'est le menu !!! Positionnez en haut, juste sous le bandeau de l'application, il vous permet d'accéder à divers sous-menus :

Nous rappelons que vous pouvez utiliser les raccourcis clavier pour sélectionner un sous-menu particulier. Pour cela, maintenez la touche ALT enfoncée et vous allez voir certaines lettres qui se soulignent dans les titres des sous-menus. L'appui sur la touche adéquate permet alors d'ouvrir le sous-menu correspondant. Vous pouvez aussi appuyer et relâcher la touche ALT, un encadrement bleu apparaît sur un des titres du menu. En le déplaçant grâce aux flèches du clavier et en tapant sur ENTREE vous pouvez ainsi ouvrir le sous-menu correspondant.

III - Les fenêtres de l'environnement

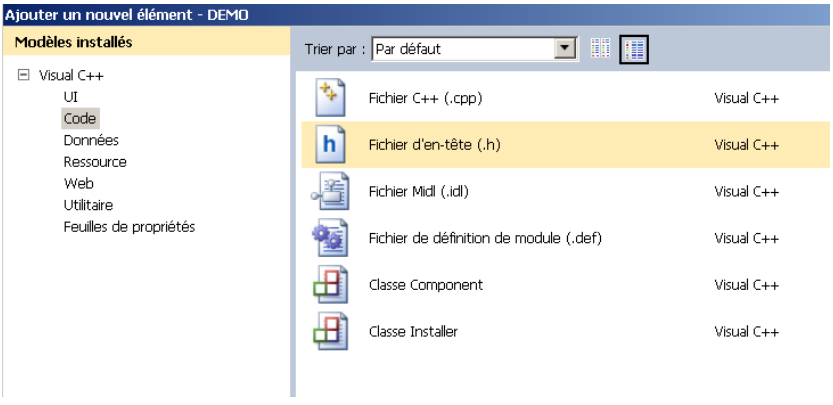
Il existe une multitude de fenêtres sous l'environnement Visual, il n'est pas rare de les perdre et de ne plus savoir comment les retrouver. Les fenêtres s'organisent en quatre positionnements : au dessus, en dessous, à droite et à gauche. Vous comprendrez rapidement que la position n'est pas un critère d'importance car leurs localisations est facilement personnalisables. Généralement ces fenêtres s'ouvrent grâce au Menu Affichage. Cependant, Visual aura pu les déplacer dans Menu > Affichage > Autres fenêtres.

L'explorateur de solutions (Menu : Affichage → Explorateur de Solutions)



Cette fenêtre correspond à un mini explorateur de fichiers qui permet d'accéder rapidement et simplement à l'ensemble des fichiers de votre projet. Un simple double click sur leur nom a pour effet d'ouvrir directement le fichier dans la fenêtre d'édition. A vous de tester.

Très important : pour ajouter des fichiers de code (*.h,*.cpp) à votre projet vous devez utiliser cette interface. En effet, tout fichier non listé dans l'explorateur de solutions ne sera tout simplement PAS COMPILE et bon nombre d'erreurs vont alors apparaître car il manquera des fonctions dans votre programme.

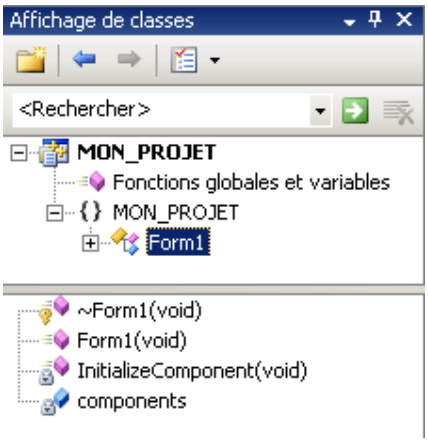


Pour créer un fichier vide, faites un click droit dans l'explorateur sur un répertoire (Fichiers d'en tête ou Fichiers sources) puis Ajouter → Nouvel élément. Dans la fenêtre, choisissez alors « code », modèles : cpp (pour un fichier source) ou .h (pour un fichier en-tête) puis donnez un nom au fichier. Votre nouveau fichier apparaît dans l'explorateur par la suite.

Si vous voulez utiliser des fichiers provenant de l'extérieur, copiez les dans le répertoire de travail et faites Ajouter → élément existant. Vous pouvez aussi ajouter plusieurs fichiers en faisant une sélection multiple dans l'interface (pour cela appuyez sur shift ou ctrl tout en cliquant sur les fichiers qui vous intéressent).

