Algorithmique
Partie 1

IUT Informatique de Lens, 1ère Année
Université d'Artois

Frédéric Koriche
koriche@cril.fr
2011 - Semestre 1

Sommaire

Modalités du cours

- Algorithmique : matière fondamentale de l'informatique, nécessaire pour aborder la plupart des autres matières
- 1,5h de cours, 3h de TD et 1,5h de TP par semaine
- Il faut travailler les cours, TD et TP, et surtout bien les comprendre.

Donc n'hésitez pas à poser des questions à vos enseignants !

Modalités du cours

- Algorithmique : matière fondamentale de l'informatique, nécessaire pour aborder la plupart des autres matières
- 1,5h de cours, 3h de TD et 1,5h de TP par semaine
- Il faut travailler les cours, TD et TP, et surtout bien les comprendre.

Donc n'hésitez pas à poser des questions à vos enseignants !

Modalités du cours

- Algorithmique : matière fondamentale de l'informatique, nécessaire pour aborder la plupart des autres matières
- 1,5h de cours, 3h de TD et 1,5h de TP par semaine
- Il faut travailler les cours, TD et TP, et surtout bien les comprendre.

Donc n'hésitez pas à poser des questions à vos enseignants !

Modalités du cours

- Algorithmique : matière fondamentale de l'informatique, nécessaire pour aborder la plupart des autres matières
- 1,5h de cours, 3h de TD et 1,5h de TP par semaine
- Il faut travailler les cours, TD et TP, et surtout bien les comprendre.

Donc n'hésitez pas à poser des questions à vos enseignants !

Plateforme d'e-learning

- Le cours sera présent sur Moodle avec les différents supports.
- Les sujets de TD, TP ainsi que divers exercices supplémentaires seront aussi placés sur Moodle.
- Même si vous disposerez des supports, *il est utile de prendre des notes pour bien comprendre !*

Plateforme d'e-learning

- Le cours sera présent sur Moodle avec les différents supports.
- Les sujets de TD, TP ainsi que divers exercices supplémentaires seront aussi placés sur Moodle.
- Même si vous disposerez des supports, *il est utile de prendre des notes pour bien comprendre !*

Plateforme d'e-learning

- Le cours sera présent sur Moodle avec les différents supports.
- Les sujets de TD, TP ainsi que divers exercices supplémentaires seront aussi placés sur Moodle.
- Même si vous disposerez des supports, *il est utile de prendre des notes pour bien comprendre !*

Contrôle des connaissances

- 3 devoirs surveillés (DS)
  - DS1 (coef. 1)
  - DS2 (coef. 1,5)
  - DS3 (coef. 2)
- 1 devoir surveillé final (1/3 de la note globale)
- Au moins un contrôle de TP
- Au moins un contrôle de TD

Le premier DS est dans un mois donc il faut travailler dès maintenant !

Contrôle des connaissances

- 3 devoirs surveillés (DS)
  - DS1 (coef. 1)
  - DS2 (coef. 1,5)
  - DS3 (coef. 2)
- 1 devoir surveillé final (1/3 de la note globale)
- Au moins un contrôle de TP
- Au moins un contrôle de TD

Le premier DS est dans un mois donc il faut travailler dès maintenant !

Contrôle des connaissances

- 3 devoirs surveillés (DS)
  - DS1 (coef. 1)
  - DS2 (coef. 1,5)
  - DS3 (coef. 2)
- 1 devoir surveillé final (1/3 de la note globale)
- Au moins un contrôle de TP
- Au moins un contrôle de TD

Le premier DS est dans un mois donc il faut travailler dès maintenant !

Contrôle des connaissances

- 3 devoirs surveillés (DS)
  - DS1 (coef. 1)
  - DS2 (coef. 1,5)
  - DS3 (coef. 2)
- 1 devoir surveillé final (1/3 de la note globale)
- Au moins un contrôle de TP
- Au moins un contrôle de TD

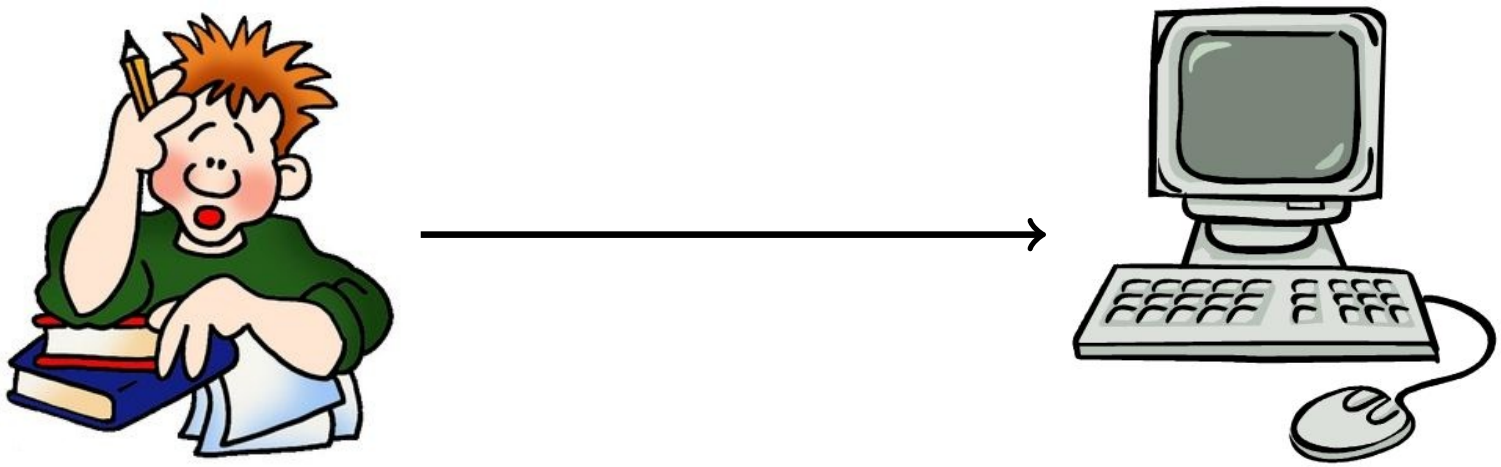
Le premier DS est dans un mois donc il faut travailler dès maintenant !

