> **Ce cours est une mise à jour du cours C++ de OpenClassRoom pour le mettre à jour pour le C++11/14. Le cours d'origine est consultable sur la page suivante : Programmez avec le langage C++, par Mathieu Nebra et Matthieu Schaller. Ce cours est sous licence CC-BY-NC-SA.**

Retourner au sommaire principal

Les logiciels nécessaires pour programmer

Maintenant que l'on en sait un peu plus sur le C++, si on commençait à pratiquer pour entrer dans le vif du sujet ?

Ah mais oui, c'est vrai : vous ne pouvez pas programmer tant que vous ne disposez pas des bons logiciels ! En effet, il faut installer certains logiciels spécifiques pour programmer en C++. Dans ce chapitre, nous allons les mettre en place et les découvrir ensemble.

Un peu de patience : dès le prochain chapitre, nous pourrons enfin commencer à véritablement programmer !

Les outils nécessaires au programmeur

Alors à votre avis, de quels outils un programmeur a-t-il besoin ? Si vous avez attentivement suivi le chapitre précédent, vous devez en connaître au moins un !

Vous voyez de quoi je parle ?

Eh oui, il s'agit du compilateur, ce fameux programme qui permet de traduire votre langage C++ en langage binaire !

Il en existe plusieurs pour le langage C++. Mais nous allons voir que le choix du compilateur ne sera pas très compliqué dans notre cas.

Bon, de quoi d'autre a-t-on besoin ? Je ne vais pas vous laisser deviner plus longtemps. Voici le strict minimum pour un programmeur :

- Un éditeur de texte pour écrire le code source du programme en C++. En théorie un logiciel comme le Bloc-Notes sous Windows ou vi sous Linux fait l'affaire. L'idéal, c'est d'avoir un éditeur de texte intelligent qui colore tout seul le code, ce qui vous permet de vous y repérer bien plus facilement. Voilà pourquoi aucun programmeur sain d'esprit n'utilise le Bloc-Notes.
- Un compilateur pour transformer (« compiler ») votre code source en binaire.
- Un débugger (« Débogueur » ou « Débugueur » en français) pour vous aider à traquer les erreurs dans votre programme (on n'a malheureusement pas encore inventé le « correcteur », un truc qui corrigerait tout seul nos erreurs).

A priori, si vous êtes un casse-cou de l'extrême, vous pourriez vous passer de débugger. Mais bon, je sais pertinemment que 5 minutes plus tard vous reviendriez me demander où on peut trouver un débugger qui marche bien.

À partir de maintenant on a 2 possibilités :

- Soit on récupère chacun de ces 3 programmes séparément. C'est la méthode la plus compliquée, mais elle fonctionne. Sous Linux en particulier, bon nombre de programmeurs préfèrent utiliser ces 3 programmes séparément. Je ne détaillerai pas cette solution ici, je vais plutôt vous parler de la méthode simple.
- Soit on utilise un programme « 3-en-1 » (oui oui, comme les liquides vaisselle) qui combine éditeur de texte, compilateur et débugger. Ces programmes « 3-en-1 » sont appelés IDE (ou en français « EDI » pour « Environnement de Développement Intégré »).

Il existe plusieurs environnements de développement. Au début, vous aurez peut-être un peu de mal à choisir celui qui vous plaît. Une chose est sûre en tout cas: vous pouvez faire n'importe quel type de

programme, quel que soit l'IDE que vous choisissez.

Les projets

Quand vous réalisez un programme, on dit que vous travaillez sur un projet. Un projet est constitué de plusieurs fichiers de code source : des fichiers .cpp, .h, les images du programme, etc.

Le rôle d'un IDE est de rassembler tous ces fichiers d'un projet au sein d'une même interface. Ainsi, vous avez accès à tous les éléments de votre programme à portée de clic. Voilà pourquoi, quand vous voudrez créer un nouveau programme, il faudra demander à l'IDE de vous préparer un « nouveau projet ».

Choisissez votre IDE

Il m'a semblé intéressant de vous montrer quelques IDE parmi les plus connus. Tous sont disponibles gratuitement. Personnellement, je navigue un peu entre tous ceux-là et j'utilise l'IDE qui me plaît selon l'humeur du jour.

