C/C++/ Ch 1 – Introduction

I. Introduction aux langages

Rappel : l'activité de programmation consiste à exprimer la résolution d'un problème à l'aide des instructions de base d'un langage de programmation. Avant de programmer, les étapes de résolution du problème ont donc dû être matérialisées en français ou dans le langage simplifié de l'algorithmique.

A. Notion de programme

Un **PROGRAMME SOURCE** est le résultat de traduction d'un algorithme (exprimé en pseudo-code, ou sous forme graphique) dans un langage de programmation. Il est stocké dans un fichier au format texte, l'extension du fichier dépendant du langage utilisé (en C : .c, en C++ : généralement .cpp).

Synonyme : code source

Un **PROGRAMME COMPILE** est le résultat de la traduction d'un programme source (exprimé dans un langage de programmation) en un langage exécutable par l'ordinateur, le langage machine. Il est stocké dans un fichier binaire, son extension est .exe (sous Windows).

Synonyme : exécutable

Une **APPLICATION INFORMATIQUE** est un programme ou un ensemble de programmes utilisables par l'utilisateur pour l'aider dans sa tâche.

Synonymes : logiciel

B. Langages de programmation

Les langages de programmation ont été conçus afin de répondre :

- A **l'évolution des technologies** liées à l'informatique :
  - au départ très proches de l'aspect technique des ordinateurs (0 et 1, le courant passe ou ne passe pas),
  - ensuite plus proche de l'expression en langage naturel, plus proche de la réflexion ;
- A **des besoins particuliers** (gestion, calcul scientifique, enseignement, systèmes d'exploitation, graphisme, intelligence artificielle, etc.)
- A **l'évolution dans la manière d'appréhender les problèmes** et de concevoir les programmes (on parle de « **paradigme** de programmation »).

C. Familles de langages de programmation

On peut classer les langages de programmation selon plusieurs critères pour constituer des familles de langages.
Voici quelques unes des caractéristiques fréquemment utilisées pour classer un langage :

- en fonction de sa proximité ou son éloignement du langage machine : générations ;
- en fonction de la manière de concevoir les programmes dans ce langage ;
- en fonction du mode d'exécution des programmes ;

Ces classements sont ne sont pas absolus : un langage pouvant appartenir à deux familles.

1. Générations de langages

Génération	**caractéristique**	**exemples**
L1G : Langages de 1ère génération	Correspond au langage machine : chaque instruction machine est codée sous forme binaire, une succession de 0 et de 1	C'est le langage de base des microprocesseurs composé d'un jeu d'instructions **Exemple (non vérifié) :** 10001010101111010101
L2G : Langages de 2nde génération	à chaque instruction machine, on associe un code mnémonique, plus simple à retenir qu'une suite de 0 et de 1 (allez trouver l'erreur dans une centaines de 0 et 1 !)	Assembleurs **Exemple (non vérifié) :** ADD 20, A LOAD A, #100
L3G : Langage de 3ème génération	La syntaxe des L3G se rapproche beaucoup plus du langage naturel (au moins du pseudo-code) On parle de langage de haut niveau	Cobol, C, Pascal, C++, Java, C#, Visual Basic, Ada, etc. **Exemple (non vérifié) :** While (nombre < 10) { ;}
L4G : Langages de 4ème génération	La syntaxe est de plus haut niveau encore : quelques instructions suffisent à construire des programmes complets de manière très rapide On parle de RAD pour Rapid Application Development.	W-langage, le langage de Windev (ICL AM : L4G des anciens mainframes ICL) **Exemple (non vérifié) :** FichierVersEcran() ;