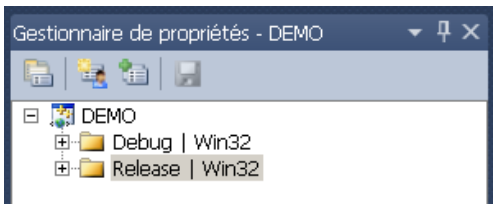
Très important 2 : pour des raisons d'optimisation de vitesse de compilation, un fichier d'entêtes précompilés est créé automatiquement par Visual. Il permet d'éviter de recompiler à chaque fois toutes les centaines de milliers de déclarations qui proviennent des headers de Windows, de Visual et de la GUI. Pour que cette fonctionnalité opère correctement, il faut que TOUS vos fichiers CPP commencent par cette ligne : `#include "stdafx.h"`. Si vous l'oubliez, vous aurez lors de la compilation le message d'erreur évasif suivant : `fatal error C1010: fin de fichier inattendue` !!!

Affichage de classes (Menu : Affichage → Affichage de classes)



Cette interface ne tient plus compte de l'organisation des fichiers. Elle liste dans la fenêtre supérieure l'ensemble des classes développées dans votre projet comme un annuaire. Le click sur le nom d'une classe fournit l'affichage de ses méthodes et de ses paramètres dans la fenêtre inférieure. Un double-click sur leurs noms vous fera accéder dans la fenêtre d'édition directement au code définissant cette fonction/variable. Pratique non ? Dans la fenêtre supérieure, vous trouvez aussi la liste des fonctions/variables globales en cliquant sur la ligne correspondante.

Le gestionnaire de propriétés



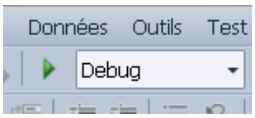
Menu : Affichage → Gestionnaire de propriétés. Il existe deux types de compilation dans les environnements de développement : « Debug » et « Release ».

Le mode Debug permet d'accéder aux fonctionnalités de débogage. En effet, lorsque le programme s'arrête sur un breakpoint, cela est rendu possible car des informations supplémentaires ont été ajoutées dans le programme exécutable. Plus les options d'optimisation seront désactivées, plus vous aurez de fonctions de débogage fonctionnelles et plus votre exécution sera lente. Comptez une perte de 30 à 50% d'efficacité en mode Debug.

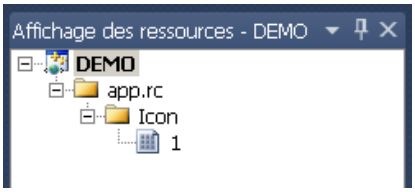
Le mode Release correspond au mode utilisé pour produire le programme distribué aux utilisateurs. Un maximum d'optimisation sont activées et le débogage est impossible, les breakpoints sont tout simplement inactifs dans ce mode. L'inconvénient majeur, hormis l'absence de débogage, est la fragilité de l'environnement face aux bugs. Ainsi, si en mode Debug lorsqu'un bug majeur stoppe votre programme, vous aurez la possibilité de reprendre la main et de diagnostiquer les variables sur la ligne produisant l'erreur. Mais, en mode Release, le programme s'arrêtera brutalement et l'environnement de développement pourra

aussi dans la foulée subir quelques dégâts (comportement anormal, blocage des menus, impossibilité de relancer le programme…). Les deux modes coexistent et vous les choisirez suivant ce que vous voulez faire. Vous pouvez précisément optimiser les options pour chacun de ces modes. Pour voir cela, faites un click droit sur – Release | Win32 – puis Propriétés communes → C/C++ → optimisation. Majoritairement, vous travaillerez 99% du temps en mode Debug et vous ne toucherez pas aux paramètres de compilation.

Vous utiliserez simplement le raccourci présent sous le menu permettant de passez facilement d'un mode à l'autre :

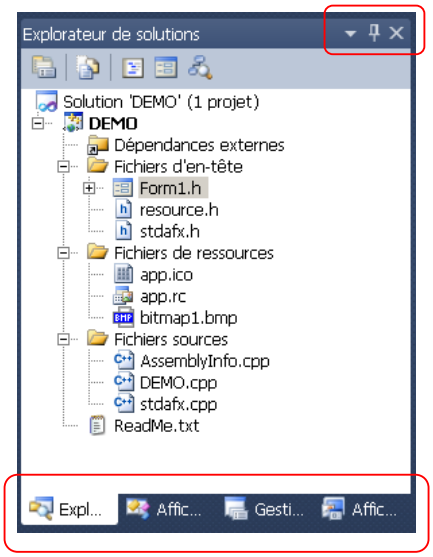


Affichage des ressources (Menu : Affichage → Affichage des ressources)



Les ressources correspondent à des données annexes utilisées lors de l'exécution de votre application. Cela permet d'embarquer directement des images, des sons… dans votre projet. Par exemple, faites un click droit sur app.rc et demandez « ajoutez une ressource », choisissez alors « bitmap ». Vous allez avoir un éditeur d'images succinct qui s'ouvre, créez alors une composition libre. Retournez dans l'explorateur de fichiers, double cliquez sur Form1.h. Faites ensuite un click droit sur votre fenêtre au milieu de l'écran et choisissez propriétés. Allez alors dans Apparence → BackGroundImage et choisissez l'image que vous venez de créer. Faites F5 pour examiner le résultat. Ces manipulations permettent de programmer rapidement sans passer par des logiciels tierces, en contrepartie les mini éditeurs embarqués dans l'environnement offrent des possibilités limitées. Vous auriez pu de manière équivalente créer votre image avec votre logiciel favori, la sauvegarder dans le répertoire du projet et l'associer au projet manuellement.

