Centre Informatique pour les Lettres et les Sciences Humaines

Apprendre C++ avec QtCreator
Etape 1 : Hello, World!

Au cours de cette première étape, nous allons faire connaissance avec un logiciel, QtCreator, et acquérir quelques notions élémentaires d'un langage de programmation, C++. Le programme que nous allons créer est une sorte de voyage dans le passé, dans la mesure où son aspect et son mode de fonctionnement évoquent plus l'informatique des années 70-80 que le XXI° siècle.

1 - Création d'un projet

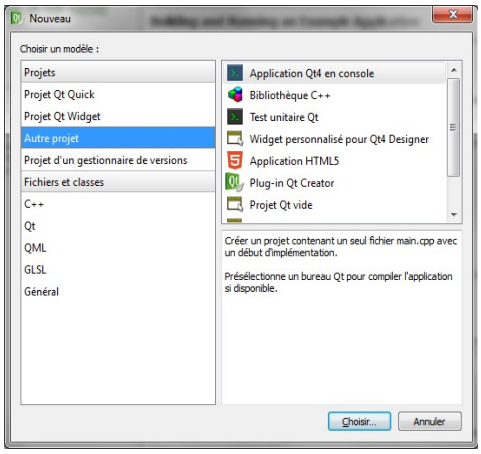
La fenêtre d'accueil de QtCreator (cf. ci-contre) se caractérise par la présence d'un certain nombres d'éléments qui resteront en permanence à notre disposition lors de l'utilisation de ce logiciel :

- Une barre de menu tout à fait classique ;

- Un bandeau gauche et un bandeau inférieur proposant divers contrôles qui permettent un accès rapide aux commandes les plus fréquemment utilisées ;

- Une zone centrale dont le contenu dépend du type d'activité en cours.

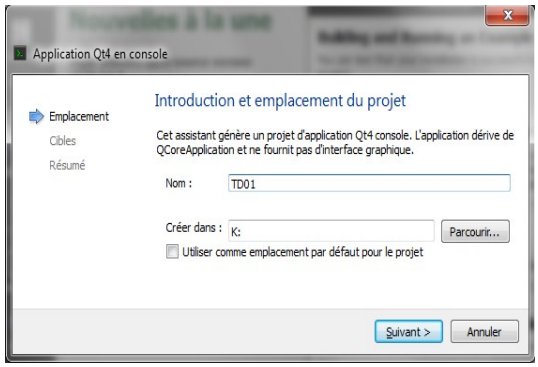
Dans le menu <Fichier>, choisissez la commande <Nouveau fichier ou projet..> ☐.



Le dialogue qui apparaît alors propose différentes options. Nous ne souhaitons pas créer un simple fichier, mais un projet complet, c'est à dire un ensemble organisé de fichiers à partir desquels QtCreator sera capable de générer un programme.

Parmi les types de projets C++ disponibles, choisissez <Autre projet> <Application Qt4 en console> et cliquez sur [Choisir] ☐.

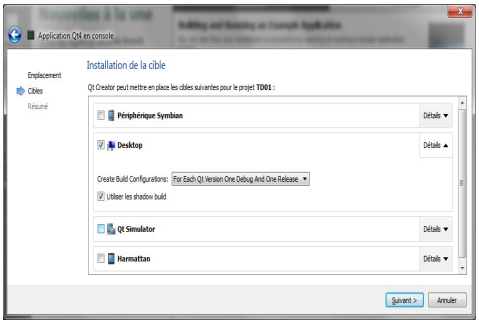
> Dès que nous aurons un peu avancé dans notre maîtrise du logiciel et du langage, nous passerons à des projets générant des programmes pourvus d'une interface graphique (<QtWidget><Application graphique Qt>).

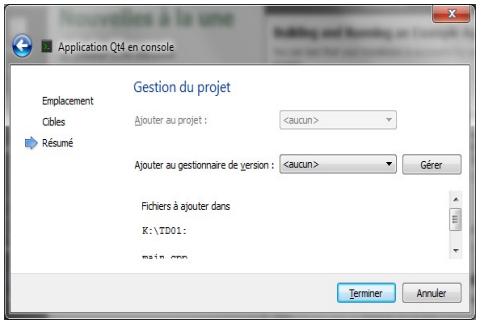


Le dialogue suivant vous permet de choisir un nom et un emplacement où sera créé le dossier à l'intérieur duquel seront rangés les fichiers constituant le projet.