- Un des IDE que je préfère s'appelle Code::Blocks. Il est gratuit et disponible pour la plupart des systèmes d'exploitation. Je conseille d'utiliser celui-ci pour débuter (et même pour la suite s'il vous plaît bien !). Fonctionne sous Windows, Mac et Linux.
- Le plus célèbre IDE sous Windows, c'est celui de Microsoft : Visual C++. Il existe à la base en version payante (chère !) mais, heureusement, il en existe une version gratuite intitulée Visual C++ Express qui est vraiment très bien (il y a peu de différences avec la version payante). Il est très complet et possède un puissant module de correction des erreurs (débuggage). Fonctionne sous Windows uniquement.
- Sur Mac OS X, vous pouvez aussi utiliser XCode, généralement

fourni sur le CD d'installation de Mac OS X. C'est un IDE très apprécié par tous ceux qui font de la programmation sur Mac.Fonctionne sous Mac OS X uniquement.

> Note pour les utilisateurs de Linux : il existe de nombreux IDE sous Linux, mais les programmeurs expérimentés préfèrent parfois se passer d'IDE et compiler « à la main », ce qui est un peu plus difficile. Vous aurez le temps d'apprendre à faire cela plus tard. En ce qui nous concerne nous allons commencer par utiliser un IDE. Si vous êtes sous Linux, je vous conseille d'installer Code::Blocks pour suivre mes explications.
> Vous pouvez aussi jeter un coup d'œil du côté de l'Eclipse pour les développeurs C/C++, très puissant et qui, contrairement à l'idée répandue, ne sert pas qu'à programmer en Java !
>
> Quel est le meilleur de tous ces IDE ?

Tous ces IDE vous permettront de programmer et de suivre le reste de ce cours sans problème. Certains sont plus complets au niveau des options, d'autres un peu plus intuitifs à utiliser, mais dans tous les cas les programmes que vous créerez seront les mêmes quel que soit l'IDE que vous utilisez. Ce choix n'est donc pas si crucial qu'on pourrait le croire.

Durant tout ce cours, j'utiliserai Code::Blocks. Si vous voulez avoir exactement les mêmes écrans que moi, surtout au début pour ne pas être perdus, je vous recommande donc de commencer par installer Code::Blocks.

Code::Blocks (Windows, Mac OS, Linux)

Code::Blocks est un IDE libre et gratuit, disponible pour Windows, Mac et Linux. Il n'est disponible pour le moment qu'en anglais. Cela ne devrait PAS vous décourager de l'utiliser. En fait, nous utiliserons très peu les menus.

Sachez toutefois que, quand vous programmerez, vous serez de toute façon confronté bien souvent à des documentations en anglais. Voilà

donc une raison de plus pour s'entraîner à utiliser cette langue.

Télécharger Code::Blocks

Rendez-vous sur la page de téléchargements de Code::Blocks.

- Si vous êtes sous Windows, repérez la section « Windows » un peu plus bas sur cette page. Téléchargez le logiciel en choisissant le programme dont le nom contient mingw (ex. : codeblocks-10.05mingw-setup.exe). L'autre version étant sans compilateur, vous aurez du mal à compiler vos programmes.
- Si vous êtes sous Linux, le mieux est encore d'installer Code::Blocks via les dépôts (par exemple avec la commande apt-get sous Ubuntu). Il vous faudra aussi installer le compilateur à part : c'est le paquet build-essential. Pour installer le compilateur et l'IDE Code::Blocks, vous devriez donc taper la commande suivante :

```
apt-get install build-essential codeblocks
```

- Enfin, sous Mac, choisissez le fichier le plus récent de la liste.

> J'insiste là-dessus : si vous êtes sous Windows, téléchargez la version du programme d'installation dont le nom inclut mingw (figure suivante). Si vous prenez la mauvaise version, vous ne pourrez pas compiler vos programmes par la suite !

Code Blocks avec mingw

L'installation est très simple et rapide. Laissez toutes les options par défaut et lancez le programme.

Code Blocks

On distingue dans la fenêtre 4 grandes sections (numérotées dans la figure suivante) :

1. La barre d'outils : elle comprend de nombreux boutons, mais seuls quelques-uns d'entre eux nous seront régulièrement utiles. J'y reviendrai plus loin.
2. La liste des fichiers du projet : c'est à gauche que s'affiche la liste de tous les fichiers source de votre programme. Notez que, sur cette capture, aucun projet n'a été créé : on ne voit donc pas encore de fichier à l'intérieur de la liste. Vous verrez cette section se remplir dans cinq minutes en lisant la suite du cours.
3. La zone principale : c'est là que vous pourrez écrire votre code en langage C++ !