Contrôle des connaissances

- 3 devoirs surveillés (DS)
  - DS1 (coef. 1)
  - DS2 (coef. 1,5)
  - DS3 (coef. 2)
- 1 devoir surveillé final (1/3 de la note globale)
- Au moins un contrôle de TP
- Au moins un contrôle de TD

Le premier DS est dans un mois donc il faut travailler dès maintenant !

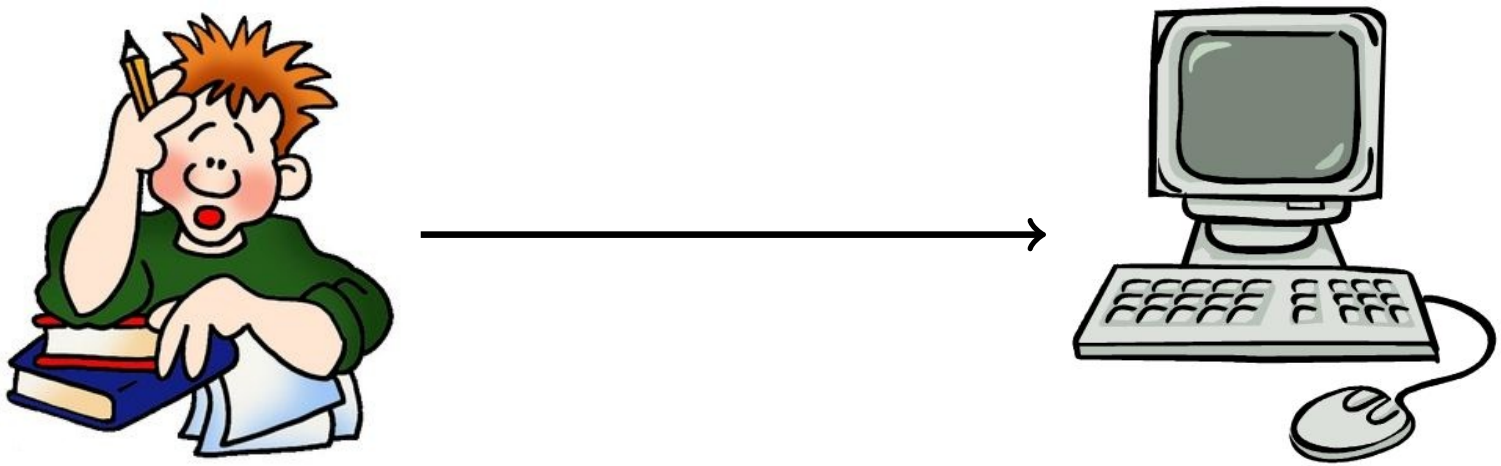
Sommaire



Programmation

Programmer c'est expliquer à une machine comment résoudre un problème. La programmation comprend trois grandes étapes :

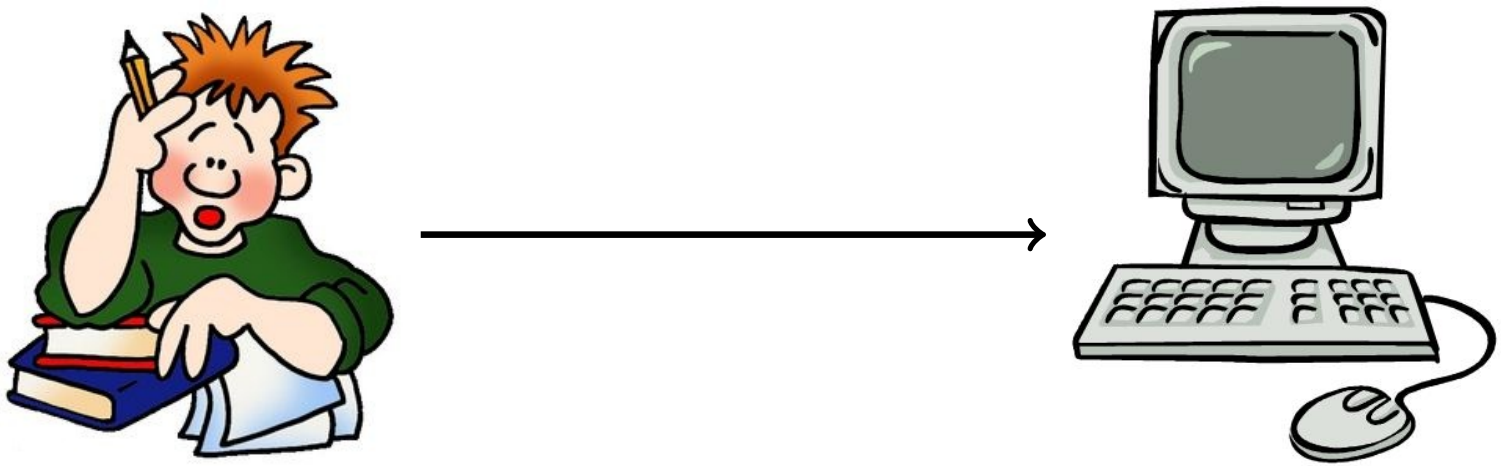
- Modélisation : énoncer le problème en termes explicites.
- Résolution : construire un algorithme qui spécifie, par une séquence d'actions, comment résoudre le problème.
- Codage : traduire l'algorithme dans un langage de programmation qui peut être traité par la machine.