Naviguer d'une fenêtre à l'autre



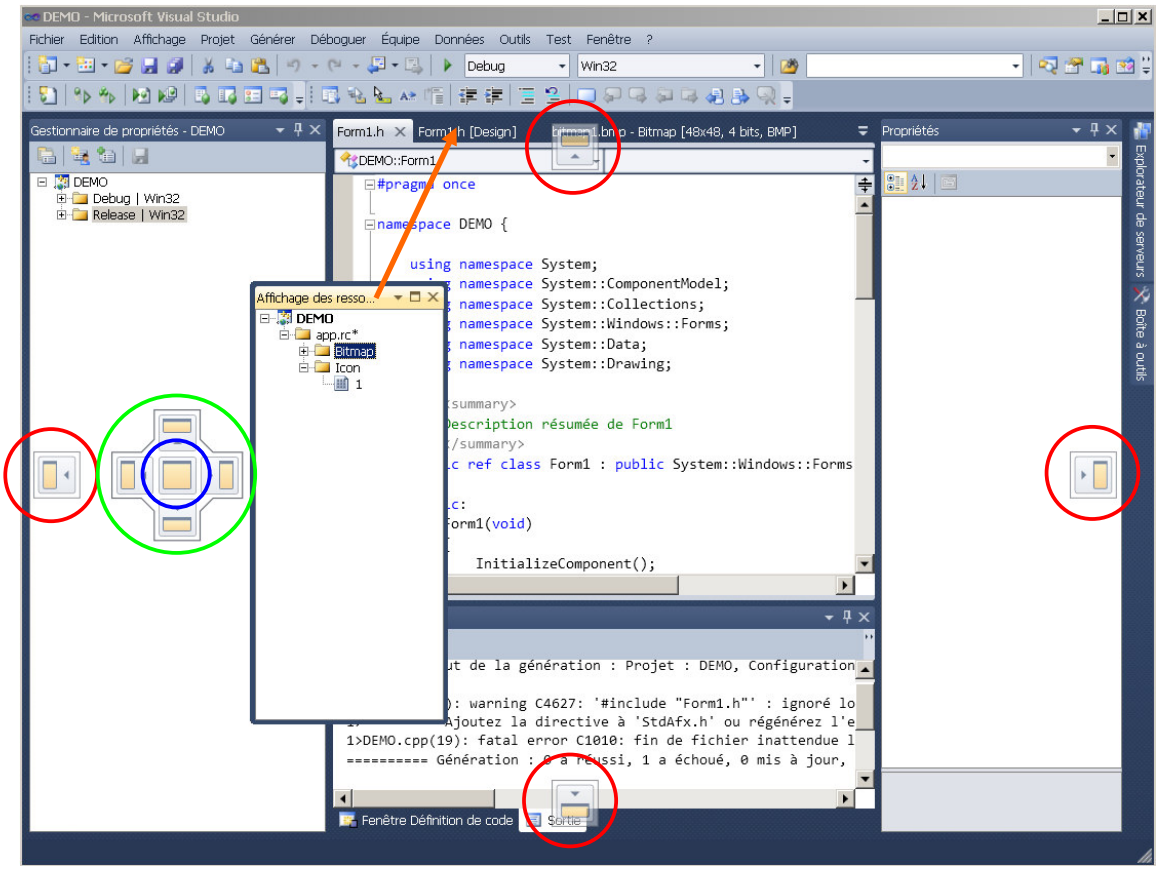
En ayant ouvert ces quatre fenêtres, des onglets sont apparues en bas à gauche. Le simple click sur l'un des onglets fait basculer d'une fenêtre à l'autre.

En haut à droite, vous avez trois icônes spéciaux. La croix fait disparaître l'onglet en cours (attention donc si vous y tenez !!!), il faudra alors le rouvrir en passant par Menu → Affichage… La punaise permet de fixer les onglets, c'est-à-dire qu'ils sont visibles en permanence à l'écran. Si vous décliquez la punaise, les onglets vont se masquer sur le bord gauche de l'écran, voir illustration à droite.

Dans ce cas là, il suffit de passer le curseur de la souris sur le titre de l'onglet pour le faire se rouvrir, en cliquant sur la punaise vous fixez le groupe d'onglets à nouveau. Le petit triangle tête vers le bas fournit l'ensemble de ces fonctions mais sous la forme d'un menu : « masquer » veut dire fermer définitivement (!!), « masquer automatiquement » est équivalent à la punaise, « Ancrer » correspond à la punaise activée et « flottante » permet de décrocher l'onglet et de le transformer en une fenêtre indépendante qui flotte au dessus de l'environnement.

