Algorithmique & Programmation

Résolution de problèmes

De l'énoncé à l'algorithme

yann.secq@univ-lille.fr

ABIDI Sofiene, ALMEIDA COCO Amadeu, BONEVA Iovka, CASTILLON Antoine,
DELECROIX Fabien, LEPRETRE Éric, SANTANA MAIA Deise, SECQ Yann

Saison 2023-2024 / R1.01+R1.02 :
Initiation au développement

Quelques éléments (simples!) de méthode

- Bien comprendre l'énoncé
- Identifier les informations significatives
- Etablir le cœur de la résolution
- Exécution et traces d'exécutions
- La traduction vers le langage d'implémentation

Une démarche

- Lire et comprendre l'objectif à atteindre (ie. correspond à l'étude du cahier des charges)
- Déterminer les informations significatives
- Essayer d'atteindre l'objectif avec quelques exemples
- Déterminer un algorithme général de résolution du problème
- Vérifier la validité de l'algorithme en effectuant une trace d'exécution (au minimum !)
- Traduire l'algorithme dans le langage d'implémentation

1. Lire et comprendre l'objectif à atteindre

- Paraît évident, mais pas toujours aussi simple que cela y paraît …
- Relire plusieurs fois l'énoncé
- Identifier le but à atteindre
- Poser des questions si il y a des ambiguïtés
- Expliquer les choix d'interprétation

2. Déterminer les informations significatives

- Séparer le fond de la forme !
- Quelles sont les informations significatives ?
  - Quelle est la nature de ces informations ?
  - Quelles sont les informations qui évoluent au cours du temps ?
  - Quel(s) est (sont) le(s) résultat(s) à produire ?

RESPECTER L'ENONCE !!

3. Faire quelques exemples “à la main”

- Impossible de trouver l’algorithme d’un problème donné si vous ne l’avez pas résolu vous même !
- Si j’étais une machine …
  - Résoudre le problème sur papier avec un exemple concret :
    - Permet de vérifier que l’on a compris le problème à résoudre
    - Permet d’identifier les informations utilisées
    - Permet d’identifier les calculs à effectuer
    - Donne une première intuition de l’algorithme

4. Déterminer un algorithme général résolvant le problème

- Abstraire et généraliser la résolution effectuée sur papier (ou « à la main »).
  - Identifier les entrées et les sorties
  - Déterminer si il y a des constantes et quel est leur type
  - Déterminer les variables et leur type
  - Déterminer les différentes phases du calcul
  - Expliciter et détailler chacune de ces phases

5. Effectuer une trace d'exécution

- Sélectionner les informations à visualiser lorsqu'il y a beaucoup de variables
- Dérouler l'algorithme « pas à pas »
- Détecter les éventuelles erreurs de calculs
- Vérifier la concordance entre le résultat fourni et la spécification
- Tester les cas limites …

6. Traduire l'algorithme dans le langage d'implémentation

- Une fois l'algorithme validé, on passe à sa traduction
- Bien commenter le code source
- Etre attentif aux déclarations de variables et à leur type, ainsi qu'au **NOMMAGE PERTINENT**
- Une fois que le programme tourne, le tester avec différents jeux de valeurs en entrée

Valeur d'un polynôme en un point donné

- Soit le polynôme : $y = 3x^2+2x+1$ (**x et y réels**)
- On désire réaliser un programme calculant la valeur en un point de ce polynôme.

- Alors, que fait-on maintenant ? :)

Valeur d'un polynôme en un point

- Spécification du problème à résoudre :
- But: calculer la valeur en un point du polynôme `f(x) = 3x²+2x+1` (`x` étant réel).
- Entrée: un nombre réel (`x`) représentant le point dont on veut calculer l'image avec le polynôme (`f`)
- Sortie: la valeur du polynôme en ce point (`f(x)`)

Une solution possible

```
class CalculPolynome extends Program {
  void algorithm() {
    print("x = ");
    double x = readDouble();

    double f_x = 3*x*x + 2*x + 1;

    println("f(x)=3*x*x+2*x+1) );
    println("Pour x="+x+", f("+x+")="+f_x);
  }
```

Généraliser le problème

- Pourquoi ne pas traiter tous les polynômes de la forme: $f(x)=ax^2+bx+c$?
- Que faut-il changer dans l'algorithme ?
- Il est souvent intéressant de généraliser, car cela permet de réutiliser le même code en traitant plus de situations à l'aide du même algorithme !