Programmation

Programmer c'est expliquer à une machine comment résoudre un problème. La programmation comprend trois grandes étapes :

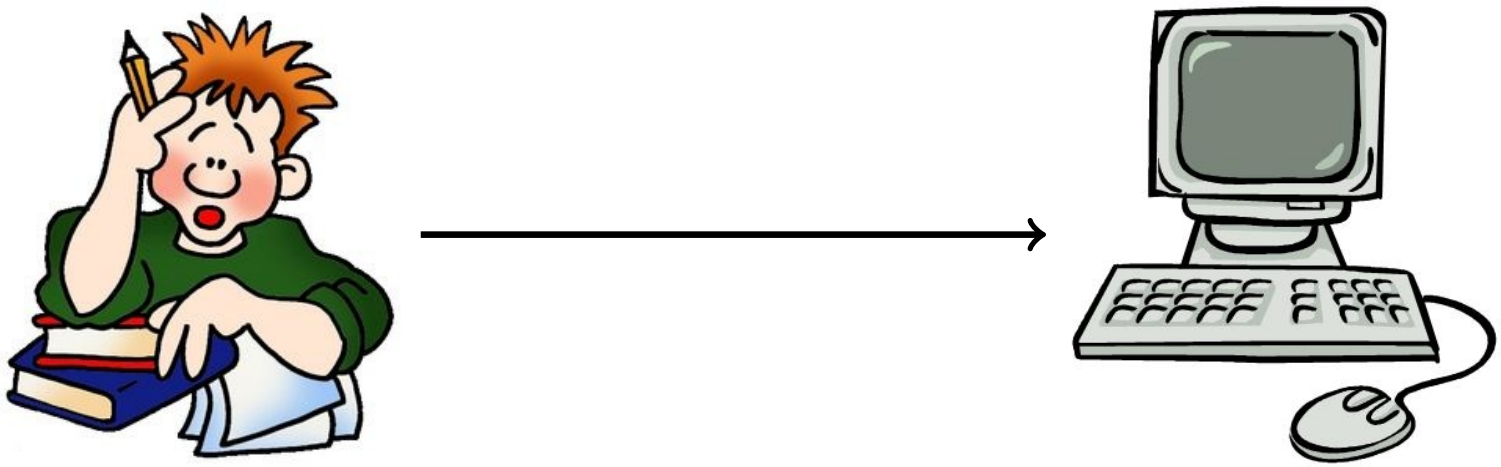
- Modélisation : énoncer le problème en termes explicites.
- Résolution : construire un algorithme qui spécifie, par une séquence d'actions, comment résoudre le problème.
- Codage : traduire l'algorithme dans un langage de programmation qui peut être traité par la machine.



Programmation

Programmer c'est expliquer à une machine comment résoudre un problème. La programmation comprend trois grandes étapes :

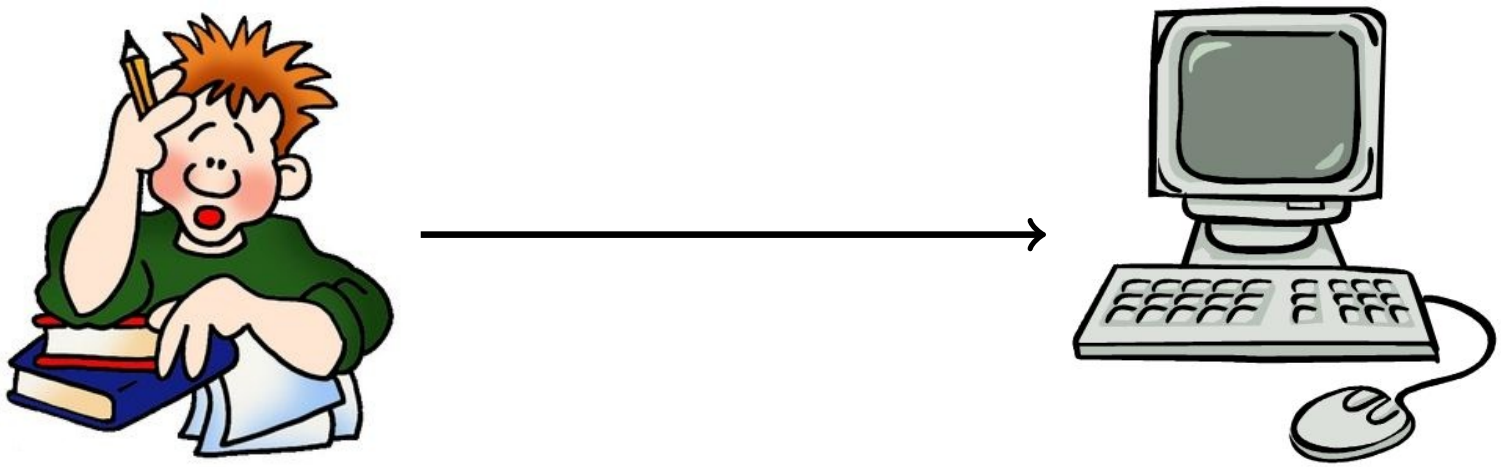
- Modélisation : énoncer le problème en termes explicites.
- Résolution : construire un algorithme qui spécifie, par une séquence d'actions, comment résoudre le problème.
- Codage : traduire l'algorithme dans un langage de programmation qui peut être traité par la machine.



Programmation

Programmer c'est expliquer à une machine comment résoudre un problème. La programmation comprend trois grandes étapes :

- Modélisation : énoncer le problème en termes explicites.
- Résolution : construire un algorithme qui spécifie, par une séquence d'actions, comment résoudre le problème.
- Codage : traduire l'algorithme dans un langage de programmation qui peut être traité par la machine.



Programmation

Programmer c'est expliquer à une machine comment résoudre un problème. La programmation comprend trois grandes étapes :

- Modélisation : énoncer le problème en termes explicites.
- Résolution : construire un algorithme qui spécifie, par une séquence d'actions, comment résoudre le problème.
- Codage : traduire l'algorithme dans un langage de programmation qui peut être traité par la machine.

L'algorithme du "café chaud"

- Quel est le problème ?

J'aimerais que Tom (mon robot) me fasse un bon café chaud !

L'algorithme du "café chaud"

- Quel est le problème ?

J'aimerais que Tom (mon robot) me fasse un bon café chaud !

- De quels ingrédients Tom dispose-t-il ?