La zone de notification : aussi appelée la « Zone de la mort », c'est ici que vous verrez les erreurs de compilation s'afficher si votre code comporte des erreurs. Cela arrive très régulièrement !

Intéressons-nous maintenant à une section particulière de la barre d'outils. Vous trouverez dans l'ordre les boutons suivants : Compiler, Exécuter, Compiler & Exécuter et Tout recompiler (figure suivante). Retenez-les, nous les utiliserons régulièrement.

Icônes de compilation

- Compiler : tous les fichiers source de votre projet sont envoyés au compilateur qui se charge de créer un exécutable. S'il y a des erreurs (ce qui a de fortes chances d'arriver), l'exécutable n'est pas créé et on vous indique les erreurs en bas de Code::Blocks.
- Exécuter : cette icône lance juste le dernier exécutable que vous avez compilé. Cela vous permet donc de tester votre programme

et voir ainsi ce qu'il donne. Dans l'ordre, si vous avez bien suivi, on doit d'abord compiler puis exécuter le binaire obtenu pour le tester. On peut aussi utiliser le 3ème bouton…
- Compiler & Exécuter : pas besoin d'être un génie pour comprendre que c'est la combinaison des 2 boutons précédents. C'est d'ailleurs ce bouton que vous utiliserez le plus souvent. Notez que s'il y a des erreurs pendant la compilation (pendant la génération de l'exécutable), le programme n'est pas exécuté. À la place, vous avez droit à une liste d'erreurs à corriger.
- Tout reconstruire : quand vous choisissez de Compiler, Code::Blocks ne recompile en fait que les fichiers modifiés depuis la dernière compilation et pas les autres. Parfois (je dis bien parfois) vous aurez besoin de demander à Code::Blocks de vous recompiler tous les fichiers. On verra plus tard quand on a besoin de ce bouton et vous verrez plus en détail le fonctionnement de la compilation dans un chapitre futur. Pour l'instant, on se contente d'apprendre le minimum nécessaire pour ne pas tout mélanger. Ce bouton ne nous sera donc pas utile de suite.

> Je vous conseille d'utiliser les raccourcis plutôt que de cliquer sur les boutons, parce que c'est quelque chose qu'on fait vraiment très très souvent. Retenez en particulier qu'il faut appuyer sur la touche F9 pour Compiler & Exécuter.

Créer un nouveau projet

Pour créer un nouveau projet, c'est très simple : allez dans le menu File > New > Project. Dans la fenêtre qui s'ouvre, choisissez Console application (figure suivante).

Nouveau projet

> Comme vous pouvez le voir, Code::Blocks propose de réaliser bon nombre de types de programmes différents qui utilisent des bibliothèques connues comme la SDL (2D), OpenGL (3D), Qt et wxWidgets (Fenêtres) etc. Pour l'instant, ces icônes n'ont d'intérêt que cosmétique car les bibliothèques ne sont pas installées sur votre ordinateur, vous ne pourrez donc pas les faire marcher.
> Nous nous intéresserons à ces autres types de programmes bien plus tard. En attendant il faudra vous contenter de Console car vous n'avez pas encore le niveau nécessaire pour créer les autres types de programmes.

Cliquez sur Go pour créer le projet. Un assistant s'ouvre.

La première page ne servant à rien, cliquez sur Next. On vous demande ensuite si vous allez faire du C ou du C++ : répondez C++ (figure suivante).

Nouveau projet C++

On vous demande le nom de votre projet et dans quel dossier seront enregistrés les fichiers source (figure suivante).

Nom et dossier du projet

Enfin, la dernière page vous permet de choisir de quelle façon le programme doit être compilé. Vous pouvez laisser les options par défaut, cela n'aura pas d'incidence pour ce que nous allons faire dans l'immédiat. (Veillez à ce qu'au moins Debug ou Release soit coché.)

Mode de compilation

Cliquez sur Finish, c'est bon ! Code::Blocks vous crée un premier projet contenant déjà un tout petit peu de code source.