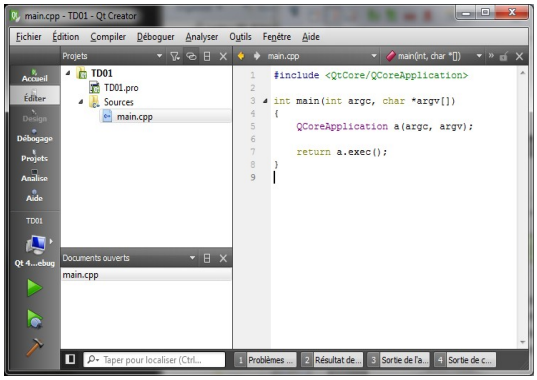
Une fois ces informations fournies, cliquez sur le bouton [Suivant >] ☐.

Les deux fenêtres suivantes (représentées ci-dessous) ne demandent aucune intervention de votre part, cliquez simplement sur le bouton [Suivant] de la première ☐ puis sur le bouton [Terminer] de la seconde ☐.





2 - Le contenu initial du fichier main.cpp



Une fois le projet créé, QtCreator passe spontanément en mode Edition (le mode auquel permet d'accéder le bouton [Edit] qui figure dans le bandeau gauche). Dans ce mode, le logiciel propose par défaut une zone de navigation qui permet de choisir le fichier sur lequel nous souhaitons intervenir.

Deux choix nous sont proposés : un fichier .pro qui décrit la façon dont notre programme va être construit et dont nous n'allons pas nous occuper pour l'instant, et le fichier main.cpp.

Double-cliquez sur ce dernier pour en faire apparaître le contenu.

```
1 #include <QtCore/QCoreApplication>
2 int main(int argc, char *argv[])
3 {
4 QCoreApplication a(argc, argv);
5 return a.exec();
6 }
```

En dépit de sa brièveté, le texte contenu dans le fichier main.cpp appelle de nombreux commentaires et il est fort possible que tout ne vous semble pas clair dès la première séance : tout est nouveau, et il faut un peu de temps et de pratique pour s'habituer et découvrir qu'il n'y a là rien de réellement compliqué.

De quoi s'agit-il ?

D'un texte écrit en C++ (d'où le choix de l'extension .cpp pour le nom du fichier) qui décrit les opérations que notre programme devra exécuter lorsque nous le mettrons en route.

> Attention à ne pas confondre un texte en C++ avec le programme qu'il permet de générer. Des traitements complexes sont nécessaires pour créer effectivement un programme exécutable à partir d'un texte source rédigé dans un langage de programmation tel que C++. C'est QtCreator qui va se charger d'appliquer ces traitements, mais leur succès n'est pas garanti : si le texte source ne respecte pas les règles du langage, il peut s'avérer impossible de l'utiliser pour créer un programme.

Dans son état actuel, le texte source comporte deux parties : une directive d'inclusion (ligne *1*) et la définition d'une fonction (lignes *2* à *6*).

Notion de fonction

La description des opérations devant être effectuées par un programme peut s'avérer très longue (plusieurs milliers de pages, dans certains cas). Pour faciliter la rédaction et la correction de ce texte, on préfère le scinder en unités d'une taille plus maîtrisable, un peu comme on divise le texte d'un roman en chapitres. En C++, ces unités s'appellent des fonctions et leur définition obéit à des règles très précises :

- La définition d'une fonction commence par un en-tête (ligne *2*) qui mentionne le nom de la fonction (l'équivalent du titre du chapitre, `main` dans le cas présent) et différentes informations auxquelles nous n'allons pas nous intéresser au cours de cette première étape.

- L'en-tête est suivi d'un couple d'accolades (lignes *3* et *6*) qui délimitent ce qu'on appelle le corps de la fonction. A l'intérieur de ce corps (lignes *4* et *5*) figurent les instructions qui spécifient les traitements qui auront lieu lorsque la fonction sera exécutée.

> Le corps d'une fonction constitue un cas particulier de bloc d'instructions : une liste d'instructions délimitée par un couple d'accolades.