L'algorithme du "café chaud"

- Quelles actions élémentaires Tom est-il capable de faire ?
  - Verser le café (liquide) dans la tasse
  - Verser le café moulu dans le filtre
  - Brancher la cafetière
  - Remplir la cafetière d'eau
  - Mettre le filtre dans la cafetière
  - Allumer/Eteindre la cafetière
  - Attendre que le café remplisse le pichet de la cafetière

- Quelle séquence d'actions doit accomplir Tom ?
  1. Brancher la cafetière
  2. Mettre le filtre dans la cafetière
  3. Verser le café moulu dans le filtre
  4. Remplir la cafetière d'eau
  5. Allumer la cafetière
  6. Attendre que le café remplisse le pichet de la cafetière
  7. Eteindre la cafetière
  8. Verser le café dans la tasse

L'algorithme du "café chaud"

- Quelles actions élémentaires Tom est-il capable de faire ?
  - Verser le café (liquide) dans la tasse
  - Verser le café moulu dans le filtre
  - Brancher la cafetière
  - Remplir la cafetière d'eau
  - Mettre le filtre dans la cafetière
  - Allumer/Eteindre la cafetière
  - Attendre que le café remplisse le pichet de la cafetière

- Quelle séquence d'actions doit accomplir Tom ?
  1. Brancher la cafetière
  2. Mettre le filtre dans la cafetière
  3. Verser le café moulu dans le filtre
  4. Remplir la cafetière d'eau
  5. Allumer la cafetière
  6. Attendre que le café remplisse le pichet de la cafetière
  7. Eteindre la cafetière
  8. Verser le café dans la tasse

Modélisation

Enoncé d'un problème

Un problème est spécifié par des données et un résultat :

- Données : c'est l'ensemble des objets (ingrédients) que nous avons à notre disposition
- Résultat : c'est la solution que nous cherchons à obtenir

Instance

Une instance de problème consiste en une valeur associée à chaque donnée du problème.

Exemple : le problème est divisible par

- Données : deux nombres entiers n et d
- Résultat : “oui” si n est divisible par d, et “non” sinon

Une instance du problème

- pour les valeurs $n = 20$ et $d = 10$,
- le résultat est la valeur “vrai”

Exemple : le problème est divisible par

- Données : deux nombres entiers n et d
- Résultat : “oui” si n est divisible par d, et “non” sinon

Une instance du problème

- pour les valeurs $n = 20$ et $d = 10$,
- le résultat est la valeur “vrai”

Exemple : le problème du tri

- Données : une séquence de n nombres réels $\langle x_1, x_2, \cdots, x_n \rangle$
- Résultat : une permutation $\langle x'_1, x'_2, \cdots, x'_n \rangle$ de la séquence telle que $x'_1 \leq x'_2 \leq \cdots \leq x'_n$

Une instance du problème

- pour la séquence :

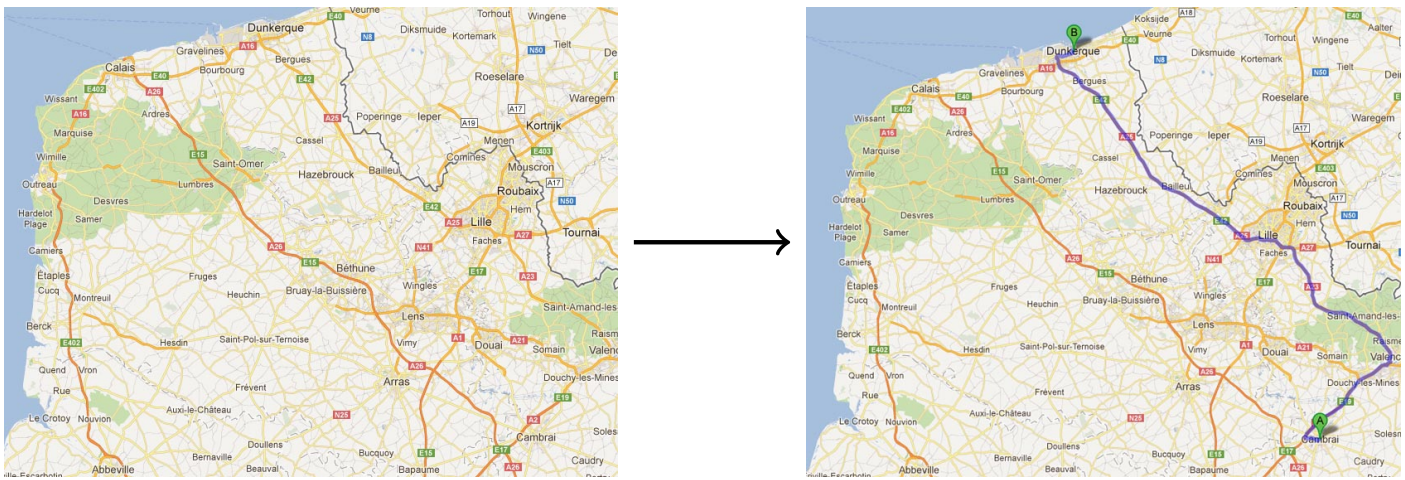
- le résultat est

Exemple : le problème de la recherche du chemin le plus court

- Données : une carte routière, une ville de départ et une ville d'arrivée
- Résultat : l'itinéraire le plus court (en kilomètres) depuis la ville de départ jusqu'à la ville d'arrivée

Une instance du problème

Pour la carte Nord Pas de Calais, la ville de Cambrai et la ville de Dunkerque, le chemin le plus court est :



Résolution

Algorithme

Un algorithme est une séquence d'actions (appelées *instructions*) qui permet de passer des données du problème au résultat attendu.

Le mot "algorithme" vient du mathématicien perse *Al Khuwarizmi*. Les premières écritures d'algorithmes datent de 5000 ans.