Un peu de fun

Lorsqu'une fenêtre est flottante, on peut faire un click droit sur son bandeau de titre et demander à ce qu'elle devienne ancrable. La fenêtre va se repositionner à sa position précédente. Remettez la fenêtre en mode flottant. Cliquez maintenant sur le bouton gauche de votre souris sur le bandeau de titre de la fenêtre. Ceci fait apparaître à l'écran une interface assez surprenante.



Vous trouvez 4 ancres ici en rouge, permettant de positionner la fenêtre flottante entre le bord de l'écran et la fenêtre désignée par l'ancre. Faites glisser cette fenêtre sur une des quatre ancres (en haut, en bas, à gauche, à droite). Lorsque vous approchez d'une ancre, l'endroit où la fenêtre va être localisée est

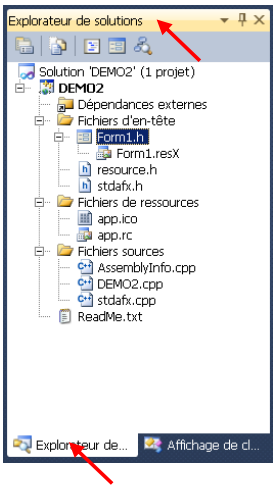
indiqué par une ombre bleue à l'écran. Si cela vous convient, vous pouvez alors relâcher le bouton de la souris.

En vert se trouve une interface de positionnement active seulement dans la fenêtre où se trouve l'icône de la souris. Elle permet de choisir comment vous voulez positionner la fenêtre flottante par rapport à cette fenêtre (en dessous, au dessus, à droite, à gauche).

Pour grouper deux fenêtres dans une seule avec un système d'onglet, il suffit de choisir le carré au centre de la fenêtre de positionnement (en bleu) ou de positionner le bandeau de la fenêtre sur le bandeau des onglets (flèche en orange).

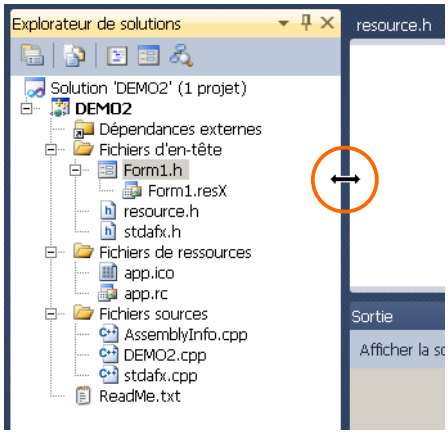
Passez un certain temps à tester les multiples possibilités de cette interface

Ps : parfois l'explorateur de solutions bogue et ouvre les fichiers non pas dans la fenêtre d'édition au centre mais dans sa propre fenêtre (petit farceur). Ce problème peut être corrigé en mettant l'explorateur en fenêtre flottante et en l'encrant tout à gauche.



Si vous voulez déplacer un groupe de fenêtres (plusieurs fenêtres regroupées grâce à des onglets), vous pouvez simplement cliquer sur le bandeau du groupe et faire glisser votre souris, le groupe devient alors flottant. Si vous voulez « décrocher » seulement une fenêtre à l'intérieur du groupe, faîtes la même manipulation en cliquant/glissant sur l'onglet de la fenêtre concernée.

Pour régler la position de la séparation entre deux groupes de fenêtres, il vous suffit de positionner votre curseur de souris à leur frontière, il changera alors de forme et vous pourrez par un simple cliquer/glisser faire changer la position du bord.



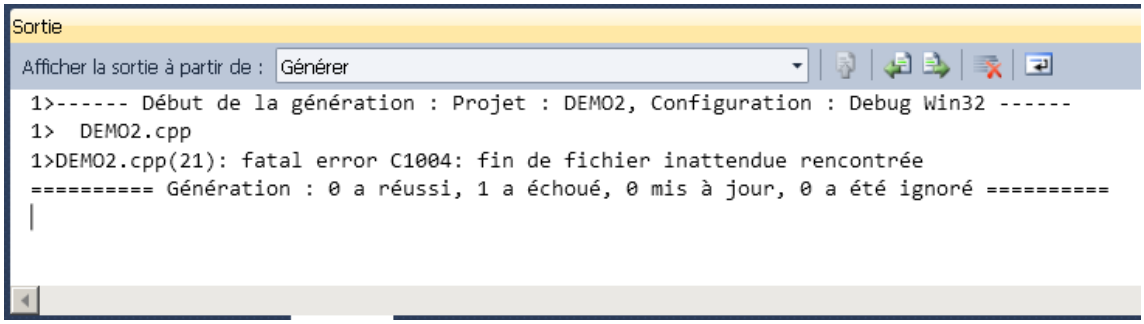
Réinitialisation de l'espace de travail

Si vous fermez une fenêtre et que vous la réactivez par le menu Affichage, elle réapparaîtra à sa dernière position. N'espérez donc pas récupérer les positions

