Chapitre 1

Introduction

Langages informatiques

- Un langage informatique est un outil permettant de donner des ordres (**instructions**) à la machine
  - A chaque instruction correspond une action du processeur
- Intérêt : écrire des **programmes** (suite consécutive d'instructions) déstinés à effectuer une tache donnée
  - Exemple: un programme de gestion de comptes bancaires
- Contrainte: être compréhensible par la machine

Langage machine

- Langage **binaire**: l'information est exprimée et manipulée sous forme d'une suite de bits
- Un **bit** (*binary digit*) = 0 ou 1 (2 états électriques)
- Une combinaison de 8 bits= 1 **Octet** ➔ $2^8 = 256$ possibilités qui permettent de coder tous les caractères alphabétiques, numériques, et symboles tels que ?,*,&, …
  - Le code **ASCII** (*American Standard Code for Information Interchange*) donne les correspondances entre les caractères alphanumériques et leurs représentation binaire, Ex. A= 01000001, ?=00111111
- Les opérations logiques et arithmétiques de base (addition, multiplication, …) sont effectuées en binaire

L'assembleur

- Problème: le langage machine est difficile à comprendre par l'humain

- Idée: trouver un langage compréhensible par l'homme qui sera ensuite converti en langage machine
  - **Assembleur** : exprimer les instructions élémentaires de façon symbolique

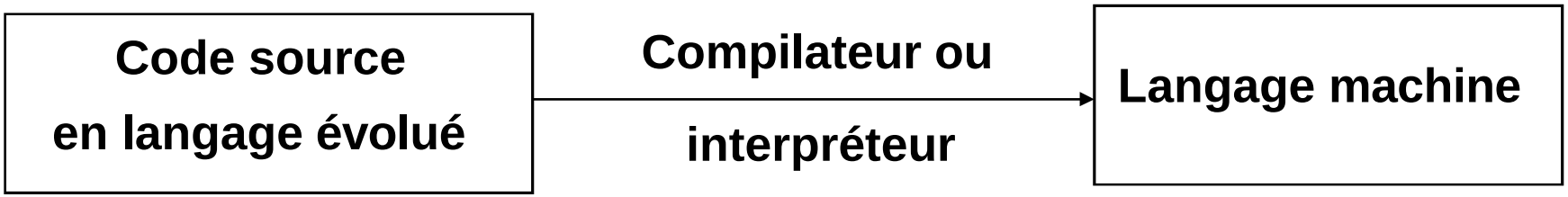
ADD A, 4
LOAD B —traducteur→ langage machine
MOV A, OUT
...

  - +: déjà plus accessible que le langage machine
  - -: dépend du type de la machine (n'est pas ***portable***)
  - -: pas assez efficace pour développer des applications complexes

⟹ **Apparition des langages évolués**

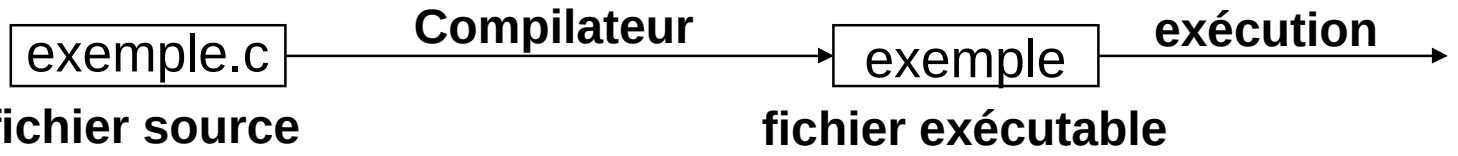
Langages haut niveau

- Intérêts multiples pour le haut niveau:
  - proche du langage humain «anglais» (compréhensible)
  - permet une plus grande portabilité (indépendant du matériel)
  - Manipulation de données et d'expressions complexes (réels, objets, a*b/c, …)
- Nécessité d'un traducteur (compilateur/interpréteur), exécution plus ou moins lente selon le traducteur