Une solution plus générale

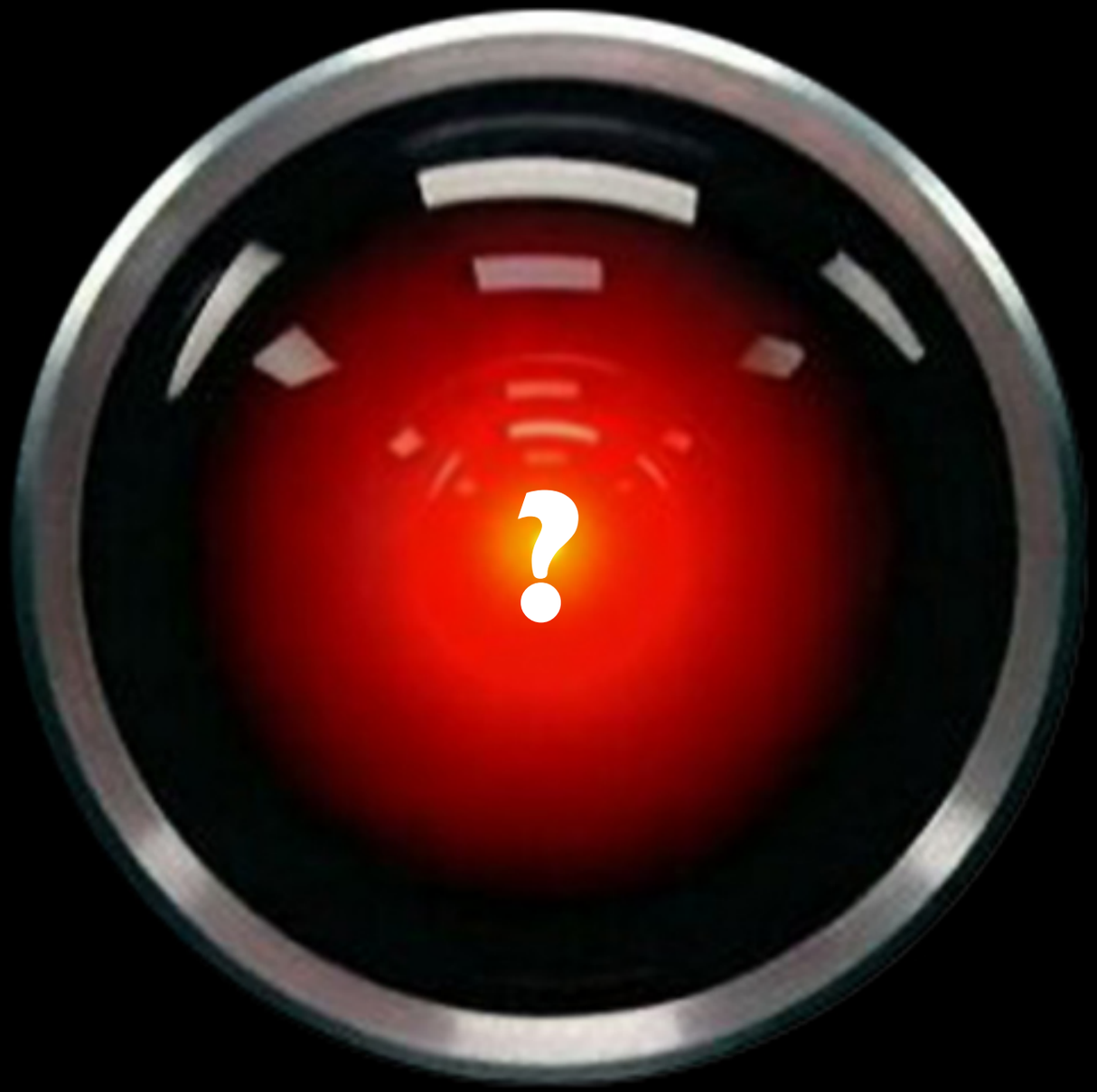
```
class CalculPolynome extends Program {
 void algorithm() {
    // définition du polynôme du second degré
    print("a = "); double a = readDouble();
    print("b = "); double b = readDouble();
    print("c = "); double c = readDouble();
    println(" f(x)="+a+"*x*x+"+b+"*x+"+c+")");

    print("x = "); double x = readDouble();
    // calcul de l'image de x par le polynôme
    double f_x = a*x*x + b*x + c;

    // Affichage du résultat de l'image de x par f
    println("Pour x="+x+", f("+x+")="+f_x);
}
```

Avant l'introduction de la notion de fonction …

Méthode à suivre !

