3- Algorithmique

Définition : algorithmique

Critère algorithmique élémentaire

Une application courante ou un problème est automatisable (traitable par informatique) si

- Il est possible de définir et décrire parfaitement les données et les résultats de sortie
- Il est possible de décomposer le passage de ces données vers ces résultats en une suite finie d'opérations élémentaires dont chacune peut être exécutée par une machine

> *L'algorithmique est la transformation de la connaissance que l'humain possède sur un problème en actions élémentaires exécutées par l'ordinateur*

Ensemble de règles opératoires dont l'application permet de résoudre un problème au moyen d'un nombre fini d'opérations (ou actions)

Exemple 1: Fabrication d'un pain la « machine » réalisant ces actions élémentaires n'est bien sur pas un ordinateur !

Entrées : farine, eau, sel, levure Sortie : pain cuit

Opérations séquentielles

- Battre ensemble les ingrédients
- Faire monter la pâte 1h à 25°C
- Faire cuire 20mn à 200°C

De l' algorithmes au programme :

Programme :

codage d'un algorithme afin que l'ordinateur puisse exécuter les actions décrites dans l'algorithme

doit être écrit dans un langage « **compréhensible »** par l'ordinateur

c'est un langage de programmation (Assembleur (micropro), C, Fortran, Pascal, Cobol …)

A la conception d ’un ordinateur, est défini l ’ensemble des opérations élémentaires qu ’il peut réaliser. Ces opérations doivent être les plus simples possible pour diminuer la complexité des circuits électroniques. L ’ensemble des opérations élémentaires est appelé **langage machine**.

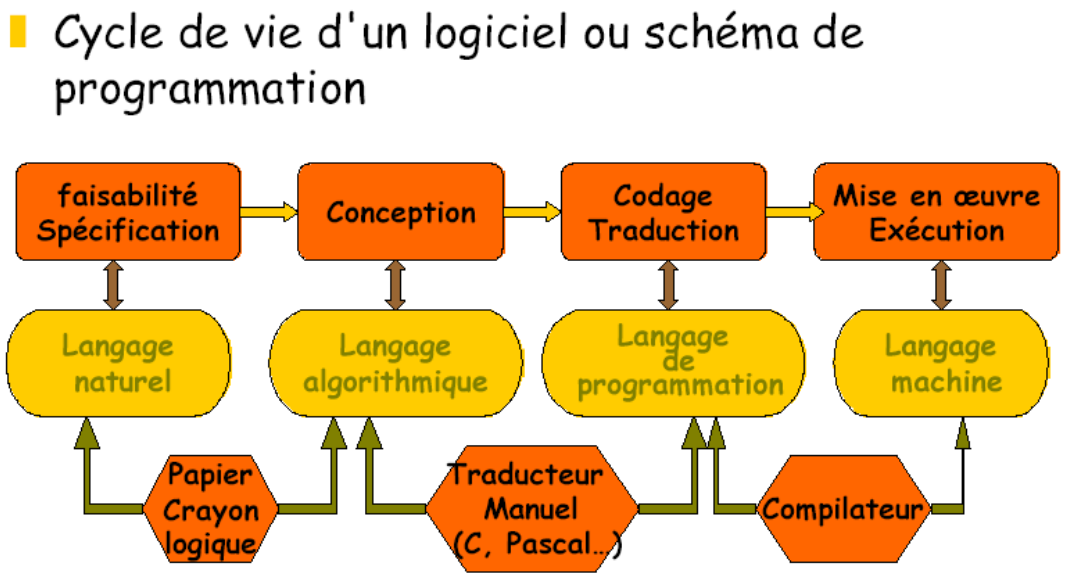
Un programme en "code-machine" est une suite d'instructions élémentaires, composées uniquement de 0 et de 1, exprimant les opérations de base que la machine peut physiquement exécuter: instructions de calcul (addition, ...) ou de traitement ("et" logique, ...), instructions d'échanges entre la mémoire principale et l'unité de calcul ou entre la mémoire principale et une mémoire externe, des instructions de test qui permettent par exemple de décider de la prochaine instruction à effectuer.

