

BASES DE LA PROGRAMMATION

INTRODUCTION

ORDINATEUR

- Ordinateur = machine capable de traiter des données
- Les ordinateurs ne sont pas intelligents
 - ils sont même particulièrement idiots
 - il faut leur dire *exactement* ce qu'ils doivent faire

PROGRAMME

- **Programme** = ensemble d'instructions pouvant être comprises par un ordinateur pour effectuer une tâche

LANGAGE DE PROGRAMMATION

- **Langage de programmation** = langage utilisé pour communiquer avec la machine sous forme de programme
- Un tel langage est construit de telle sorte qu'il soit :
 - suffisamment compréhensible pour qu'un humain puisse concevoir des programmes
 - formel et précis pour que la machine puisse le traduire en programme qu'elle est capable d'exécuter

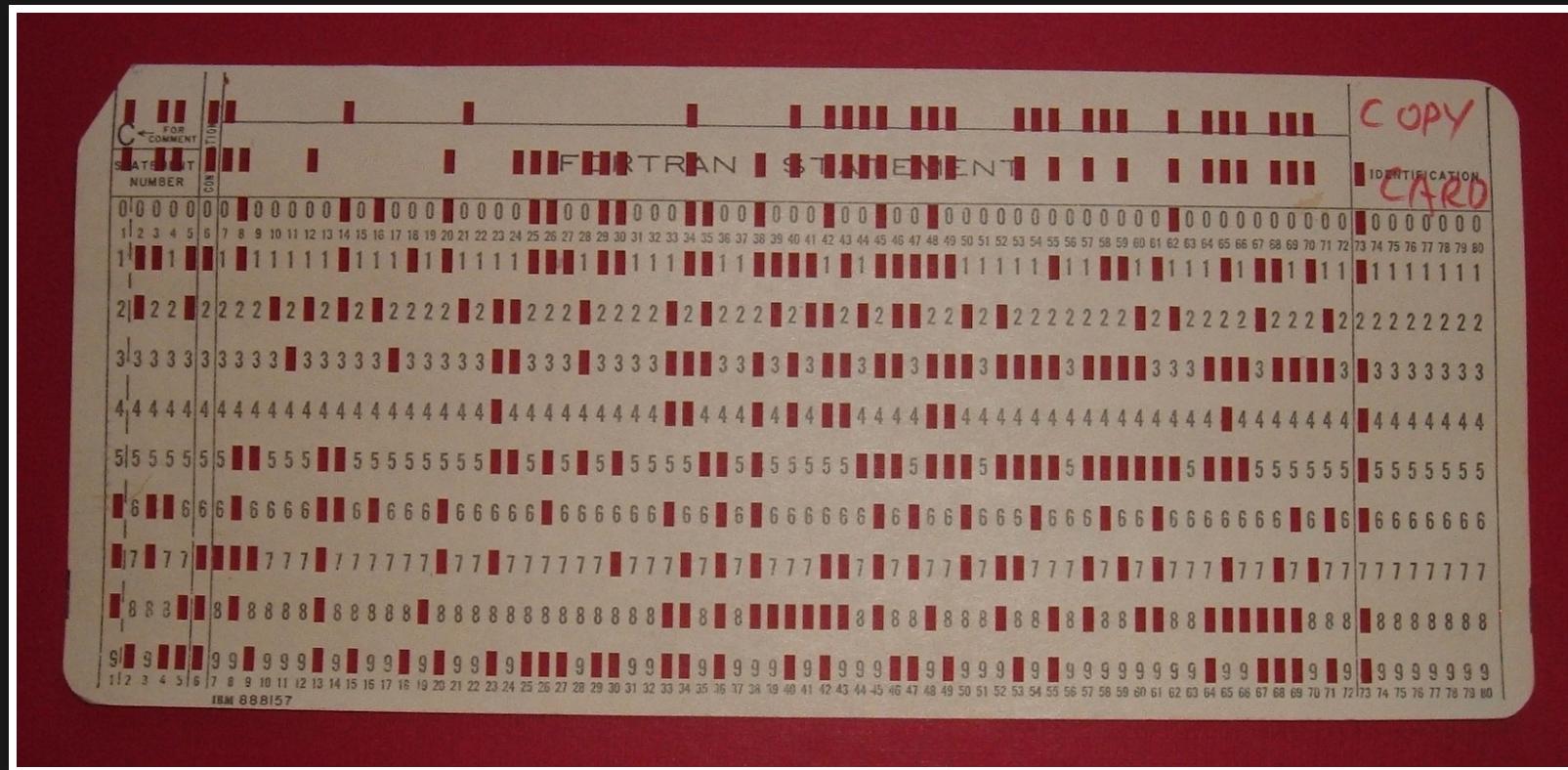
CODE SOURCE

- **Code source** = représentation textuelle du programme dans le langage de programmation considéré
 - vous écrirez vous-mêmes le code source de vos programmes
- On parle aussi des **sources** d'un programme : c'est la même chose