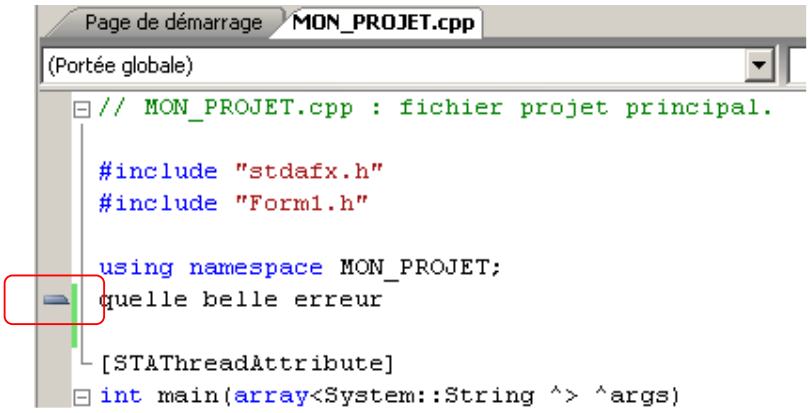
d'origine des fenêtres de cette manière. Pour réinitialiser les fenêtres de l'environnement comme elles étaient à la première ouverture de Visual, effectuez dans le Menu : Fenêtre → Rétablir la disposition de fenêtre.

Fenêtre de sortie (Menu Affichage → Sortie)

Il s'agit d'une console texte dans laquelle le compilateur affiche tous ses messages : warning, erreurs et autres informations. Elle est située dans le groupe des fenêtres du bas de l'écran, veuillez l'activer. Lancez une recompilation du projet : Menu → Régénérez la solution et regardez les messages dans la fenêtre de sortie. Vous voyez apparaître les noms des fichiers qui sont compilés successivement. Allez dans le fichier Mon_nom_de_projet.cpp et tapez n'importe quoi à l'intérieur. Relancez une compilation et examinez les messages d'erreur dans la fenêtre de sortie :

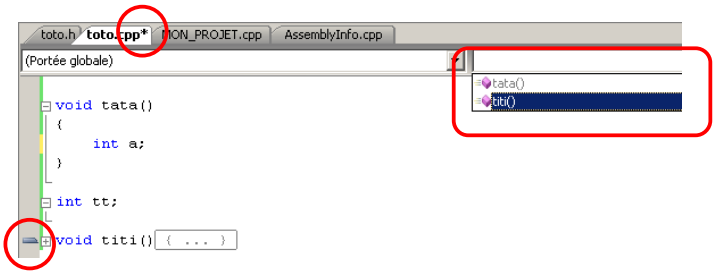


En double cliquant dans la fenêtre de sortie sur une ligne où est citée une erreur : vous faites apparaître une ligne en bleu indiquant l'erreur actuellement sélectionnée. Si vous regardez la fenêtre centrale d'édition de code, vous vous apercevrez que le fichier concerné a été affiché et qu'un marqueur est positionné à la ligne où l'erreur est détectée :



IV Fenêtre centrale d'édition (Menu : Affichage → Code)

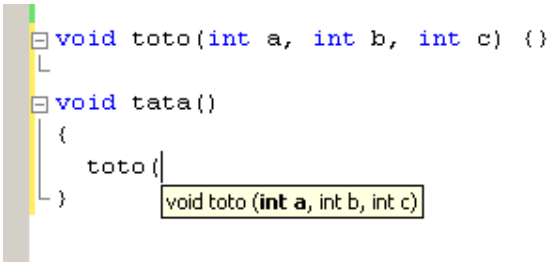
Cette fenêtre est généralement centrée au milieu de l'écran. Fondée sur un système d'affichage par onglet, vous pouvez passer d'un fichier à l'autre en cliquant sur ses onglets. Vous pouvez aussi ouvrir de nouveaux fichiers à partir de l'explorateur de projet vu précédemment.



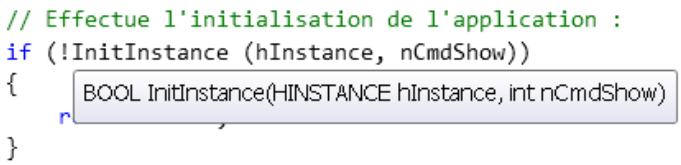
L'étoile signale que des modifications ont été effectuées dans le fichier depuis sa dernière sauvegarde. Un simple CTRL-S permet de sauvegarde le document en cours d'édition et l'étoile disparaît alors.
Le + permet de réduire le code de la fonction sur une seule ligne. Ainsi on peut compacter simplement l'affichage du code source pour augmenter la lisibilité.

En haut à droite, vous trouvez une liste contenant les fonctions du fichier, elle permet d'y accéder rapidement.