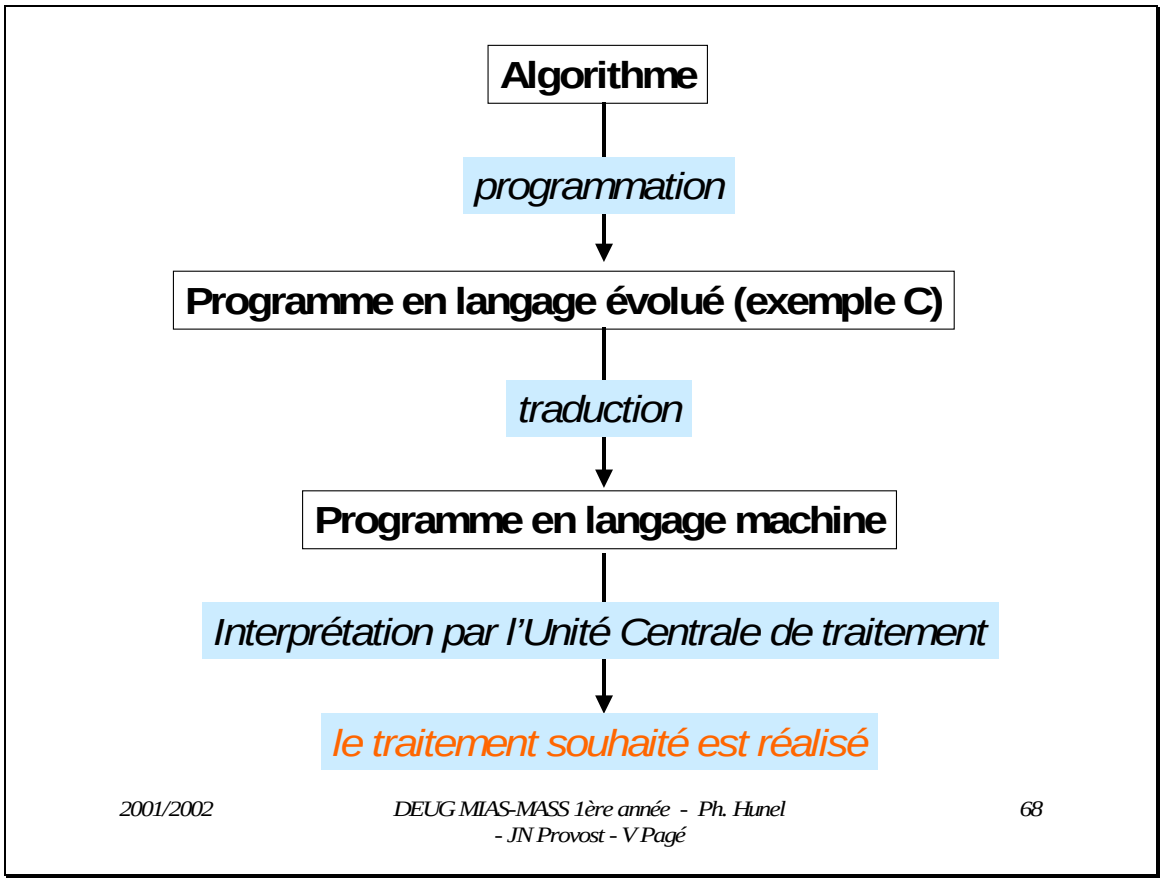
Voici en code machine de l'IBM 370 l'ordre de charger la valeur 293 dans le registre "3":

01011000 0011 00000000000100100100

charger 3 293

Ce type de code binaire est le seul que la machine puisse directement comprendre et donc réellement exécuter. Tout programme écrit dans un langage évolué devra par conséquent être d'abord traduit en code-machine avant d'être exécuté.





Importance des algorithmes :

- **La calculabilité des algorithmes** (convergence de l'algorithme)
  la méthode existe t'elle ?