Compilateur/interpréteur

- Compilateur: traduire le programme entier une fois pour toutes



  - + plus rapide à l'exécution
  - + sécurité du code source
  - - il faut recompiler à chaque modification

- Interpréteur: traduire au fur et à mesure les instructions du programme à chaque exécution

  - + exécution instantanée appréciable pour les débutants
  - - exécution lente par rapport à la compilation

Langages de programmation:

- Deux types de langages:
  - Langages procéduraux
  - Langages orientés objets

- Exemples de langages:
  - **Fortran, Cobol, Pascal, C, …**
  - **C++, Java, …**

Historique du C

- Le langage C a été conçu en 1972 dans «Bell Laboratories » par *Dennis Ritchie* avec l'objectif d'écrire un système d'exploitation (UNIX).

- En 1978, une première définition rigoureuse du langage C (*standard K&R-C*) a été réalisée par *Kernighan et Ritchie* en publiant le livre «The C Programming Language ».

- Le succès du C et l'apparition de compilateurs avec des extensions particulières ont conduit à sa normalisation.

- En 1983, l'organisme ANSI (American National Standards Institute) chargeait une commission de mettre au point une définition explicite et portable pour le langage C. Le résultat est le *standard ANSI-C*.

Caractéristiques du C

- Universel : n'est pas orienté vers un domaine d'application particulier (applications scientifiques, de gestion, …)
- Près de la machine : offre des opérateurs qui sont proches de ceux du langage machine (manipulations de bits, d'adresses, …) ➔ efficace
- Modulaire: peut être découpé en modules qui peuvent être compilés séparément
- Portable: en respectant le standard ANSI-C, il est possible d'utiliser le même programme sur plusieurs systèmes (hardware, système d'exploitation)

Remarque : Une programmation efficace et compréhensible en C n'est pas facilement accessible à des débutants

Programme source, objet et exécutable

- Un programme écrit en langage C forme un texte qu'on nomme *programme ou code source*, qui peut être formé de plusieurs fichiers sources

- Chaque fichier source est traduit par le compilateur pour obtenir un *fichier ou module objet* (formé d'instructions machine)

- Ce fichier objet n'est pas exécutable tel quel car il lui manque les instructions exécutables des fonctions standards appelées dans le fichier source (printf, scanf, …) et éventuellement d'autres fichiers objets