2. Paradigmes de programmation[1]

Manière concevoir la programmation	caractéristique	Exemples de langages
PROGRAMMATION IMPERATIVE	des instructions manipulent des données constituant l'état du programme (effets de bord)	C, VB, PHP, C++, etc.
Programmation procédurale	*On conçoit un programme comme un ensemble de procédures et fonctions qui vont modifier les variables d'état du programme.*	*C, Cobol, Pascal, Basic Fortran* ***PHP*** ***C++**, Cobol, Pascal, Java, C#, Visual Basic Bash*
Programmation objets	*On associe des variables, des procédures et des fonctions pour former une classe d'objets ; un programme sera constitué d'un ensemble de ces classes.* *A l'exécution, le programme créé des objets (instanciation) qui s'envoient des messages (appels de méthodes d'instances)*	*Eiffel, Smalltalk* ***Java*** *Avec possibilité de créer des classes d'objets :* ***C++**, Cobol Objet,Pascal Objet (Delphi), C#, VB.net, **PHP version 5***
PROGRAMMATION FONCTIONNELLE	Des fonctions manipulent des ensembles de données constitués par d'autres fonctions (pas d'effet de bord)	Scheme, Lisp XSLT
PROGRAMMATION LOGIQUE	Un programme est une suite de déclaration de règles et un « moteur d'inférence » va déduire des conclusions à partir de ces règles	Prolog

3. Modes d'exécution des programmes

Mode exéc.	caractéristique	exemples
interprétés	Les instructions du programme source sont exécutées par un programme interpréteur : celui-ci lit chaque instruction et la traduit en instruction machine (au fur et à mesure) **L'exécution est lente**	**Logo** **PHP** Bash **Javascript**, VBscript
compilés	Le programme source est compilé **L'exécution est très rapide**	C, **C++**, Pascal
semi-compilés	Le source est traduit en un code semi-compilé (on parle de ***bytecode*** - non directement exécutable par la machine). Un programme particulier, appelé « machine virtuelle » (*en anglais : **VM**, Virtual Machine*) se charge de convertir le bytecode en langage machine au moment de l'exécution. Un « ramasse-miettes » (*en anglais garbage collector*) nettoie la mémoire **L'exécution est rapide**	**Java** C#, VB.net, etc.

[1] Paradigmes de programmation : manière « de voir les choses » en programmation

.

D. Programmes et types d'interactions

Interactions avec l'utilisateur :

Programmes	caractéristique	exemples
En mode Console, en traitement par lots (Batch)	Un programme exécute un traitement avec très peu d'interactions avec l'utilisateur ou sans interaction pour le batch. A chaque moment, on connaît la prochaine instruction qui sera exécutée	Apprentissage, Calculs longs et complexes
Evènementielles	L'exécution du programme est pilotée par les évènements déclenchés par les interactions de l'utilisateur avec l'interface graphique. La prochaine instruction dépendra de l'action de l'utilisateur (click sur tel ou tel bouton, etc., = non prévisible)	Suites bureautiques, Logiciels de gestion, de dessin, etc. = les applications actuelles

Interactions entre programmes :

Client serveur	applications faisant appel à des services distants	navigateur et serveur Web, client FTP et serveur FTP, application et serveur de données (SGBD)

.Les applications en mode console et évènementielles peuvent également être du type client–serveur, c'est-à-dire faire appel à des services distants.

E. Critères qualitatifs des logiciels

Un certain nombre de règles (*anglais : rules*) d'écriture des programmes vont permettre de construire des applications de qualité.
Quelques uns des aspects qualitatifs d'une application :

- La **maintenabilité** (anglais maintainability) : qualité d'un programme source à être modifié ; qualité du code source à être lisible – lisibilité (anglais : readability) (=utiliser des commentaires, des noms de variables clairs, créer des modules indépendants, etc.)
- La **fiabilité** (anglais : reliability) : comportement prévisible (= répondre exactement aux besoins de l'utilisateur, prévoir les erreurs et les gérer, etc.) ;
- La **performance** (anglais performance) (=optimiser les calculs, etc.)
- La **sécurité** (anglais : security) : en conflit avec la performance dans la mesure où elle va nécessiter l'ajout de code source supplémentaire, cette qualité est néanmoins fondamentale (=ajouter du code de gestion des erreurs de saisie, etc.);
- La **portabilité** (anglais : portability) (=écrire sans utiliser des particularités spécifiques à l'ordinateur utilisé, etc.)
- la **disponibilité** (anglais : availability)