Un algorithme pour le problème mettre au carré

- Données : un nombre réel x
- Résultat : le carré de x

Algorithme : élèveAuCarré

---

variables

  réel x

  réel y

début

  lire x

  $y \longleftarrow x \times x$

  afficher y

fin

L'algorithme utilise deux variables de type réel

Le corps de l'algorithme lit la valeur de x, affecte le carré de cette valeur à y, et affiche y

Un algorithme pour le problème mettre au carré

- Données : un nombre réel x
- Résultat : le carré de x

Algorithme : élèveAuCarré

---

variables
  réel x
  réel y

début
  lire x
  $y \longleftarrow x \times x$
  afficher y
fin

L'algorithme utilise deux variables de type réel

Le corps de l'algorithme lit la valeur de x, affecte le carré de cette valeur à y, et affiche y

Un algorithme pour le problème mettre au carré

- Données : un nombre réel x
- Résultat : le carré de x

Algorithme : élèveAuCarré

---

variables

  réel x

  réel y

début

  lire x

  $y \longleftarrow x \times x$

  afficher y

fin

L'algorithme utilise deux variables de type réel

Le corps de l'algorithme lit la valeur de x, affecte le carré de cette valeur à y, et affiche y

Construire un Algorithme

Maîtriser l'algorithmique requiert deux qualités :

- L'intuition : savoir décomposer le problème en une série d'instructions.
- La rigueur : vérifier que la série d'instructions aboutit bien au résultat attendu.

Les deux qualités *s'acquièrent par l'expérience !*

Ecriture Algorithmique

Un algorithme est un manuel utilisateur pour résoudre un problème. Il ne dépend pas

- du langage de *programmation* dans lequel il sera codé
- de la *machine* qui exécutera le programme correspondant

Le langage algorithmique (ou *pseudo-code*) utilise des instructions faciles à comprendre et calculer.

Construire un Algorithme

Maîtriser l'algorithmique requiert deux qualités :

- L'intuition : savoir décomposer le problème en une série d'instructions.
- La rigueur : vérifier que la série d'instructions aboutit bien au résultat attendu.

Les deux qualités *s'acquièrent par l'expérience !*

Ecriture Algorithmique

Un algorithme est un manuel utilisateur pour résoudre un problème. Il ne dépend pas

- du langage de *programmation* dans lequel il sera codé
- de la *machine* qui exécutera le programme correspondant

Le langage algorithmique (ou *pseudo-code*) utilise des instructions faciles à comprendre et calculer.

Analyser un Algorithme

- Correction : peut-on garantir que pour chaque instance du problème, l'algorithme retourne le résultat attendu ?
- Complexité : de combien de temps et d'espace (mémoire) l'algorithme a-t-il besoin pour résoudre le problème ?

Note

- Dès ce semestre, nous étudierons la correction d'algorithmes simples
- La complexité algorithmique sera étudiée au second semestre

Analyser un Algorithme

- Correction : peut-on garantir que pour chaque instance du problème, l'algorithme retourne le résultat attendu ?
- Complexité : de combien de temps et d'espace (mémoire) l'algorithme a-t-il besoin pour résoudre le problème ?

Note

- Dès ce semestre, nous étudierons la correction d'algorithmes simples
- La complexité algorithmique sera étudiée au second semestre

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

Le programme utilise les fonctions entrées-sorties

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

Le programme utilise l'espace de noms de la bliothèque C++ standard

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

Le programme déclare les variables x et y de type réel

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

Fonction principale du programme ; le corps de la fonction est défini entre les accolades

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

La valeur de x est reçue depuis l'entrée standard (le clavier)

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

La valeur de y prend le carré de la valeur de x

Code C++ de l'algorithme élèveAuCarré

Codage

Programme

Un programme est la traduction d'un algorithme dans un langage qui peut être utilisé (compilé ou interprété) par la machine.

```
#include <iostream>
using namespace std;

float x;
float y;

main()
{
cin >> x;
y = x * x;
cout << y;
}
```

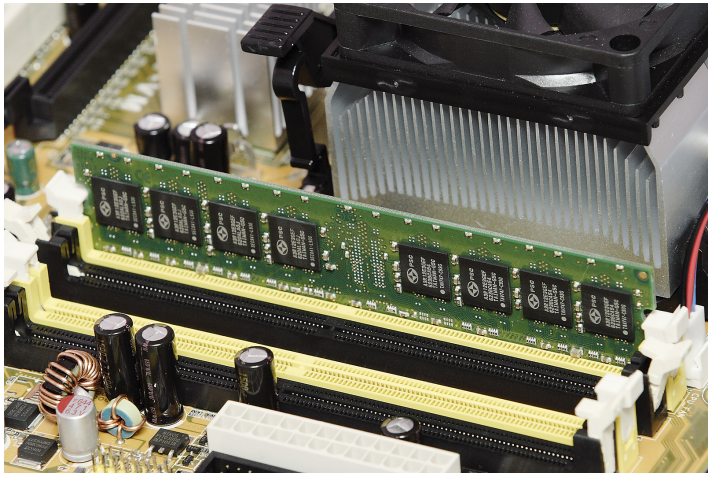
La valeur de y est envoyée à la sortie standard (l'écran)

Code C++ de l'algorithme élèveAuCarré

Sommaire

Données

En informatique, une donnée est la représentation d'un objet abstrait ou concret du monde. Dans une machine chaque donnée est *codée* et *stockée* dans une mémoire.

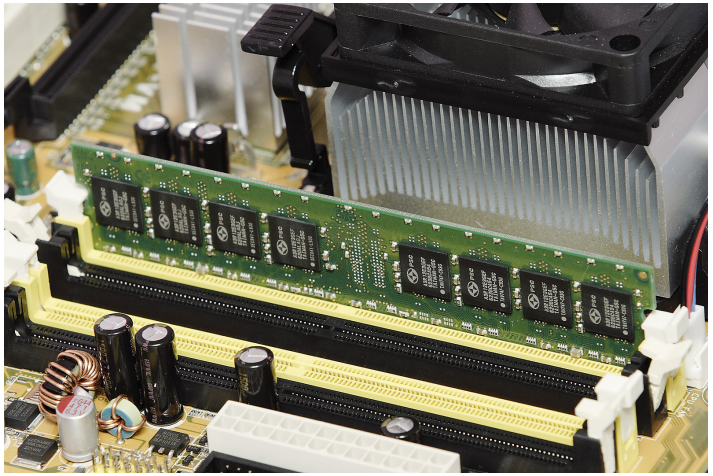


Toute donnée est spécifiée par :

- son nom : il désigne la donnée dans l'algorithme, et doit donc être le plus significatif possible.
- son type : il décrit le domaine de valeurs que peut prendre la donnée.
- sa nature : variable (peut changer de valeur) ou constante (ne peut pas changer de valeur).