Les fonctions s'appellent les unes les autres, sauf `main()`

L'intérêt d'une fonction réside dans le traitement que décrivent les instructions contenues dans son corps. Pour que ce traitement soit effectué, il faut que la fonction soit appelée, c'est à dire qu'une instruction mentionnant le nom de la fonction soit exécutée.

Où peut donc bien figurer une telle instruction ?

Aucun suspens : dans le corps d'une autre fonction (c'est le seul endroit où des instructions exécutables ont le droit de figurer).

Si l'exécution d'une fonction ne peut être déclenchée que par une autre fonction elle-même en cours d'exécution, qui commence ?

Par définition, le lancement d'un programme se traduit par l'exécution des instructions contenues dans la fonction `main()`, qui jouit donc d'un statut exceptionnel : elle n'a pas besoin d'être appelée par une de ses collègues pour se mettre au travail.

> Le statut de `main()` est même encore plus exceptionnel que ça : son appel explicite est *interdit*.

La présence d'une fonction `main()` est donc indispensable pour qu'un texte source puisse donner naissance à un programme. C'est la raison pour laquelle la création d'un projet par QtCreator s'accompagne de la rédaction automatique d'une fonction `main()` minimale.

Types et variables

Les traitements effectués par un programme s'appliquent à des données représentées dans la mémoire de l'ordinateur. En C++, les notions de variable et de type permettent de définir les conventions de codage utilisées pour représenter ces données, ainsi que les opérations élémentaires qui peuvent leur être appliquées.

On peut voir les variables comme des sortes de boîtes : elles portent un nom (l'équivalent d'une étiquette collée sur la boîte et permettant de la désigner sans ambiguïté), ont un contenu (ce qu'on appellera la valeur de la variable) mais ne peuvent pas contenir n'importe quoi (la nature de ce qu'une variable peut contenir dépend de son type).

> Si une boîte est de type "étui à violon", il sera impossible d'y ranger un saxophone ou une contrebasse, sans parler d'un porte-avions.

Notre exploration du langage va commencer par l'usage de variables de différents types, et nous devons d'ores et déjà nous faire à l'idée que ces types se répartissent en trois familles :

- Les types standard sont définis une fois pour toutes par la norme internationale régissant le langage C++ (ISO/IEC 14882:1998). Ces types sont disponibles quel que soit le contexte et permettent de créer des variables dans le corps de n'importe quelle fonction. Ils sont simples et peu nombreux (trois, fondamentalement, un peu plus si on considère toutes les variantes offertes).

- D'autres types ont été créés par des programmeurs utilisant le langage C++ et sont mis à votre disposition par l'intermédiaire de librairies. On peut se représenter une librairie comme une collection de fragments de programmes pré-fabriqués, qui vont nous permettre de monter nos programmes beaucoup plus rapidement et facilement que s'il nous fallait utiliser uniquement les briques élémentaires proposées par le langage lui-même. Pour utiliser un type défini dans une librairie, il faut réunir deux conditions : la librairie en question doit être disponible (sous la forme d'un fichier .lib) au moment où le programme est généré **ET** le fichier dans lequel le type va être utilisé doit déclarer cette intention (en général au moyen de directives d'inclusion analogues à celle figurant sur la ligne *1*). Ces types peuvent être très élaborés et se compter par milliers. L'usage d'une librairie implique donc une consultation fréquente de sa documentation.

- La troisième famille regroupe les types que vous définirez vous-même au cours de l'écriture de votre programme. Leur nombre et leur niveau d'élaboration ne dépend donc que de vous, mais leur utilisation exige évidemment qu'ils soient parfaitement définis au moment où vous demanderez à QtCreator de créer un programme à partir de votre texte source.

Le corps d'une fonction peut contenir des définitions de variables

La ligne *5* illustre la façon dont une variable est créée : le corps d'une fonction contient une ligne indiquant un type, un nom et une valeur d'initialisation.

```
QCoreApplication a(argc, argv);
```

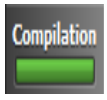
Il s'agit ici d'un type défini dans la librairie Qt (ce qui exige la directive d'inclusion de la ligne *1*) et la variable est baptisée `a`. La complexité du type `QCoreApplication` ne nous permet pas d'entrer pour l'instant dans les détails de l'initialisation de la variable `a` et de l'usage qui en est fait à la ligne *7*.

Avant d'explorer plus avant les joies de la création de variables de types plus simples, voyons un peu à quoi ressemble le programme dans son état initial.