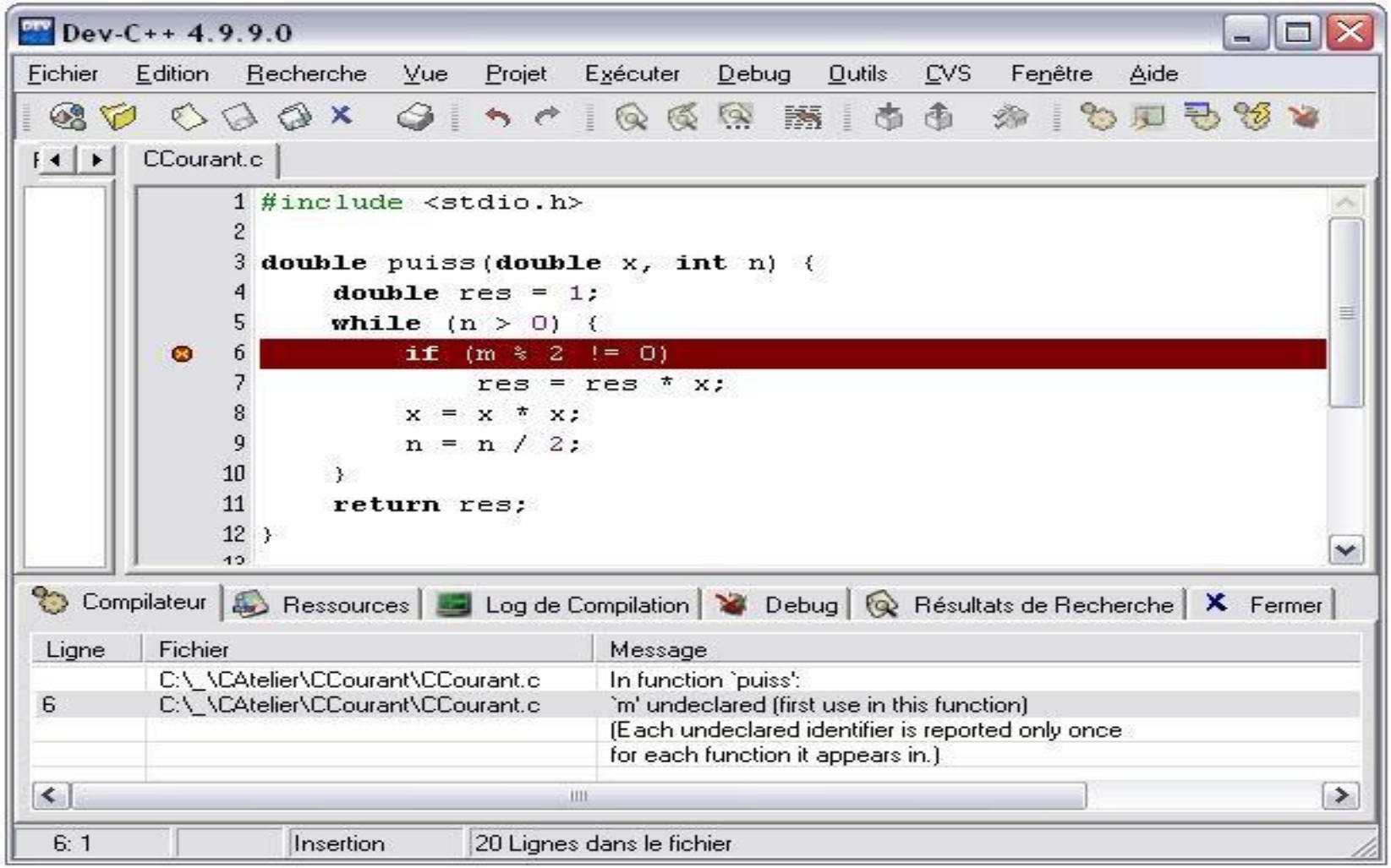
- *L'éditeur de liens* réunit les différents modules objets et les fonctions de la bibliothèque standard afin de former *un programme exécutable*

Remarque : la compilation est précédée par une phase de prétraitement (inclusion de fichiers en-tête) réalisé par le *préprocesseur*

Compilateurs C

- Pour pouvoir écrire des programmes en C, vous avez besoin d'un compilateur C sur votre machine
- Il existe plusieurs compilateurs respectant le standard ANSI-C. Une bonne liste est disponible sur : c.developpez.com/compilateurs/
- Nous allons utiliser l'environnement de développement Dev-C++ avec le système d'exploitation Windows
- Vous pouvez télécharger Dev-C++ librement, par exemple sur le site www.bloodshed.net

Exemple d'une fenêtre Dev-C++



Composantes d'un programme C

- Directives du préprocesseur
  - inclusion des fichiers d'en-tête (fichiers avec extension .h)
  - définitions des constantes avec **#define**
- déclaration des variables globales
- définition des fonctions (En C, le programme principal et les sous-programmes sont définis comme fonctions)
- Les commentaires : texte ignoré par le compilateur, destiné à améliorer la compréhension du code

exemple :

```
#include<stdio.h>
main()
  {
    printf( "notre premier programme C \n");
   /*ceci est un commentaire*/
  }
```

Remarques sur ce premier programme

- *#include<stdio.h>* informe le compilateur d'inclure le fichier stdio.h qui contient les fonctions d'entrées-sorties dont la fonction printf
- La fonction **main** est la fonction principale des programmes en C: Elle se trouve obligatoirement dans tous les programmes. L'exécution d'un programme entraîne automatiquement l'appel de la fonction **main**.
- L'appel de printf avec l'argument "notre premier programme C\n" permet d'afficher : notre premier programme C et \n ordonne le passage à la ligne suivante
- En C, *toute* instruction simple est terminée par un point-virgule **;**
- Un commentaire en C est compris entre // et la fin de la ligne ou bien entre /* et */