- **La complexité des algorithmes** (nombre d'opérations nécessaires)

- **L'efficacité des algorithmes** (vitesse des algo: raisonnable)
  Temps d'exécution et taux d'occupation de la mémoire centrale

Exemple: Calcul de la circonférence d'un cercle de r=12

Ici la « machine » réalisant les opérations élémentaires est un ordinateur !!

Analyse : il faut connaître la valeur de r, calculer 2π r, afficher le résultat

Solution 1:

algorithme	Programme C
DEBUT Écrire (2*3.14159*12) FIN	#include <stdio.h> int main (void) { printf(" %10.5f\n ", 2*3.14159*12) ; }

Notion de variable ⌊__⌋ x dans laquelle on peut placer une valeur ou aller chercher une valeur pour faire du calcul

Solution 2:

algorithme	Programme C

Réel : x
DEBUT
Lire (x)
Écrire (2*3.14159*x)
FIN
	#include <stdio.h>
	int main (void)
	{
	double x ;
	printf("tapez la valeur du rayon\n") ;
	scanf(" %f ",&x) ;
	printf(" %10.5f\n ", 2*3.14159*x) ;
	}

3 variables réelles : ⌊__⌋ ⌊__⌋ ⌊__⌋

rayon pi perimetre

Solution 3:

algorithme	Programme C

Réel : rayon, Pi,Perimetre
DEBUT
Pi=3.14159
Écrire ('Entrer la valeur du rayon:')
Lire (rayon)
Perimetre<- 2*Pi*rayon
Écrire ('la circonférence du cercle est',Perimetre)
FIN
	#include <stdio.h>
	int main (void)
	{
	double Rayon,Pi , Perimetre ;
	/* lecture du rayon */
	printf(" tapez la valeur du rayon\n") ;
	scanf(" %f ",&Rayon) ;
	/* initialisation de Pi /*
	Pi=3.14159 ;
	/* calcul du perimetre et écriture */
	Perimetre=2*Pi*Rayon ;
	printf("%10.5f\n", Perimetre) ;
	}

Méthodologie simple

1. Définir clairement le problème
2. Chercher une méthode de résolution (formules…)
3. Définir les entrées nécessaires et les résultats obtenus
4. Écrire l'algorithme (langage algorithmique)

- Analyse méthodique descendante

Si le problème est trop complexe, le décomposer en sous-problèmes, et appliquer, pour chacun, la méthodologie simple

Décomposition du problème = description de + en + détaillée du problème = ensemble d'opérations élémentaires traductibles en langage de programmation

En résumé

Structure d'un algorithme		Exemple: calcul du salaire net avec plafond
Définition des données	- Nature (entrées, sorties, variables) - Type (entier, booléen, string…) - Nom (x, rayon,…)	- Entrée : *réel :* SH (Salaire Horaire), PR (Pourcentage de Retenues), P (Plafond); *entier :* NH (Nombre d'Heures) - Variables : *réel :* SB (Salaire de Base), R (Retenues) - Sorties : *réel :* SN (Salaire Net)
Initialisation	- Lecture des données - Initialisation des variables	**Ecrire** ('Entrer les données : salaire horaire, % de retenues, plafond et nombre d'heures de travail) **Lire** (SH, PR, P, NH)
Corps de l'algorithme	- Instructions de calculs - boucles - Etc..	SB=SH*NH */salaire de base* R=SB*PR */retenues* **SI** R>P **ALORS** R=P **FINSI** */plafonnement* SN=SB-R */salaire net*
Sortie	- Sauvegarde et écriture des résultats	**Ecrire** ('Le salaire net vaut', SN)

Langage algorithmique

- **Données :**valeur introduite par l'utilisateur (au clavier ou dans un fichier)

- Constante : identificateur dont la valeur, utilisée dans le programme, ne change pas lors de l'exécution (par ex. π)

- Variable: identificateur dont la valeur, utilisée dans le programme, peut changer lors de l'exécution (par ex. rayon d'un cercle)

Cette variable se réfère à un emplacement précis en mémoire centrale

Il est indispensable de:

- Définir la nature des données, constantes et variables
- Choisir les identificateurs les plus appropriés
- Pouvoir référencer tous les genres de valeurs (entier,reel,caractére…)

Ainsi une variable a un nom fixe (identificateur de variable)
Un type de contenu fixe (caractere, entier, reel….)
Par contre son contenu peut varier au cours de l'éxécution du programme

L'échange de données entre programme et utilisateur

pour recevoir de l'information de l'extérieur
- **Lire (x)** où x est une variable qui va prendre la valeur donnée par l'utilisateur au clavier

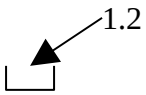
Écrire : pour fournir de l'information à l'extérieur
- **Écrire (x)** : la valeur contenue dans la variable x sera affichée
- Écrire ('Bonjour') : Bonjour sera écrit à l'écran

L'affectation : Opération qui consiste à attribuer une valeur à une variable notée :

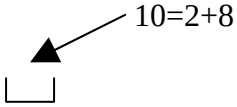
variable <-- valeur

Exemples:

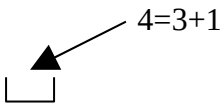
x <-- 1.2 (x prend la valeur 1.2)



n <-- 2+8 (n prend la valeur 10)



i <-- i+1 (i est incrémenté de 1)



ECRITURE D'ALGORITHMES :

Opérateurs arithmétiques :

+, -, *, /

- Exemple : x <- x-6; surface <- pi*sqr(rayon)

Opérateurs logiques

-Sur variables booléennes ou logiques : vrai/faux, 1/0, oui/non

- La structure de séquence

suite d'opérations (instructions) éxécutées dans l'ordre…Le paquet peut être precedé de DEBUT et suivi de FIN

- La structure de sélection simple

SI condition **ALORS**
    expression1
SINON
    expression2
FINSI

Recherche du Max de A et B:

algorithme	Programme C

Entrées : réel: A, B
Sortie : réel : max
DEBUT
ECRIRE(« donnez A et B »)
LIRE(A,B)
 	
SI A>B ALORS
    max <- A
SINON
    max<- B
FINSI
 	
ECRIRE(« Le maximum est»,max)
FIN
	#include <stdio.h>
	int main (void)
	{
	double A,B,max ;
	/* lecture données */
	printf(" tapez les valeurs de A et B\n") ;
	scanf(" %lf,%lf " , &A , &B) ;
	/* calcul du maximum */
	if (A > B)
	    max = A;
	else
	    max = B;
	/* écriture*/
	printf(" le max de %lf et %lf est %lf\n ",A,B,max);
	}

Exercice : Dérouler et comprendre l'algorithme et le programme en langage C suivants :
Pour les entrées 100.4 , 56 puis - 24.2 , 2.12 puis 3 , 3

algorithme	Programme C

Entrées : réel: A, B
Sortie : réel : max
DEBUT
ECRIRE(« donnez A et B »)
LIRE(A,B)
 	
SI A>B ALORS
max <- A
SINON
Max <- B
FINSI
 	
ECRIRE(« Le maximum est»,max)
FIN
	#include <stdio.h>
	int main (void)
	{
	double A,B,max ;
	/* lecture données */
	printf(" tapez les valeurs de A et B\n") ;
	scanf(" %lf,%lf " , &A , &B) ;
	/* calcul du maximum */
	if (A > B)
	max = A;
	else
	max = B;
	/* écriture*/
	printf(" le max de %lf et %lf est %lf\n ",A,B,max);
	}

Exercice : écrire un algorithme permettant de faire le calcul suivant :
En entrée on a deux **entiers** x et y , et le programme produit la sortie suivante :
x+y si x >=y et 2*x +2*y si x<y

répétitions (nombre connu)

POUR variable **DE** valeur1 **À** valeur2 **FAIRE**
    Expression
FINPOUR

Pour calculer X^n (n >0) et stocker le résultat dans Y:

algorithme	Programme C

Entrées : réel : X, entier : n
Sorties : réel : Y
Variables : entier : i
DEBUT
Lire (X,n)
 	