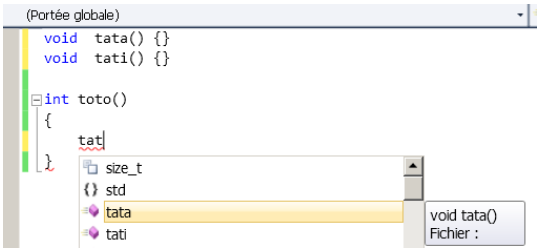
L'affichage automatique des paramètres permet quand vous avez tapé le nom d'une fonction suivi d'une parenthèse ouvrante d'avoir en visuel la liste des paramètres de cette fonction.



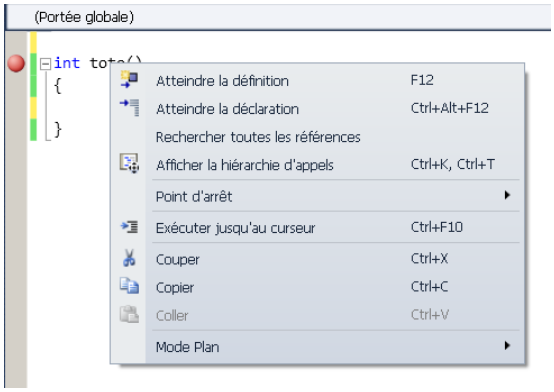
Le module gérant ce type de fonctionnalités s'appellent << intellisense >> sous Visual 2010. Il permet aussi d'obtenir le prototype d'une fonction en mettant le curseur sur son nom dans l'éditeur :



Il permet encore d'auto-compléter les frappes clavier en vous proposant par exemple l'ensemble des fonctions/variables qui commencent par les lettres que vous venez de taper :



ATTENTION : malheureusement dans la dernière version de visual 2010, les fonctions INTELLISENSE ne sont plus actives pour les projets en C++ / CLI (C++ pour les applications fenêtrées dernière génération) alors qu'ils fonctionnaient très bien dans les versions 2005 & 2008. Cause officielle invoquée par l'équipe Microsoft : nous n'avons pas eu le temps d'upgrader l'Intellisense pour les nouvelles fonctionnalités de C++… Nous sommes en plein dans la logique Microsoft, le version suivante du logiciel n'implique par forcément une amélioration de ce dernier mais le plus souvent une régression. Vous ne devriez donc pas avoir cette fonctionnalité sur les machines de l'école ou chez vous.



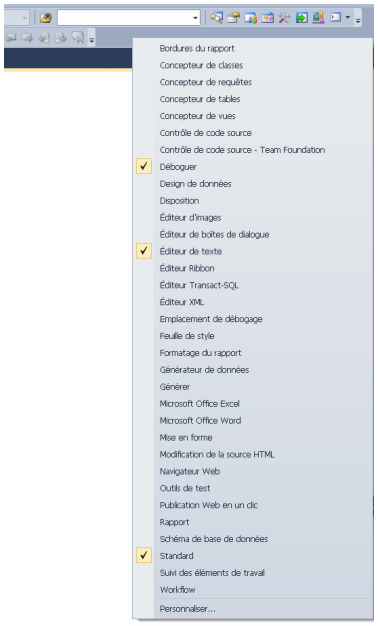
En effectuant un click droit sur une fonction, vous avez accès à diverses opérations comme connaître les fonctions appelantes, trouver les déclarations, trouver la définition. Vous pouvez aussi créer un breakpoint sur la ligne courante, il apparaît alors un rond rouge. Il est plus simple de cliquer directement à l'endroit du rond rouge, pour le créer ou le faire disparaitre. Lorsque le programme en cours d'exécution rencontre un breakpoint, il s'arrête et vous rend la main. Nous étudierons cela dans le chapitre suivant. Un click droit sur le breakpoint permet de lui donner certaines propriétés supplémentaires comme par exemple, s'arrêter si le programme passe sur cette ligne et que la valeur courante de la variable 'i' est égale à 17 ou que la condition 'i≤j+1' est vrai. Le raccourci clavier F9 est très souvent utilisé pour placer un breakpoint sur la ligne courante.

Nous citons des commandes non accessibles directement dans l'éditeur, mais ayant une action à l'intérieur de celui-ci :

- Edition → Rechercher → Recherche rapide : recherche de texte classique
- Edition → Rechercher → Remplacement rapide : transforme une chaîne de caractère en une autre
- Remplacement « dans les fichiers », effectue les recherches et/ou les remplacements à l'intérieur de tous les fichiers du projet. Commande très puissante ; en particulier pour retrouver tous les appels d'une fonction
- Edition → Avancé : vous accédez à plusieurs options de refactoring de code… à regarder par vous-même, vous comprendrez l'utilité. Nous citons particulièrement la commande CTRL+K suivi de CTRL+F qui réécrit tout le code sélectionné en mettant les espaces & tabulations correctes pour un style lisible.