3 - Une console comme interface utilisateur

Pour demander à QtCreator de créer un programme à partir des fichiers de notre projet, cliquez sur le bouton représentant un marteau qui est situé tout en bas du bandeau gauche  .

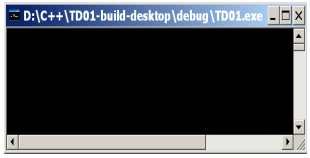
Comme nous n'avons pas encore touché au contenu du fichier main.cpp, il devrait se prêter de bonne grâce aux traitements qui l'attendent et dont le succès va se traduire par le passage du témoin de construction à l'état "construction réussie" (cf. ci-contre).



> Selon la puissance de l'ordinateur que vous utilisez et la complexité du projet concerné, les traitements permettant la création du programme peuvent durer plus ou moins longtemps. Dans le cas présent, une durée de 4 à 5 secondes peut être considérée comme normale.

Une fois la construction réussie, vous pouvez lancer l'exécution du programme en cliquant sur la flèche verte située immédiatement au dessous du témoin de construction ☐.

L'interface utilisateur de notre programme est une fenêtre classique (avec sa barre de titre, sa case de fermeture ou tous les dispositifs normaux proposés par votre système) qui simule un environnement de type "console texte", c'est à dire un dispositif où la communication entre programme et utilisateur se résume à un échange de messages écrits.



Dans ce type d'interface, le programme affiche un message (typiquement, une question) et attend que l'utilisateur saisisse un texte au clavier et achève sa saisie en pressant la touche [Entrée]. Le programme cherche alors à exploiter le texte saisi par l'utilisateur, puis pose éventuellement une nouvelle question, et ainsi de suite.

> Ce type d'interaction était le seul connu avant l'apparition des interfaces dites "graphiques", qui se caractérisent par l'utilisation de la souris et de dispositifs logiciels tels que les icônes, les menus et les différents types de boutons que vous connaissez. Du point de vue de l'écriture des programmes, ces deux contextes diffèrent radicalement. Dans un univers "console", c'est le programme qui décide du déroulement des opérations : il pose ses questions les unes après les autres, dans un ordre choisi par le programmeur, et sera incapable d'exploiter une saisie qui ne correspondrait pas à ses attentes. Dans une interface graphique, au contraire, c'est l'utilisateur qui a l'initiative et le programme doit être conçu pour réagir correctement à des sollicitations qui lui sont adressées dans un ordre imprévisible.

Dans son état actuel, le programme n'affiche rien et ne se prête à aucune saisie de la part de son utilisateur. La seule chose que vous puissiez en faire, c'est cliquer sur la case de fermeture de la fenêtre, ce qui met fin à l'exécution du programme ☐.

4 - Utilisation de la console

L'affichage et la saisie de texte dans une console ne sont pas des opérations supportées directement par le langage C++.

> Ce langage n'est pas spécialement destiné à créer des programmes évoluant dans ce contexte. La plupart des programmes que vous allez écrire utiliseront une interface graphique, et il faut aussi signaler que de nombreux programmes écrits en C++ sont destinés à des machines dépourvues du couple écran/clavier qui équipe les micro-ordinateurs actuels.

Affichage de texte

Pour utiliser la console, nous allons donc avoir recours à la librairie standard, que QtCreator met à la disposition de nos programmes sans que nous ayons autre chose à faire que d'annoncer notre intention d'utiliser certaines de ses fonctionnalités :

```
1 #include <QtCore/QCoreApplication>
2 #include <iostream>
3 int main(int argc, char *argv[])
4 {
5 QCoreApplication a(argc, argv);
6 std::cout << "Bonjour !";
7 return a.exec();
8 }
```

Cette précaution oratoire étant prise (ligne 2), nous sommes en mesure de demander (ligne 6) à notre fonction d'afficher un message dans sa console. Trois points sont à souligner :

- Le texte que nous voulons voir apparaître est placé entre guillemets.

- La demande d'affichage est symbolisée par deux chevrons qui forment une sorte de flèche.

- Cette flèche envoie le texte en direction de la console, désignée par une appellation qui peut sembler un peu barbare mais qui obéit à une certaine logique : `std::` rappelle qu'il s'agit d'un dispositif relevant de la librairie **st**andar**d**, le `c` est simplement l'initiale du mot "console" et le `out` indique que la console est ici utilisée comme une sortie (le message va du programme vers l'utilisateur, c'est à dire de l'intérieur de la machine vers l'extérieur).