Données

En informatique, une donnée est la représentation d'un objet abstrait ou concret du monde. Dans une machine chaque donnée est *codée* et *stockée* dans une mémoire.

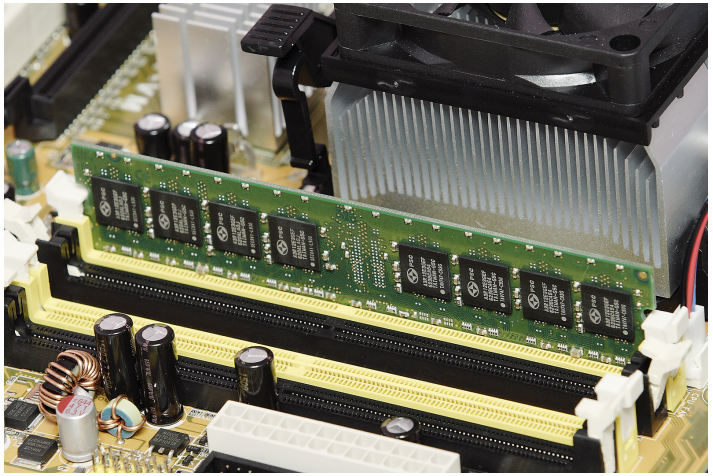


Toute donnée est spécifiée par :

- son nom : il désigne la donnée dans l'algorithme, et doit donc être le plus significatif possible.
- son type : il décrit le domaine de valeurs que peut prendre la donnée.
- sa nature : variable (peut changer de valeur) ou constante (ne peut pas changer de valeur).

Données

En informatique, une donnée est la représentation d'un objet abstrait ou concret du monde. Dans une machine chaque donnée est *codée* et *stockée* dans une mémoire.



Toute donnée est spécifiée par :

- son nom : il désigne la donnée dans l'algorithme, et doit donc être le plus significatif possible.
- son type : il décrit le domaine de valeurs que peut prendre la donnée.
- sa nature : variable (peut changer de valeur) ou constante (ne peut pas changer de valeur).

Type simple

Une donnée de type simple ne contient qu'une seule information

Type	Domaine de valeurs	Représentation (finie) en C++
booléen	{vrai,faux}	`bool`
caractère	table ASCII	`char`
entier	$\mathbb{Z}$	`int`
réel	$\mathbb{R}$	`float`

Type simple

Une donnée de type simple ne contient qu'une seule information

Type	Domaine de valeurs	Représentation (finie) en C++
booléen	{vrai,faux}	`bool`
caractère	table ASCII	`char`
entier	$\mathbb{Z}$	`int`
réel	$\mathbb{R}$	`float`

Codage des types simples

En programmation (C++) la valeur d'un type simple est codée sur un certain nombre de bits et stockée en mémoire. Pour une machine avec adressage 32 bits, le codage est :

Type C++	Codage	Domaine de valeurs
`bool`	8 bits	$\{0,1\}$
`char`	8 bits	$\{0,255\}$
`int`	32 bits	$\{\pm 2,147,483,648\}$
`float`	32 bits	$\{\pm 3.410^{\pm 38}\}$ (7 décimales)

Type simple

Une donnée de type simple ne contient qu'une seule information

Type	Domaine de valeurs	Représentation (finie) en C++
booléen	{vrai,faux}	`bool`
caractère	table ASCII	`char`
entier	$\mathbb{Z}$	`int`
réel	$\mathbb{R}$	`float`

Table ASCII

La table des codes ASCII (American Standard Code for Information Interchange) utilise un codage 7 bits pour normaliser les caractères (lettres, chiffres, ponctuation) du clavier. La table étendue (8 bits) permet de coder les caractères accentués, mais il existe différentes normes !

1 ┌	33 !	65 A	97 a	129 ü	161 ¡	193 Á	225 á
2 ┐	34 "	66 B	98 b	130 ,	162 ¢	194 Â	226 â
3 └	35 #	67 C	99 c	131 ƒ	163 £	195 Ã	227 ã
4 ┘	36 $	68 D	100 d	132 „	164 ¤	196 Ä	228 ä
5 |	37 %	69 E	101 e	133 …	165 ¥	197 Å	229 å
6 –	38 &	70 F	102 f	134 †	166 ¦	198 Æ	230 æ
7 •	39 '	71 G	103 g	135 ‡	167 §	199 Ç	231 ç
8 ◘	40 (	72 H	104 h	136 ˆ	168 ¨	200 È	232 è
9	41)	73 I	105 i	137 ‰	169 ©	201 É	233 é
10	42 *	74 J	106 j	138 Š	170 ª	202 Ê	234 ê
11 ♂	43 +	75 K	107 k	139 ‹	171 «	203 Ë	235 ë
12 ▫	44 ,	76 L	108 l	140 Œ	172 ¬	204 Ì	236 ì
13	45 -	77 M	109 m	141 ▯	173 -	205 Í	237 í
14 ♫	46 .	78 N	110 n	142 Ž	174 ®	206 Î	238 î
15 ☼	47 /	79 O	111 o	143 ▯	175 ¯	207 Ï	239 ï
16 ►	48 0	80 P	112 p	144 ▯	176 °	208 Ð	240 ð
17 ◄	49 1	81 Q	113 q	145 '	177 ±	209 Ñ	241 ñ
18 ↕	50 2	82 R	114 r	146 '	178 ²	210 Ò	242 ò
19 ‼	51 3	83 S	115 s	147 "	179 ³	211 Ó	243 ó
20 ¶	52 4	84 T	116 t	148 "	180 ´	212 Ô	244 ô
21 §	53 5	85 U	117 u	149 •	181 µ	213 Õ	245 õ
22 ▬	54 6	86 V	118 v	150 –	182 ¶	214 Ö	246 ö
23 ↨	55 7	87 W	119 w	151 —	183 ·	215 ×	247 ÷
24 ↑	56 8	88 X	120 x	152 ˜	184 ¸	216 Ø	248 ø
25 ↓	57 9	89 Y	121 y	153 ™	185 ¹	217 Ù	249 ù
26 →	58 :	90 Z	122 z	154 š	186 º	218 Ú	250 ú
27 ←	59 ;	91 [	123 {	155 ›	187 »	219 Û	251 û
28	60 <	92 \	124 |	156 œ	188 ¼	220 Ü	252 ü
29	61 =	93]	125 }	157 ▯	189 ½	221 Ý	253 ý
30	62 >	94 ^	126 ~	158 ž	190 ¾	222 Þ	254 þ
31	63 ?	95 _	127 ⌂	159 Ÿ	191 ¿	223 ß	255 ÿ
32	64 @	96 `	128 €	160	192 À	224 à	