UN PROGRAMME, POUR LA MACHINE

```
01010000 01110010 01101111 01100111 01110010 01100001 01101101 01110011  
00100000 01110111 01100101 01110010 01100101 00100000 01101111 01110010  
01101001 01100111 01101001 01101110 01100001 01101100 01101100 01111001  
00100000 01110111 01110010 01101001 01110100 01110100 01100101 01101110  
00100000 01101001 01101110 00100000 01101101 01100001 01100011 01101000  
01101001 01101110 01100101 00100000 01101100 01100001 01101110 01100111  
01110101 01100001 01100111 01100101 00101100 00100000 01100001 00100000  
01110011 01100101 01110010 01101001 01100101 01110011 00100000 01101111  
01100110 00100000 00110000 00100000 00100110 00100000 00110001 00101100  
00100000 01100010 01100101 01100011 01100001 01110101 01110011 01100101  
00100000 01110100 01101000 01101001 01110011 00100000 01101001 01110011  
00100000 01110100 01101000 01100101 00100000 01101111 01101110 01101100  
01111001 00100000 01101100 01100001 01101110 01100111 01110101 01100001  
01100111 01100101 00100000 01100011 01101111 01101101 01110000 01110101  
01110100 01100101 01110010 01110011 00100000 01110101 01101110 01100100  
01100101 01110010 01110011 01110100 01100001 01101110 01100100 00101110
```

CARTES PERFORÉES (ANNÉES 1950)



SOFT
WATER
SKEL / SKELA
ERIK
PEHT
KÄRÖN
KÄRÖN

ID JOB (N6320,7255,2,2000,0,SRI=DEFER), "REINHOLD", MSGLEVEL=1

ASSEMBLEUR (ANNÉES 1960)

```
str:  
    .ascii "Bonjour\n"  
    .global _start  
  
_start:  
    movl $4, %eax  
    movl $1, %ebx  
    movl $str, %ecx  
    movl $8, %edx  
    int $0x80  
    movl $1, %eax  
    movl $0, %ebx  
    int $0x80  
  
;Compilation:  
;as code.s -o code.o  
;ld code.o -o code  
  
;Execution:  
;./code
```

LANGUAGE C (ANNÉES 1970)

```
#include <stdio.h>

void main() {
    printf("Bonjour!\n");
}
```

QUELQUES LANGAGES POPULAIRES

- C
- C++
- Java
- Kotlin
- C#
- Python
- Go
- JavaScript
- TypeScript
- PHP
- Swift
- Dart/Flutter
- Ruby
- Powershell
- Scratch
- Rust
- Ada
- Lua
- OCaml
- Clojure
- Elixir

ALGORITHME

- **Algorithme** = énoncé pas à pas des actions à effectuer pour remplir une tâche
- Un algorithme est comme un guide, une **recette** pour résoudre un problème
- Un **algorithme** pourra être **implémenté** en utilisant un **langage de programmation** sur un **ordinateur** ; le **programme** résultant pourra alors être **exécuté**

EXEMPLE DE PROBLÈME

CALCULER LE SALAIRE BRUT D'UN EMPLOYÉ PAYÉ À L'HEURE

ALGORITHME CORRESPONDANT CALCULER LE SALAIRE BRUT D'UN EMPLOYÉ PAYÉ À L'HEURE

1. Récupérer le nombre d'heures travaillées
2. Récupérer le taux horaire
3. Multiplier le nombre d'heures par le taux horaire
4. Afficher le résultat

LE LANGUAGE JAVA

- Java est l'un des langages les plus utilisés dans le monde
 - polyvalent
 - multiplateforme
 - nombreuses bibliothèques et composants
 - orienté objets (réutilisation, modularisation)
 - gestion automatique de la mémoire

IMPLEMENTATION

- Implémenter un programme, une fonctionnalité, une correction de bugs, etc., c'est écrire le code source qui correspond

IMPLEMENTATION EN JAVA DE L'ALGORITHME DU CALCUL DE SALAIRE BRUT

MOTS-CLÉS RÉSERVÉS DU LANGAGE

- Certains mots-clés ne doivent pas être utilisés autrement que ce pour quoi le langage les destine, par exemple : `package`, `public`, `class`, `static`, `void`...
 - selon le thème sélectionné, on les reconnaît à leur couleur distincte dans l'IDE
 - Java compte des dizaines de mots réservés
 - le compilateur nous dira de toute façon si l'on tente d'utiliser de manière inappropriée un mot réservé

COMMENTAIRES

- Un **commentaire** est un message que l'on met dans le code source et que le compilateur va ignorer
- Un commentaire est donc **uniquement destiné aux développeurs**

```
// ceci est un commentaire sur une ligne seule
```

```
int nb; // commentaire spécifié après du code Java, sur la même ligne
```

VARIABLE

- Une **variable** désigne un emplacement mémoire qui stocke des données
- On va mettre dans une variable toute donnée que l'on va vouloir « retenir » pour utilisation ultérieure
- Une variable a toujours :
 - **un nom**
 - **une valeur**
- En Java, qui est un langage **typé statiquement**, on doit aussi spécifier le **type** de la variable