- Analyser, décomposer, expérimenter, décrire, spécifier …
  - **et seulement à la fin : coder !**
- Lire et comprendre l'objectif à atteindre
- Déterminer les informations significatives
- Faire quelques exemples « à la main »
- Déterminer un algorithme général de résolution
- Effectuer des traces d'exécution de votre algorithme
- Traduire l'algorithme dans le langage d'implémentation
- N'oubliez pas de d'indenter et de commenter votre algorithme

MOVIEWALLPAPERS.NET



Rendre la monnaie

- Comment rendre la monnaie de la manière la plus efficace possible ?
- On dispose des coupures suivantes : 20, 10, 5, 2 et 1 €
- Entrée: un nombre représentant la somme en euros à rendre
- Sortie: le nombre minimum de billets de 20 euros, de 10 euros, de 5 euros et de pièces de 2 et de 1 euro à rendre

Version « explicite »

```
void algorithm()(
  int somme = readInt();
  int nb20, reste20, nb10, reste10, nb5,
  reste5, nb2, reste2, nb1, reste1;

  nb20    = somme   / 20;
  reste20 = somme   % 20;
  nb10    = reste20 / 10;
  reste10 = reste20 % 10;
  nb5     = reste10 / 5;
  reste5  = reste10 % 5;
  nb2     = reste5  / 2;
  reste2  = reste5  % 2;
  nb1     = reste2  / 1;
  reste1  = reste2  % 1;

  println("20 x "+ nb20 + "10 x " + nb10 + "5 x " + nb5 +
          "2 x " + nb2 + "1 x "  +nb1};
}
```

Version « un seul reste »

```
void algorithm()(
  int somme = readInt();
  int nb20, nb10, nb5, nb2, nb1, reste;

  nb20  = somme / 20;
  reste = somme % 20;
  nb10  = reste / 10;
  reste = reste % 10;
  nb5   = reste / 5;
  reste = reste % 5;
  nb2   = reste / 2;
  reste = reste % 2;
  nb1   = reste / 1;

  println("20 x "+ nb20 + "10 x " + nb10 + "5 x " + nb5 +
"2 x " + nb2 + "1 x "  +nb1};
}
```

Version « une seule variable, mais re-calculs »

```
void algorithm() {
    int somme = readInt();

    String piecesARendre =
       "20 x "+(somme / 20)+                          // nb billets 20€
       "10 x "+((somme % 20) / 10)+                   // nb billets 10€
       " 5 x "+(((somme % 20) % 10) / 5)+             // nb billets  5€
       " 2 x "+((((somme % 20) % 10) % 5) / 2)+       // nb pièces   2€
       " 1 x "+((((somme % 20) % 10) % 5) % 2);       // nb pièces   1€

    println(piecesARendre);
}
```

Version « un seul reste »

```
void algorithm()(
  int somme = readInt();
  int nb20, nb10, nb5, nb2, nb1, reste;
  somme = reste
  nb20  = sommereste / 20;
  reste = sommereste % 20;
  nb10  = reste / 10;
  reste = reste % 10;
  nb5   = reste / 5;
  reste = reste % 5;
  nb2   = reste / 2;
  reste = reste % 2;
  nb1   = reste / 1;
  nb1   = reste % 1;
  println("20 x "+ nb20 + "10 x " + nb10 + "5 x " + nb5 +
"2 x " + nb2 + "1 x "  +nb1};
}
```

Redondances !

Version « avec répétition »

```
void algorithm()(
  int somme = readInt();
  print(somme + " = ");

  for (int coupure = 20; coupure > 0; coupure = coupture/2) {
    print(coupure+" x "+(somme / coupure)+"€ ");
    somme = somme % coupure;
  }
}
```

Le jeu de Nim (fan-tan, tiouk-tiouk ...)

- Jeu à 2 joueurs à information complète
- *n* allumettes/jetons/cailloux au début du jeu
- chaque joueur retire de 1 à 3 allumettes par coup
- le joueur prenant la dernière allumette a perdu