Y=1
POUR i DE 1 À n FAIRE
Y = Y * X
FINPOUR
 	
Écrire (X, 'puissance', n, 'vaut', Y)
FIN
	#include <stdio.h>
	int main (void)
	{
	double X,Y; int n , i ;
	/* lecture de X et n */
	printf(" tapez les valeurs de X et n \n") ;
	scanf("%lf,%d", &X , &n) ;
	/* calcul de X à la puissance n /*
	Y=1.0 ;
	for (i=1,(i<n),i=i+1)
	Y = Y * X;
	/* resultat */
	printf(" %lf puissance %d vaut %lf \n ", X , n ,Y) ;
	}

Pour calculer la moyenne de 16 notes:

algorithme	Programme C

Entrées : réel : Note
Sorties : réel : Moyenne
Variables : réel : Somme, entier : i
 	
DEBUT
Somme=0
POUR i DE 1 À 16 FAIRE
Lire (Note)
Somme = Somme + Note
FINPOUR
Moyenne = Somme / 16
Écrire ('La moyenne des 16 notes vaut', Moyenne)
FIN
	#include <stdio.h>
	int main (void)
	{
	double note , moyenne , somme ;
	int i ;
	/* initialisation de la somme */
	somme = 0.0 ;
	
	/* calcul de la somme */
	for (i=1,(i<16), i=i+1)
	{
	/* lecture de la note courante */
	printf(" donnez la note numéro %d \n ",i) ;
	scanf("%lf",,&note) ;
	/* ajout de la note à somme */
	somme = somme + note;
	}
	
	/* resultat et sortie du resultat */
	moyenne = somme /16 ;
	printf(" la moyenne est %lf \n",moyenne) ;
	}

La boucle TANT QUE: structure répétitive,
nombre de répétition inconnu

TANT-QUE condition **FAIRE**
expression
FINTANTQUE

La boucle REPETER…TANTQUE : structure répétitive, nombre de répétition inconnu

REPETER
Expression
TANTQUE condition

peut s'écrire

REPETER
Expression
JUSQUA condition opposée

Exemple
Pour calculer X^n (n >0) et stocker le résultat dans Y:

algorithme	Programme C

Entrées : réel : X, entier : n
Sorties : réel : Y
Variables : entier : i
 	
DEBUT
Lire (X , n)
 	
Y = 1
i = 1
TANTQUE i <= n FAIRE
Y = Y * X
i = i+1
FINTANTQUE
Écrire (X, 'puissance', n, 'vaut', Y)
FIN
	#include <stdio.h>
	int main (void)
	{
	double X,Y; int n , i ;
	/* lecture de X et n */
	printf(" tapez les valeurs de X et n \n") ;
	scanf("%lf,%d",&X, &n) ;
	
	/* calcul de X à la puissance n /*
	Y=1.0 ;
	i=1;
	while (I <= n)
	{
	Y = Y * X;
	i=i+1;
	}
	/* resultat */
	printf(" %lf puissance %d vaut %lf \n ", X ,n , Y) ;
	}

algorithme	Programme C

Entrées : réel : X, entier : n
Sorties : réel : Y
Variables : entier : i
 	
DEBUT
Lire (X , n)
 	
Y = 1
i = 1
REPETER | REPETER
Y = Y * X | Y = Y * X
i = i+1 | i=i +
TANTQUE i <= n JUSQUA i > n
Écrire (X, 'puissance', n, 'vaut', Y)
FIN
	#include <stdio.h>
	int main (void)
	{
	double X,Y; int n , i ;
	/* lecture de X et n */
	printf(" tapez les valeurs de X et n \n") ;
	scanf("%lf,%d",&X, &n) ;
	
	/* calcul de X à la puissance n /*
	Y=1.0 ;
	i=1;
	do
	{
	Y = Y * X;
	i=i+1;
	}
	while (i <= n)
	/* resultat */
	printf(" %lf puissance %d vaut %lf \n ", X ,n , Y) ;
	}