L'écriture du code source devra tenir compte de ces règles afin de construire des applications de qualité.
Cf.
https://www.securecoding.cert.org/confluence/display/seccode/CERT+Secure+Coding+Standards

F. Etapes de programmation et outils associés

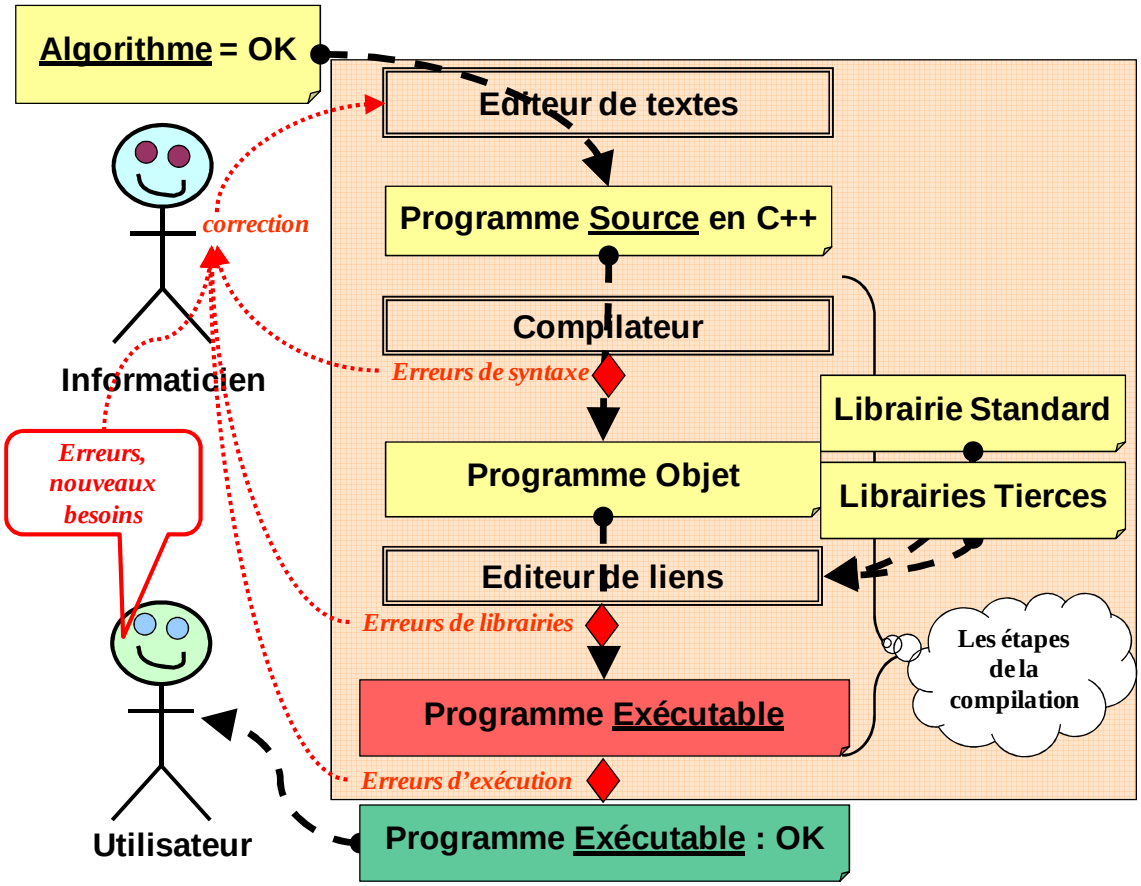

Figure 1 : les étapes de construction d'une programme informatique

1. Editeur de texte

Un éditeur de texte est un logiciel permettant d'écrire un fichier texte (suite de caractères sans mise en forme, non formaté).
Un bon éditeur de texte, pour la programmation, doit offrir :

- la **coloration syntaxique** : coloration des mots-clefs du langage de programmation ;
- la **complétion de code** : proposition de syntaxe des instructions en fonction du contexte de saisie ;

Exemples :

- Pspad sous Windows, Gedit sous Linux, Scite, Notepad++
- emacs, Vim (Linux) et Gvim (Windows)