Type structuré

Une donnée structurée contient une collection d'informations de type simple.

Type	Exemple de représentation en C++
tableau (statique)	`int t[10];`
chaîne (dynamique)	`string c;`

Un exemple de traitement de chaînes

Algorithme : afficheChaîne

```
variable
    chaîne nom

début
    afficher "Entrez votre nom : "
    lire nom
    afficher "Votre nom est : " nom
fin
```

```cpp
#include <iostream>
#include <string>
using namespace std;

string nom;

main()
{
cout << "Entrez votre nom: ";
cin >> nom;
cout << "Votre nom est: " << nom << endl;
}
```

Code C++ de afficheChaîne

Type structuré

Une donnée structurée contient une collection d'informations de type simple.

Type	Exemple de représentation en C++
tableau (statique)	`int t[10];`
chaîne (dynamique)	`string c;`

Un exemple de traitement de chaînes

Algorithme : afficheChaîne

variable
  chaîne nom

début
  afficher "Entrez votre nom : "
  lire nom
  afficher "Votre nom est : " nom
fin

```
#include <iostream>
#include <string>
using namespace std;

string nom;

main()
{
cout ≪ "Entrez votre nom: ";
cin ≫ nom;
cout ≪ "Votre nom est: " ≪ nom ≪ endl;
}
```

Code C++ de afficheChaîne

Sommaire

Opérateurs

Un opérateur est une fonction qui prend en entrée une liste de valeurs et retourne en sortie une valeur. La position d'un opérateur par rapport à ses opérandes peut être :

- *préfixe* : par exemple, l'inversion de signe `-x`.
- *infixe* : par exemple, l'addition `x + y`.
- *postfixe* : par exemple, l'incrémentation `x++`.

Opérateurs

Un opérateur est une fonction qui prend en entrée une liste de valeurs et retourne en sortie une valeur. La position d'un opérateur par rapport à ses opérandes peut être :

- *préfixe* : par exemple, l'inversion de signe `-x`.
- *infixe* : par exemple, l'addition `x + y`.
- *postfixe* : par exemple, l'incrémentation `x++`.

Opérateurs arithmétiques sur les entiers

Les entrées et la sortie sont des entiers

Nom	Pseudo-code	C++	Exemple
Addition	$+$	`+`	`i + j`
Soustraction	$-$	`-`	`i - j`
Multiplication	$\times$	`*`	`i * j`
Division entière	/	`/`	`5 / 2`
Reste	mod	`%`	`5 % 2`
Inversion de signe	$-$	`-`	`-i`

Opérateurs arithmétiques sur les réels

Au moins une des entrées est un réel et la sortie est un réel

Nom	Pseudo-code	C++	Exemple
Addition	$+$	`+`	`x + y`
Soustraction	$-$	`-`	`x - y`
Multiplication	$\times$	`*`	`x * y`
Division réelle	/	`/`	`5.0 / 2`
Inversion de signe	$-$	`-`	`-x`

Opérateurs arithmétiques sur les entiers

Les entrées et la sortie sont des entiers

Nom	Pseudo-code	C++	Exemple
Addition	$+$	`+`	`i + j`
Soustraction	$-$	`-`	`i - j`
Multiplication	$\times$	`*`	`i * j`
Division entière	/	/	`5 / 2`
Reste	mod	%	`5 % 2`
Inversion de signe	$-$	`-`	`-i`

Opérateurs arithmétiques sur les réels

Au moins une des entrées est un réel et la sortie est un réel

Nom	Pseudo-code	C++	Exemple
Addition	$+$	`+`	`x + y`
Soustraction	$-$	`-`	`x - y`
Multiplication	$\times$	`*`	`x * y`
Division réelle	/	/	`5.0 / 2`
Inversion de signe	$-$	`-`	`-x`

Opérateurs de comparaison

Les entrées sont des caractères ou nombres et la sortie est un booléen

Nom	Pseudo-code	C++	Exemple
Est égal à	=	==	`i == j`
Plus petit que	<	<	`i < j`
Plus grand que	>	>	`i > j`
Plus petit ou égal à	≤	<=	`i <= j`
Plus grand ou égal à	≥	>=	`i >= j`

Opérateurs logiques

Les entrées et la sortie sont des booléens

Nom	Pseudo-code	C++	Exemple
Et	et	&&	`a && b`
Ou	ou	||	`a || b`
Non	non	!	`! a`

Opérateurs de comparaison

Les entrées sont des caractères ou nombres et la sortie est un booléen

Nom	Pseudo-code	C++	Exemple
Est égal à	=	==	`i == j`
Plus petit que	<	<	`i < j`
Plus grand que	>	>	`i > j`
Plus petit ou égal à	≤	<=	`i <= j`
Plus grand ou égal à	≥	>=	`i >= j`

Opérateurs logiques

Les entrées et la sortie sont des booléens

Nom	Pseudo-code	C++	Exemple
Et	et	&&	`a && b`
Ou	ou	||	`a || b`
Non	non	!	`! a`

Sommaire

Instructions

Une instruction est une action que doit accomplir l'algorithme. Il existe cinq catégories d'instructions :

- La déclaration
- L'affectation
- La lecture/écriture
- Les tests
- Les boucles

Déclaration

Toute donnée utilisée par un algorithme doit être déclarée, afin que la machine puisse lui réserver un espace mémoire.