Ajoutez les lignes 2 et 6 à votre fichier main.cpp ☐.

En C++, les instructions exécutables se concluent par un point virgule.

> Attention aux fautes de frappe. Les deux chevrons constituent un mot du langage C++, il faut donc éviter de briser ce mot en laissant un espace entre eux...

Reconstruisez, puis exécutez votre programme ☐.

Bravo ! Comme des milliers d'apprentis programmeurs avant vous, vous venez de faire vos premiers pas en C++ sous la forme d'un "Hello, world!" absolument réglementaire (bien que francophone).

Saisie de texte

Récupérer un texte saisi par l'utilisateur est une opération un peu plus compliquée. On peut deviner que la console va être désignée par `std::cin` (puisque le message va maintenant de l'extérieur vers l'intérieur de la machine) et qu'une flèche composée de deux chevrons va diriger le texte provenant de la console vers...

Vers quoi, justement ?

Pour que le programme soit en mesure d'utiliser l'information par la suite, il FAUT que celle-ci soit stockée en mémoire. Il nous faut donc créer une variable, c'est à dire une boîte susceptible de contenir le nom que l'utilisateur va nous donner. La librairie standard offre justement, sous le nom de `std::string`, un type qui convient parfaitement pour stocker des chaînes de caractères. Comme la variable doit exister avant que la saisie ne s'effectue, nous écrirons donc :

```
1  int main(int argc, char *argv[])
2  {
3  QCoreApplication a(argc, argv);
4  std::cout << "Comment vous appelez-vous ? ";
5  std::string nomUtilisateur;
6  std::cin >> nomUtilisateur;
```

L'exécution de la ligne *5* crée une variable nommée `nomUtilisateur`, de type `std::string`.
L'exécution de la ligne *6* range dans cette boîte la séquence de lettres que l'utilisateur a tapée. Ceci signifie que l'exécution de la ligne *6* peut durer très longtemps : tant que l'utilisateur n'a pas pressé la touche [Entrée], la saisie est en cours et le programme ne peut pas continuer.

Une fois la saisie terminée, le contenu de la boîte peut être utilisé, même si nous n'avons évidemment aucune idée du nom de l'utilisateur au moment où nous écrivons le programme :

```
7  std::cout << "Bonjour, ";
8  std::cout << nomUtilisateur;
9  std::cout << ". Je suis absolument ravi de faire votre connaissance !";
10 return a.exec();
11 }
```

Le phénomène intéressant est illustré sur la ligne *8* : ce n'est pas le nom de la variable qui est affiché sur l'écran, mais son contenu.

Lorsque le contexte s'y prête, le nom d'une variable désigne le contenu de celle-ci.

Donnez à votre fonction `main()` le contenu décrit ci-dessus ☐ et faites construire le programme ☐.
Vérifiez que l'exécution de celui-ci est conforme à vos attentes ☐.

Les caractères apparaissent à l'écran à mesure que vous les tapez pour indiquer votre nom. Il s'agit là d'un simple écho proposé par la console elle-même, pour vous permettre de savoir où vous en êtes et de corriger une éventuelle erreur. Le programme, lui, n'aura accès à votre saisie que lorsque vous finirez par presser la touche [Entrée].

5 - Affectation

Il arrive évidemment que le contenu d'une variable ne dépende pas (ou pas seulement) d'une saisie effectuée par l'utilisateur. Pour illustrer cette situation, nous allons introduire dans notre programme des variables d'un type standard : `int`. Ces boîtes peuvent contenir des valeurs numériques entières. Notre programme va simplement demander son age à l'utilisateur, puis utiliser cette information pour calculer en quelle année celui-ci aura 100 ans.

```
1  #include <QtCore/QCoreApplication>
2  #include <iostream>
3  #include <qdate>

4  int main(int argc, char *argv[])
5  {
6  QCoreApplication a(argc, argv);
7  std::cout << "Comment vous appelez-vous ? ";
8  std::string nomUtilisateur;
9  std::cin >> nomUtilisateur;
10 std::cout << "Bonjour, ";
11 std::cout << nomUtilisateur;
12 std::cout << ". Je suis absolument ravi de faire votre connaissance !";
13 std::cout << "\nQuel age avez-vous ? ";
14 int ageActuel(0);
15 std::cin >> ageActuel;
```

Les ligne *13* à *16* n'ont rien de bien nouveau. Remarquez simplement que la variable `ageActuel` est initialisée avec la valeur 0 (ligne *14*), une précaution judicieuse dans le cas des types standard.