V Les barres d'outils

En dessous du Menu se trouve la zone des barres d'outils. Une barre d'outils regroupe une série d'icônes spécifiques à un thème (compilation, éditeur de fenêtre…). Ces différentes barres peuvent être affichées en passant par Menu → Affichage → Barre d'outils. Vous obtenez la liste complète des barres affichables en faisant un click droit n'importe où sur la zone des barres d'outils :

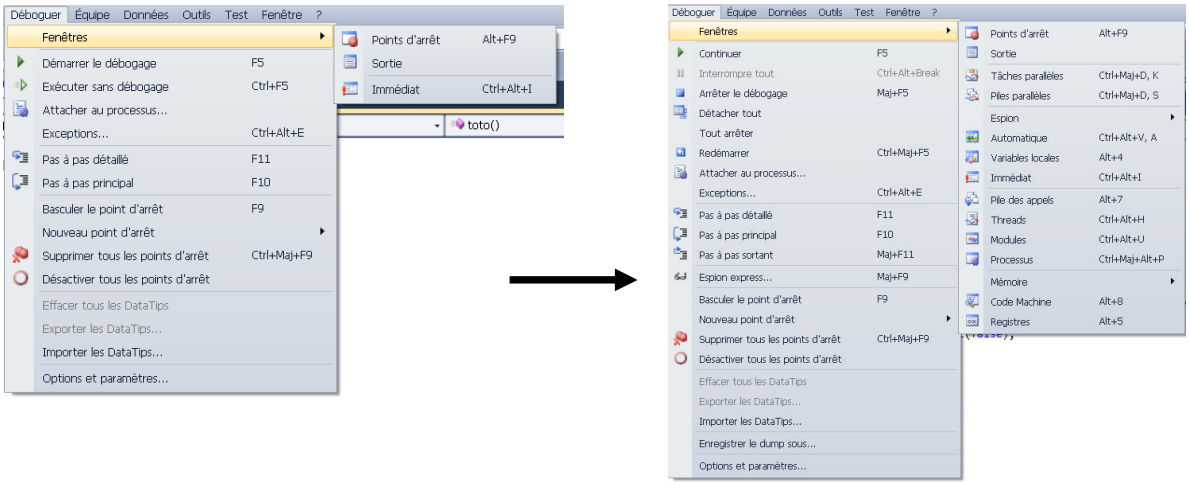


Peu nous serons finalement utiles. Nous utiliserons principalement la barre « Standard » et éventuellement les barres « Debug » et « Générer ». Il faut préciser que toutes les fonctionnalités accessibles par les barres d'outils sont également disponibles à l'intérieur des menus. Il s'agit juste d'une manière plus esthétique et plus pratique d'accéder aux différentes opérations.

Vous pouvez voir ici la barre d'outils Générer. A gauche se trouver une série de points verticaux permettant de déplacer cette barre en cliquant/glissant là où il y a de la place. Vous pouvez la faire bouger dans la zone des barres d'outils, s'en suit alors un véritable jeu de domino où les barres se poussent les unes contre les autres, très énervant à la longue… La flèche à droite de la barre correspond aux options qui permettent d'insérer de nouveaux icônes dans la même thématique. En effet par défaut, une barre d'outils ne présente que les icones des opérations les plus souvent utilisées. Vous pouvez ainsi personnaliser vos barres d'outils.

VI Debug

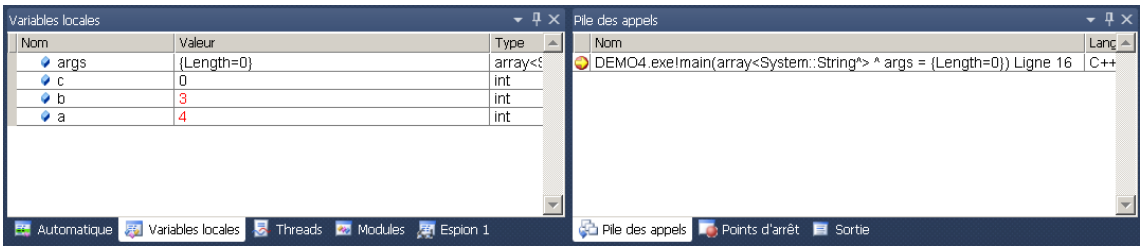
Plusieurs fenêtres sont utilisées spécifiquement pour déboguer un programme. Tout d'abord, ne vous faîtes pas piéger, car certaines options ne sont accessibles uniquement lorsque le programme est arrêté par un breakpoint !! Ainsi le menu Déboguer change complètement d'apparence une fois le programme arrêté par un breakpoint, voir ci-dessous :



Pour connaitre les différentes fonctionnalités du débogage, vous aurez un TP sur ce sujet précis. Nous donnons succinctement le rôle des fenêtres de Debug :