Dans le panneau de gauche intitulé Projects, développez l'arborescence en cliquant sur le petit + pour afficher la liste des fichiers du projet. Vous devriez avoir au moins un fichier main.cpp que vous pouvez ouvrir en faisant un double-clic dessus.

Et voilà !

Visual C++ (Windows seulement)

Quelques petits rappels sur Visual C++ :

- c'est l'IDE de Microsoft.
- il est à la base payant, mais Microsoft en a sorti une version gratuite intitulée Visual C++ Express.

Nous allons bien entendu voir ici la version gratuite, Visual C++ Express (figure suivante).

Aperçu de Visual C++

> Quelles sont les différences avec le « vrai » Visual ?

Il n'y a pas d'éditeur de ressources (vous permettant de dessiner des images, des icônes ou des fenêtres). Mais bon, entre nous, on s'en moque parce qu'on n'aura pas besoin de s'en servir dans ce livre. Ce ne sont pas des fonctionnalités indispensables, bien au contraire.

Vous trouverez les instructions pour télécharger Visual C++ Express à cette adresse :

Site de Visual C++ Express Edition

Sélectionnez Visual C++ Express Français un peu plus bas sur la page.

Visual C++ Express est en français et est totalement gratuit. Ce n'est donc pas une version d'essai limitée dans le temps.

Installation

L'installation devrait normalement se passer sans encombre. Le programme d'installation va télécharger la dernière version de Visual sur Internet. Je vous conseille de laisser les options par défaut.

À la fin, on vous dit qu'il faut vous enregistrer dans les 30 jours. Pas de panique, c'est gratuit et rapide mais il faut le faire. Cliquez sur le lien qui vous est donné : vous arrivez sur le site de Microsoft. Connectez-vous avec votre compte Windows Live ID (équivalent du compte Hotmail ou MSN) ou créez-en un si vous n'en avez pas, puis répondez au petit questionnaire.

À la fin du processus, on vous donne à la fin une clé d'enregistrement. Vous devez recopier cette clé dans le menu ? > Inscrire le produit.

Créer un nouveau projet

Pour créer un nouveau projet sous Visual, allez dans le menu Fichier > Nouveau > Projet. Sélectionnez Win32 dans le panneau de gauche puis Application console Win32 à droite.

Entrez un nom pour votre projet, par exemple « test ».

Ajout d'un projet

Validez. Une nouvelle fenêtre s'ouvre.

Assistant application

Cette fenêtre ne sert à rien. Par contre, cliquez sur Paramètres de l'application dans le panneau de gauche.

Paramètres de l'application

Veillez à ce que l'option Projet vide soit cochée comme à la figure suivante. Puis, cliquez sur Terminer.

Ajouter un nouveau fichier source

Votre projet est pour l'instant bien vide. Faites un clic droit sur le dossier Fichiers sources situé dans le panneau de gauche puis allez dans Ajouter > Nouvel élément.

Ajout d'un nouvel élément

Une fenêtre s'ouvre. Sélectionnez Fichier C++ (.cpp). Entrez le nom « main.cpp » pour votre fichier.

Ajout d'un fichier

Cliquez sur Ajouter. C'est bon, vous allez pouvoir commencer à écrire du code !

La fenêtre principale de Visual

Voyons ensemble le contenu de la fenêtre principale de Visual C++

Express. On va rapidement se pencher sur ce que signifie chacune des parties :

Fenêtre de Visual C++

1. La barre d'outils : tout ce qu'il y a de plus standard. Ouvrir, Enregistrer, Enregistrer tout, Couper, Copier, Coller etc. Par défaut, il semble qu'il n'y ait pas de bouton de barre d'outils pour compiler. Vous pouvez les rajouter en faisant un clic droit sur la barre d'outils puis en choisissant Déboguer et Générer dans la liste. Toutes ces icônes de compilation ont leur équivalent dans les menus Générer et Déboguer. Si vous choisissez Générer, cela crée l'exécutable (cela signifie « Compiler » pour Visual). Si vous sélectionnez Déboguer / Exécuter, on devrait vous proposer de compiler avant d'exécuter le programme. La touche F7 permet de générer le projet et F5 de l'exécuter.
2. La liste des fichiers du projet : dans cette zone très importante, vous voyez normalement la liste des fichiers de votre projet. Cliquez sur l'onglet Explorateur de solutions, en bas, si ce n'est déjà fait. Vous devriez voir que Visual crée déjà des dossiers pour séparer les différents types de fichiers de votre projet (« sources », « en-têtes » et « ressources »). Nous verrons un peu plus tard quels sont les différentes sortes de fichiers qui constituent un projet.
3. La zone principale : c'est là qu'on modifie les fichiers source.
4. La zone de notification : c'est là encore la « zone de la mort », celle où l'on voit apparaître toutes les erreurs de compilation. C'est également dans le bas de l'écran que Visual affiche les informations de débuggage quand vous essayez de corriger un programme buggé. Je vous ai d'ailleurs dit tout à l'heure que j'aimais beaucoup le débugger de Visual et je pense que je ne suis pas le seul.