> Les types fournis par les librairies sont généralement assez sophistiqués pour qu'une variable qui vient d'être créée n'ait pas un contenu fantaisiste. Dans le cas des types standard, mieux vaut prendre soin de les initialiser explicitement avec une valeur significative. Ici, un age nul signifie clairement que l'utilisateur n'a pas (encore ?) répondu.

```
16 int anneeActuelle(0);
17 anneeActuelle = QDate::currentDate().year();
```

Comme son nom l'indique, la variable `anneeActuelle` est destinée à contenir l'année courante. Plutôt que d'indiquer littéralement cette valeur dans le texte source (ce qui conduirait à un programme ayant une date limite d'utilisation), nous obtenons la valeur correcte grâce à la librairie Qt (en espérant, évidemment, que l'ordinateur sur lequel le programme sera exécuté aura bien une date courante exacte). Cette utilisation de la librairie Qt exige la présence de la directive d'inclusion de la ligne *3*.

En plus du recours à la librairie Qt, la ligne *17* fait intervenir une opération d'affectation : le signe = exige que la valeur fournie par Qt soit rangée dans la variable `anneeActuelle`.

Lorsqu'il figure à gauche de l'opérateur d'affectation (le signe =), le nom d'une variable désigne la variable elle-même, où il s'agit de ranger une nouvelle valeur (en oubliant l'ancienne).

La fin du programme ne devrait comporter aucune surprise pour vous, si ce n'est un petit raffinement à la ligne *20*, où deux choses sont affichées par une même instruction : un texte littéral (placé entre guillemets) puis, grâce à un second double chevrons, la valeur d'une variable.

```
18 int anneeCentenaire(0);
19 anneeCentenaire = anneeActuelle + 100 - ageActuel;
20 std::cout << "Vous serez donc centenaire en " << anneeCentenaire;
21 return a.exec();
22 }
```

> La ligne *19* procède, elle-aussi, à une affectation. La valeur qui doit être stockée dans la variable `anneeCentenaire` est obtenue par un calcul qui fait intervenir une addition (notée +) et une soustraction (notée -). Ce genre de calcul utilise évidemment les valeurs contenues dans les variables dont le nom est mentionné (`anneeActuelle` et `ageActuel`, dans cet exemple).

Si vous donnez à votre fichier main.cpp le contenu décrit ci-dessus, vous pourrez, une fois éliminées toutes les fautes de frappe, faire construire votre programme et jouer avec .

6 - Corriger les fautes de frappe

Comme tous les langages de programmation, C++ est peu enclin à pardonner les approximations. Une simple erreur de ponctuation, un caractère omis ou une majuscule qui devient minuscule suffisent à déclencher une avalanche de protestations de la part de QtDesigner.

Même le plus maniaque des débutants passe son temps à faire des fautes de ce genre, qui sont une source importante de frustration.

Il n'y a pas de solution miracle : c'est seulement avec l'expérience que vient le privilège d'éviter le plus souvent (et de corriger très facilement) ces petites imperfections. Vient alors le plaisir de commettre des erreurs bien plus graves (et plus difficiles à corriger), mais ceci est une autre histoire...

En attendant, le seul remède vraiment efficace si vous n'arrivez pas à trouver ce qui empêche la création de votre programme, c'est de montrer votre texte source à quelqu'un qui a des yeux un peu plus entraînés : le forum est là pour ça !

7 - Exercice

Créez un programme affichant dans une console le message "Nous sommes en 2011".

> Cet affichage devra être effectué sans que l'utilisateur ait à effectuer une quelconque saisie.
>
> Il va sans dire que, sans qu'il soit nécessaire de modifier le texte source et de reconstruire votre programme, celui-ci devrait être capable d'afficher "Nous sommes en 2012" s'il est exécuté l'année prochaine, puis "Nous sommes en 2013" s'il est exécuté l'année d'après, et ainsi de suite.