- Déclaration d'une variable réelle :

Pseudo-code	C++
variable réel x	`float x;`

- Déclaration d'une constante réelle :

Pseudo-code	C++
constante réel pi $\longleftarrow 3.14$	`const float pi = 3.14;`

Note

- En pseudo-code les variables (constantes) sont regroupées sous un bloc **variables**
- En C++ les variables de même type peuvent être déclarées ensembles. Par exemple, `float x,y,z;`

Déclaration

Toute donnée utilisée par un algorithme doit être déclarée, afin que la machine puisse lui réserver un espace mémoire.

- Déclaration d'une variable réelle :

Pseudo-code	C++
variable réel x	`float x;`

- Déclaration d'une constante réelle :

Pseudo-code	C++
constante réel pi $\longleftarrow 3.14$	`const float pi = 3.14;`

Note

- En pseudo-code les variables (constantes) sont regroupées sous un bloc **variables**
- En C++ les variables de même type peuvent être déclarées ensembles. Par exemple, `float x,y,z;`

Déclaration

Toute donnée utilisée par un algorithme doit être déclarée, afin que la machine puisse lui réserver un espace mémoire.

- Déclaration d'une variable réelle :

Pseudo-code	C++
variable réel x	`float x;`

- Déclaration d'une constante réelle :

Pseudo-code	C++
constante réel pi $\longleftarrow 3.14$	`const float pi = 3.14;`

Note

- En pseudo-code les variables (constantes) sont regroupées sous un bloc **variables**
- En C++ les variables de même type peuvent être déclarées ensembles. Par exemple, `float x,y,z;`

Déclaration

Toute donnée utilisée par un algorithme doit être déclarée, afin que la machine puisse lui réserver un espace mémoire.

- Déclaration d'une variable réelle :

Pseudo-code	C++
variable réel x	`float x;`

- Déclaration d'une constante réelle :

Pseudo-code	C++
constante réel pi $\longleftarrow 3.14$	`const float pi = 3.14;`

Note

- En pseudo-code les variables (constantes) sont regroupées sous un bloc **variables**
- En C++ les variables de même type peuvent être déclarées ensembles. Par exemple, `float x,y,z;`

Affectation

L'instruction d'affectation permet de changer la valeur d'une variable en appliquant un opérateur d'affectation

- L'affectation standard

Pseudo-code	C++
$y \leftarrow x$	`y = x;`

- L'incrémentation et la décrémentation

Pseudo-code	C++
$x \leftarrow x+1$	`x++;`
$x \leftarrow x-1$	`x--;`

Note

En C++ il existe de nombreux autres opérateurs d'affectation permettant d'appliquer une opération arithmétique (ou logique) sur l'entrée et d'affecter le résultat sur l'entrée.

Par exemple, l'instruction $x \longleftarrow x+2$ peut être traduite en C++ par `x += 2;`

Pour des raisons de lisibilité, il est conseillé d'utiliser ces opérateurs avec parcimonie !

Affectation

L'instruction d'affectation permet de changer la valeur d'une variable en appliquant un opérateur d'affectation

- L'affectation standard

Pseudo-code	C++
$y \leftarrow x$	`y = x;`

- L'incrémentation et la décrémentation

Pseudo-code	C++
$x \leftarrow x+1$	`x++;`
$x \leftarrow x-1$	`x--;`

Note

En C++ il existe de nombreux autres opérateurs d'affectation permettant d'appliquer une opération arithmétique (ou logique) sur l'entrée et d'affecter le résultat sur l'entrée.

Par exemple, l'instruction $x \longleftarrow x+2$ peut être traduite en C++ par `x += 2;`

Pour des raisons de lisibilité, il est conseillé d'utiliser ces opérateurs avec parcimonie !

Affectation

L'instruction d'affectation permet de changer la valeur d'une variable en appliquant un opérateur d'affectation

- L'affectation standard

Pseudo-code	C++
$y \leftarrow x$	`y = x;`

- L'incrémentation et la décrémentation

Pseudo-code	C++
$x \leftarrow x+1$	`x++;`
$x \leftarrow x-1$	`x--;`

Note

En C++ il existe de nombreux autres opérateurs d'affectation permettant d'appliquer une opération arithmétique (ou logique) sur l'entrée et d'affecter le résultat sur l'entrée.

Par exemple, l'instruction $x \longleftarrow x+2$ peut être traduite en C++ par `x += 2;`

Pour des raisons de lisibilité, il est conseillé d'utiliser ces opérateurs avec parcimonie !

Lecture/écriture

Dans un premier temps, la lecture et l'écriture vont se limiter à la communication par l'intermédiaire du clavier et de l'écran.

- La lecture standard (clavier)

Pseudo-code	C++
lire x	`cin >> x;`

- L'affichage standard (écran)

Pseudo-code	C++
afficher x y	`cout << x << y;`

Note

Dans la suite, nous verrons que la lecture (écriture) de fichiers est similaire à la lecture (écriture) standard.

Lecture/écriture

Dans un premier temps, la lecture et l'écriture vont se limiter à la communication par l'intermédiaire du clavier et de l'écran.

- La lecture standard (clavier)

Pseudo-code	C++
lire x	`cin >> x;`

- L'affichage standard (écran)

Pseudo-code	C++
afficher x y	`cout << x << y;`

Note

Dans la suite, nous verrons que la lecture (écriture) de fichiers est similaire à la lecture (écriture) standard.

Lecture/écriture

Dans un premier temps, la lecture et l'écriture vont se limiter à la communication par l'intermédiaire du clavier et de l'écran.

- La lecture standard (clavier)

Pseudo-code	C++
lire x	`cin >> x;`

- L'affichage standard (écran)

Pseudo-code	C++
afficher x y	`cout << x << y;`

Note

Dans la suite, nous verrons que la lecture (écriture) de fichiers est similaire à la lecture (écriture) standard.