- Variables locales : permet de visualiser les valeurs de toutes les variables locales de la fonction courante. Cela permet d'éviter de taper le nom de chaque variable pour accéder à leur valeur.
- Automatique : identique à variables locales, sauf qu'elle affiche uniquement les variables présentes sur les deux dernières lignes de code parcourues.
- Espion : donnez des expressions, elles seront alors évaluées en permanence. Par exemple : i≥5 ou tout simplement « a » si vous voulez accédez à la valeur de cette variable. Pour les plus gourmands, vous avez par défaut quatre fenêtres espion disponibles, appelées Espion 1 à 4.
- Pile des appels : permet de connaître l'historique des fonctions appelantes. Fonctionnalité très pratique : vous pouvez double cliquer sur une ligne particulière dans cette fenêtre et vous accédez au code de la fonction appelante en question et toutes les valeurs de ses variables locales sont consultables. Vous avez donc accès à toutes les valeurs des variables locales précédant l'appel de la fonction où se trouve le breakpoint.
- Point d'arrêt : liste l'ensemble des points d'arrêts positionnés dans les différents fichiers. Permet de les activer / désactiver par un simple click en cochant ou décochant la ligne concernée. Le rond rouge prend alors un aspect creux pour signifier que le breakpoint est désactivé (qu'est-ce qu'on ne ferait pas pour un débogueur chevronné). Utiliser plus de trois breakpoints dans un programme est très rare, cette fenêtre n'est utile que pour des situations exagérément désespérées ☺

La disposition des fenêtres de débogage est généralement en bas de l'écran sur deux colonnes :



CONCLUSION et CONSEILS FINALS

J'espère que vous avez apprécié cette introduction qui reste relativement très dense. Je liste les derniers conseils qui pourront vous être utiles :

- Export de vos documents : inutile de zipper la totalité de votre répertoire de travail avant de le sauvegarder ou de l'envoyer par mail, même pour un projet vide, les fichiers temporaires de compilation font gonfler l'archive à plusieurs mégas. Pour éviter cela, effacez les répertories Debug ou Release qui peuvent être reconstruits à la volée. Vous n'aurez plus que les fichiers textes et de configuration de taille très faible. Surtout n'oubliez pas d'archiver votre travail de temps en temps, histoire de ne pas tout perdre à cause d'une maudite clef USB qui rend l'âme.

- Recompilation : il se peut que par moment, vous ayez l'impression que Visual déraille et se met à raconter n'importe quoi. Cela arrive, erreur signalée à une ligne où il n'y a rien d'écrit, fonction retournant une valeur correspondant à l'ancienne version de cette fonction, refus de compilation alors qu'un fichier a été modifié… Soyez tolérant !! L'environnement encaisse déjà pas mal de vos erreurs la plupart du temps et puis c'est peut-être votre binôme qui attire la guigne tout simplement. Plus sérieusement, il y a énormément de fonctionnalités déployées pour faciliter la vie du programmeur (auto-complétion, mise à jour en dynamique des informations dans les fenêtres…). Cela a un coût, lorsque ces fonctionnalités se synchronisent mal ou que l'une d'entre elles se bloque, cela peut entraîner des comportements étranges. Je vous conseille alors de suivre les opérations suivantes jusqu'à résolution du problème :

    - Sauvegardez tous les fichiers ouverts dans la fenêtre d'édition, c'est-à-dire Menu → Fichier → Enregistrez tout. Générez la solution (F7) ou Menu → Générer → Générez la solution. Vérifiez
    - Sinon, Lancez une recompilation complète de tous les fichiers. En général, cela est suffisant : Menu → Générer → Régénérez la solution. Vérifiez.
    - Encore là !! Bon, alors demandez un nettoyage des fichiers temporaires du projet : Menu → Générer → Nettoyer la solution. Puis Régénérez la solution. Vérifiez.

- Toujours pas ?? Vous avez peut-être poussé le comique de situation jusqu'à avoir deux fois le même fichier ouvert dans la fenêtre d'édition. L'un contenant le code à jour et l'autre la version d'il y a une heure. Vérifiez.
- Il résiste ? Peut-être avez-vous utilisé deux fichiers différents mais portant le même nom (la mauvaise blague). Exemple : dans mon explorateur, projet.cpp correspond au fichier stocké dans le répertoire du projet, c'est ce dernier qui est compilé et utilisé pour construire le programme, alors que dans la fenêtre d'édition de code j'ai un projet.cpp qui est en fait celui de ma clef usb. Fermez Visual et relancez le, ouvrez votre projet (le bon, pas une version antérieure), fermez tous les fichiers ouverts dans la fenêtre d'édition et rouvrez les en cliquant dans la fenêtre d'exploration de projets. Recompilez.
- Toujours pas ? Un marabout s'impose. Il faut employer les grands moyens. Allez dans le répertoire du projet, effacez à la main les répertoires release et debug que vous croisez. Eteignez la machine. Relancez… Testez
- Désolé, je n'ai plus de conseils en stock. C'est la fin. Ou votre binôme attire vraiment la guigne, ou par chance vous avez une archive…