Voilà, on a fait le tour de Visual C++. Vous pouvez aller jeter un œil dans les options (Outils > Options) si cela vous chante, mais n'y passez pas 3 heures. Il faut dire qu'il y a tellement de cases à cocher partout qu'on ne sait plus trop où donner de la tête.

Xcode (Mac OS seulement)

Il existe plusieurs IDE compatibles Mac. Il y a Code::Blocks{} bien sûr, mais ce n'est pas le seul. Je vais vous présenter ici l'IDE le plus célèbre sous Mac : Xcode.

> Merci à prs513rosewood pour les captures d'écrans et ses judicieux conseils pour réaliser cette section.

Xcode, où es-tu ?

Tous les utilisateurs de Mac OS ne sont pas des programmeurs. Apple l'a bien compris et, par défaut, n'installe pas d'IDE avec Mac OS. Heureusement, pour ceux qui voudraient programmer, tout est prévu. En effet, Xcode est présent sur le CD d'installation de Mac OS.

Insérez donc le CD dans le lecteur. Pour installer Xcode, il faut ouvrir le paquet Xcode Tools dans le répertoire Installation facultative du disque d'installation. L'installeur démarre (figure suivante).

Installation de Xcode

Par ailleurs, je vous conseille de mettre en favori la page dédiée aux développeurs sur le site d'Apple. Vous y trouverez une foule d'informations utiles pour le développement sous Mac. Vous pourrez notamment y télécharger plusieurs logiciels pour développer. N'hésitez pas à vous inscrire à l'ADC (Apple Development Connection), c'est gratuit et vous serez ainsi tenus au courant des nouveautés.

Lancement

L'application Xcode se trouve dans le répertoire /Developer/Applications/ (figure suivante).

Lancement de Xcode

Nouveau projet

Pour créer un nouveau projet, on clique sur Create a new Xcode project, ou File > New Project. Il faut choisir le type Command Line Tool et sélectionner C++ sdtc++ dans le petit menu déroulant (figure suivante).

Nouveau projet Xcode

Une fois le projet créé, la fenêtre principale de Xcode apparaît (suivante).

Fenêtre principale de Xcode

Le fichier sdz-test (icône noire) est l'exécutable et le fichier sdz-test.1 est un fichier de documentation. Le fichier main.cpp contient le code source du programme. Vous pouvez faire un double-clic dessus pour l'ouvrir.

Vous pouvez ajouter de nouveaux fichiers C++ au projet via le menu File > New File.

Compilation

Avant de compiler, il faut changer un réglage dans les préférences de Xcode. Pour ouvrir les préférences, cliquez sur Preferences dans le menu Xcode. Dans l'onglet debugging, sélectionnez Show console dans la liste déroulante intitulée On start. C'est une manipulation nécessaire pour voir la sortie d'un programme en console (figure suivante).

Options de Xcode

> Cette manipulation n'a besoin d'être faite qu'une seule fois en tout.

Pour compiler, on clique sur le bouton Build and Run (en forme de marteau avec une petite icône verte devant) dans la fenêtre du projet. La console s'affiche alors (figure suivante).

Compilation sous Xcode

Voilà ! Vous connaissez désormais l'essentiel pour créer un nouveau projet C++ et le compiler avec Xcode.

En résumé

- Un IDE est un outil tout-en-un à destination des développeurs, qui permet de créer des programmes.
- Un IDE est composé d'un éditeur de texte, d'un compilateur et d'un debugger.
- Code::Blocks, Visual C++ et Xcode sont parmi les IDE les plus connus pour programmer en C++.
- Nous prendrons Code::Blocks comme base dans la suite de ce cours.

Retourner au sommaire principal

Cours, C++