LANGAGES STATIQUES VS DYNAMIQUES

- Dans un langage **typé dynamiquement**, le type des variables est déterminé à l'exécution, en fonction du contexte d'utilisation
 - ex : JavaScript, PHP, Python
- Dans un langage **typé statiquement**, le type des variables est spécifié à la déclaration, dans le code source
 - ex : Java, C#, TypeScript
- Les langages dynamiques sont plus flexibles tandis que les langages statiques permettent de détecter plus d'erreurs avant l'exécution

EXEMPLE DE DÉFINITION DE VARIABLE

```
int heuresTravailles = 120;
```

- Nom : heuresTravailles
- Valeur : 120
- Type : int (*integer*, un nombre entier)

QUELQUES TYPES JAVA

- Java offre 8 types dits primitifs ; parmi eux :
 - `int` : nombre entier (123, -2)
 - `double` : nombre flottant (à virgule) (1.23)
 - `boolean` : deux valeurs, `true` ou `false` (vrai ou faux)

CONCATÉNATION DE STRINGS

- Une **string** est une chaîne de caractères
 - « string » est le nom utilisé dans la majorité des langages
- Une string peut être écrite « en dur » dans le code en l'entourant de guillemets doubles : "Salaire brut : "
- On peut **concaténer** (« coller ») deux strings ensemble en les séparant par l'opérateur + : "Le salaire brut " + "est de" + " : "
- On peut indiquer des variables comme composantes de ces opérations de concaténation, même si ce ne sont pas des strings : Java va les remplacer pour nous : "Le salaire brut est de " + `salaireBrut`

NOMS DE VARIABLES DESCRIPTIFS

- Faites en sorte de nommer clairement chaque variable pour que le lecteur puisse comprendre immédiatement de quoi il s'agit
 - ht est plus court que heuresTravaillees
 - quand on travaille sur l'implémentation, on « est dedans », ht est très clair pour nous
 - mais un collègue va mettre un certain temps à deviner, et pourrait même comprendre de travers
 - *vous-même*, six mois plus tard, vous ne vous souviendrez même plus avoir écrit ce code
- Toujours prendre le temps de nommer correctement les choses

CONVENTIONS DE NOMMAGE

- Espaces interdits, - interdits
- On évite les _
- En Java, on colle tous les mots et on utilise la convention **camelCase** : je commence par une minuscule, puis majuscule pour chaque mot suivant:
 - pourcentage1
 - maVariableBienNommee
 - nbSandwiches
- Pour les choses interdites, le compilateur nous préviendra de toute façon

ENTRÉES UTILISATEUR

- Actuellement, notre programme code « en dur » les valeurs de heuresTravailées et tauxHoraires
 - il faut modifier le code source pour modifier le résultat...
- En pratique, ces valeurs pourraient provenir d'une base de données, ou bien de « champs texte » dans une interface graphique
- Ici, on va faire en sorte que l'utilisateur entre ces valeurs au clavier

UTILISER UN SCANNER

- Un Scanner va nous permettre de prendre des entrées, notamment au clavier
- Un Scanner est un **objet** en Java, il faut l'initialiser avec le mot-clé **new**, puis utiliser la **notation pointée** pour lui demander des choses
- **nextInt()** est une **méthode**, un « message » qu'on envoie à l'objet qui veut dire : « attend que l'utilisateur rentre un nombre entier »

```
Scanner clavier = new Scanner(System.in); // initialisation  
int leNombreChoisi = clavier.nextInt(); // notation pointée pour indiquer qu'on att
```

LE TYPE STRING

- Java dispose du type `String` pour stocker des chaînes de caractères
 - notez la majuscule : comme `Scanner`, c'est un type objet ; les types objets prennent toujours des majuscules, contrairement à `int`, `double`...

IMPÉMENTATION DU SALAIRE BRUT EN JAVA

```
public static void main(String[] args) {  
  
    // Récupérer le nombre d'heures travaillées  
  
    // Initialisation du Scanner pour le clavier  
    Scanner clavier = new Scanner(System.in);  
    // nextInt attend une entrée au clavier  
    System.out.print("Entrez le nombre d'heures travaillées : ");  
    int heuresTravaillees = clavier.nextInt();  
  
    // Récupérer le taux horaire  
    System.out.print("Entrez le taux horaire : ");  
    double tauxHoraire = clavier.nextDouble();  
  
    // On "ferme" la ressource clavier  
    clavier.close();  
  
    // Multiplier le nombre d'heures par le taux horaire  
    double salaireBrut = heuresTravaillees * tauxHoraire;
```

IMPLÉMENTATION - TEXTE À TROUS

- Écrire un programme qui permet d'afficher la phrase suivante une fois que l'utilisateur a entré des valeurs pour un **premier nom commun**, un **nombre**, un **deuxième nom commun** et un **adjectif** :

Il était une fois un NOM1 qui avait NOMBRE NOM2. Cela le rendait très ADJECTIF.

- Bien sûr, les mots ici en majuscules seront remplacés par les entrées utilisateur
- Le programme doit indiquer clairement ce que l'utilisateur est censé faire : ça s'appelle avoir une bonne **expérience utilisateur** (UX)