Exemple 2 Pour obliger l'utilisateur à entrer n>0 dans le calcul de X^n et éviter un échec.

algorithme	Programme C

Entrées : réel : X, entier : n
Sorties : réel : Y
Variables : entier : i
DEBUT
Lire (X)
LIRE(n)
TANTQUE NOT(n > 0) FAIRE
Ecrire ('Le nombre n'est pas strictement positif!!')
Ecrire ('Entrer un nombre positif non nul:')
Lire (n)
FINTANTQUE
 	
Y = 1
i = 1
TANTQUE i <= n FAIRE
Y = Y * X
i = i+1
FINTANTQUE
Écrire (X, 'puissance', n, 'vaut', Y)
FIN
	#include <stdio.h>
	int main (void)
	{
	double X,Y; int n , i ;
	/* lecture de X et n */
	printf("tapez la valeur de X \n") ; scanf("%lf",&X) ;
	printf(" tapez la valeur de n \n") ; scanf("%d",&n) ;
	while (!(n>0))
	{
	printf(" le nombre tapé n'est pas strict positif ! \n") ;
	printf("retapez un nombre correct!!! \n ") ;
	scanf("%d",&n) ;
	}
	/* calcul de X à la puissance n /*
	Y=1.0 ; i=1;
	while (I <= n)
	{
	Y = Y * X;
	i=i+1;
	}
	printf(" %f puissance %d vaut %f \n ", X ,n , Y) ;
	}

algorithme	Programme C

Entrées : réel : X, entier : n
Sorties : réel : Y
Variables : entier : i
 	
DEBUT
Lire (X)
 	
REPETER
Ecrire ('Entrer un nombre positif non nul:')
Lire (n)
TANTQUE n >= 0
 	
Y = 1
i = 1
TANTQUE i <= n FAIRE
Y = Y * X
i = i+1
FINTANTQUE
Écrire (X, 'puissance', n, 'vaut', Y)
FIN
	#include <stdio.h>
	int main (void)
	{
	double X,Y; int n , i ;
	/* lecture de X et n */
	printf(" tapez la valeur de X \n") ; scanf("%lf",&X) ;
	
	do
	{
	printf(" tapez la valeur de n \n") ;
	scanf("%d",&n) ;
	}
	while (n <= 0)
	
	/* calcul de X à la puissance n /*
	Y=1.0 ;
	i=1;
	while (I <= n)
	{
	Y = Y * X;
	i=i+1;
	}
	/* resultat */
	printf(" %f puissance %d vaut %f \n ", X ,n , Y) ;
	
	}

Exercice

écrire un algorithme permettant de faire le calcul suivant :
En entrée on a six entiers **sexe, an_nais, mois_nais, dept_nais, x et y** représentant le numéro de sécurité sociale de l'utilisateur (par exemple 1, 46, 04, 97, 129, 132) , et le programme produit les sorties suivantes :
« Bonjour monsieur » si c'est un homme et « Bonjour madame » si c'est une femme
« Tu es bien petit » si son age est inférieur ou égal à 6 ans
« que fais tu la a ton age » si il a entre 6 et 16 ans
« bonjour pépé » si il a plus de 16 ans
« bonjour la Guadeloupe » si dept_ nais vaut 97 et que x est un nombre commençant par 1

écrire ensuite le programme C correspondant

rappel : entier / entier donne le quotient de la division entière

Exercice

Faire un algorithme et le programme en C correspondant qui lit 20 notes et calcule deux moyennes , la moyenne des notes supérieures ou égales à 10 et la moyenne des notes inférieures à 10

Exercice
Ecrire un algorithme et le programme C correspondant qui lit un entier entre 1 et 10
Puis qui demande à un autre utilisateur ne connaissant pas le nombre qui a été rentré précédemment de donner un nombre entre 1 et 10
Dés que l'utilisateur tape le bon nombre on quitte le programme et on félicite le joueur
Sinon on lui indique si ce qu'il a donné est trop bas ou trop haut et on lui demande de redonner un nombre