2. Compilateur du langage vers le langage machine

Le compilateur traduit le langage source, exprimé dans un langage de programmation, en langage machine.
En fait, la « compilation » est un processus plus complexe qu'une simple traduction, et se déroule en 3 phases essentielles, réalisées par les programmes suivants :

- Le pré-processeur : analyse des directives de compilation (ex : #include)
- Le compilateur: il assure la traduction du code source en code machine (analyse la syntaxe du programme source et produit un programme compilé, mais non exécutable)
- L'éditeur de liens : met en relation le programme avec les librairies utilisées.

Cf. programmez121.pdf – page 76 et suivantes

3. EDI, anglais IDE

L'**EDI** (Environnement de Développement Intégré, en anglais : **IDE**, Integrated Development Environment) est un outil intégrant toutes les fonctions permettant l'écriture des sources, la compilation et le débogage[2].
Il inclut également des outils permettant la construction d'interfaces graphiques et de nombreuses bibliothèques de procédures et fonctions permettant de produire rapidement des applications.
Exemples :

- Pour C/C++ : Dev C++, Anjuta, Code::Blocks
- Pour Ada : Adagide
- Pour Java : NetBeans
- Pour Pascal Objet : Borland Delphi
- Pour VB.net et C# : Microsoft Visual Studio
- Paramétrable (plugins) : Eclipse (Open Source), IBM WSAD (commercial)

4. AGL

L'**AGL** (Atelier de Génie Logiciel) intègre toutes les phases de développement des applications à l'échelle de l'entreprise :

- Analyse et modélisation, conception des bases de données
- Un environnement de développement intégré permettant le travail en équipes de programmeur
- Des outils de tests et de distribution de logiciels.

Il utilise souvent un langage propriétaire (spécifique).
Exemple :

- Windev

II. Les langages C et C++,

A. Eléments généraux

Le langage C a été conçu par deux chercheurs des laboratoires Bell (Brian Kernighan et Dennis Ritchie – K & R) dans les années 1970 (livre de référence « The C programming Language », 1978).
Il est normalisé au début des années 1990 : il devient ISO C 90 (appelé aussi ANSI C90), puis des améliorations sont inclues dans la version ISO C 99 (afin d'assurer, autant que possible, la portabilité de C vers C++)

Le C est un langage de haut niveau intégrant des types de données de base, des structures de contrôles et des structures de données structurées. Des bibliothèques standard ajoutent les mécanismes permettant les appels aux fonctions du système d'exploitation : entrées, sorties, gestion de la mémoire, etc.

Le langage C++ (C++ 98) est une extension du langage C (Bjarne Stroustrup) afin de prendre en compte les évolutions en terme de « paradigmes de programmation[3] » : le paradigme objet est en effet aujourd'hui la référence en conception d'applications. Le

[2] Debogage : recherche et correction des défauts de conception du programme conduisant à des disfonctionnements ou bogues (en anglais : bugs)

langage C++ inclut donc de nouvelles bibliothèques de classes (non compatibles avec le C).

La portabilité du langage C/C++ (« pouvoir compiler un source aussi bien sous Linux que sous Windows », par exemple), est donc toute relative, de nombreux compilateurs incorporant en effet des bibliothèques spécifiques. Il faut donc s'attacher à n'utiliser que les éléments définis dans les standards ou des bibliothèques tierces portables.
Le langage C/C++ est le langage d'écriture des systèmes d'exploitation.

B. Exemple « helloworld »

1. En C

```
(1)  /* hello.c */
(2)  #include <stdio.h>
(3)  int main()
(4)  {
(5)       printf("hello World !") ; // afficher hello World
(6)       return 0 ;
(7)  }
```

2. En C++

```
(1)  /* hello.cpp */
(2)  #include <iostream>
(3)  using namespace std ;
(4)  int main()
(5)  {
(6)       cout << "hello World !" ; // afficher hello World
(7)       return 0 ;
(8)  }
```

[3] Paradigme de programmation : manière « de voir les choses » en programmation