Algorithmique Ch 2 : Variable, Affectation

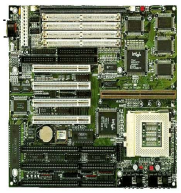
Groupe KabritBwa

13 juin 2021

1 Architecture sommaire d'un ordinateur

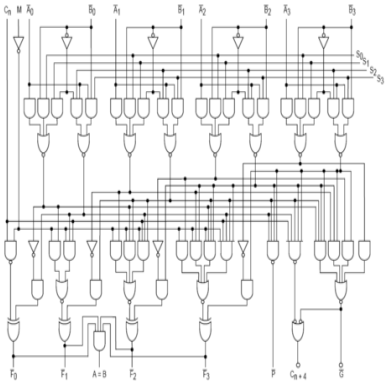
Un ordinateur est formé globalement de 3 parties :

1. **Un organe ou périphérique d'entrée**
   par exemple, un clavier, une souris, ...
2. **Une unité centrale ou carte-mère** formée elle-aussi de 3 parties :



   (a) **Une mémoire centrale** séparée en deux
      i. **Mémoire vive ou RAM (Random Access Memory**
      où l'on pourra stocker des données. les cases-mémoires qui y sont seront ce que

      nous appelelrons des variables informatiques.
      ii. **Mémoire morte ou ROM (Read Only memory)**
      Le constructeur y stocke des routines système.



(b) **Unité arithmétique et logique**
Elle contient les circuits électroniques qui permettent de faire les opérations arithmétiques $(+, -, *, /, \dots)$ et les opérations logiques $(=, <, >, <>, \dots)$

(c) **Unité de contrôle et de commande**
pilotées par le micro-processeur impulsé par une horloge qui bat la fréquence en Hz.

3. **Un organe ou périphérique de sortie**
   par exemple un écran ou une imprimante

2 3 modèles d'ordinateurs

2.1 Les super-calculateurs

En juin 2000, le groupe informatique IBM a annoncé la construction du plus rapide et du plus gros ordinateur au monde, de la taille de deux terrains de basket, pour simuler les essais nucléaires du département américain de l'Energie.
Cet ordinateur géant, appelé ASCI White est capable d'effectuer quelque $12,3$ trillions (c'est-à-dire 12300 milliards) d'opérations par seconde.
L'ASCI White va assurer le programme de sécurité et de fiabilité de l'arsenal de l'armement nucléaire américain sans avoir recours à des essais réels.
Lors des essais dans les laboratoires d'IBM, l'ordinateur a opéré $12,3$ trillions d'opérations par seconde, dépassant de 23% le cahier des charges du département de l'Energie , selon IBM.
ASCI White marque un nouveau seuil dans la technologie informatique puisque c'est le premier ordinateur à dépasser la barre des deux chiffres en téra opérations.Il faudrait 10 millions d'années à un homme pour faire le même nombre d'opérations.
Il faudra 28 camions pour transporter les différents éléments de cet ordinateur qui sera assemblé au laboratoire californien Lawrence Livermore du département de l'énergie.
(Sources : France Antilles - Vendredi 30 juin 2000)
ASCI White était un supercalculateur au Laboratoire Lawrence Livermore en Californie .
Il s'agissait d'une grappe d'ordinateurs basée sur IBM commerciale de RS/6000 SP ordinateur. 512 de ces machines ont été reliés entre eux pour ASCI White, avec 16 processeurs par nœud et 8192 processeurs au total de 6 téraoctets de mémoire et 160 téraoctets de stockage sur disque. Malgré ces statistiques redoutables, chaque processeur est lent ici à 2006, les normes, fonctionnant à une simple 375 MHz. Par conséquent, il a été presque exclusivement utilisé pour les calculs nécessitant des dizaines de processeurs. L'ordinateur pèse 106 tonnes et consommé 3 MW d'électricité avec un autre 3 MW nécessaire pour le refroidissement. Il avait une vitesse de traitement théorique de 12,3 téraflops . Le système a fonctionné IBM système d'exploitation AIX .
ASCI White était composé de trois systèmes différents, les 512 nœuds blancs, la glace et le noeud 28 gel 68 noeuds.
Le système a été construit en Poughkeepsie, New York . Achevé en Juin 2000, il fut transporté à des installations spécialement construites en Californie et officiellement inauguré le 15 Août 2001. Performances prétendait 12.300 gigaflops, bien que cela n'a pas été atteint dans les largement acceptées LINPACK tests. Le système a coûté 110 millions de dollars.
Il a été construit comme la troisième phase de l' Initiative d'informatique stratégique accéléré (ASCI) lancé par les Etats-Unis Ministère de l'énergie et de la National Nuclear Security Administration pour construire un simulateur pour remplacer direct armes de destruction massive des tests après le moratoire sur les essais a commencé par le président George HW Bush en 1992 et

prolongée par Bill Clinton en 1993. La machine a été mise hors service début Juillet 27, 2006. *(Sources : Wikipedia - Dimanche 2 décembre 2012)*

2.2 Les mini-ordinateurs

Ils sont essentiellement utilisés en gestion.

2.3 Les micro-ordinateurs

Ce sont les ordinateurs domestiques. Au début de la micro-informatique dans les années 1970, il y avait 4 grandes familles :

1. La famille des ATARI très utilisée au début par le monde musical à cause de son interface MIDI. Atari est à l'origine une entreprise américaine pionnière dans l'industrie du jeu vidéo fondée en 1972 par Nolan Bushnell et Ted Dabney. Elle va déposer son bilan le 21 janvier 2013.

2. La famille des AMIGA :

Amiga 500 avec le moniteur et lecteur de disquette externe de marque Amiga, 1987

   L'Amiga est une famille d'ordinateurs personnels commercialisée par Commodore International entre 1985 et 1994. Dans les années 1990 il était très populaire sur la scène démo, parmi les amateurs de jeux vidéo et dans l'industrie du cinéma et de la télévision.

3. La famille des IBM-PC et compatibles articulée au début autour du micro-processeur INTEL et du système d'exploitation PC-DOS pour IBM et MS-DOS(créé par la société Microsoft de Bill GATES) Il comportait un microprocesseur Intel 8088 cadencé à 4,77 MHz et une mémoire vive de 16 ko pouvant être portée à 256 ko. Il disposait selon les modèles d'aucun, un ou deux lecteurs de disquettes 5 pouces 1/4 de 160 ko simple face, 320 ko double face, 512 ko double face double densité. Il avait 5 ports ISA, bus 8 bit, pour carte d'extension comme la carte d'extension mémoire, carte vidéo CGA. Il était équipé d'un interprète du langage BASIC Microsoft en mémoire morte et pouvait gérer4 une unité de cassette externe.

IBM PC 5150

4. La famille APPLE (crée par Steeve JOBS et) articulée au début autour du microprocesseur MOTOROLA et qui depuis une dizaine d'années s'articule autour du microprocesseur INTEL mais avec un système d'exploitation particulier MACOS (un dérivé d'UNIX)

Les deux premières familles ATARI et AMIGA ont disparu.

3 Affectation

Dans un langage de programmation, lorsque l'on veut faire entrer une valeur par exemple 15 dans une case-mémoire A directement par le micro-processeur. Il suffit d'écrire pour cela l'instruction $A := 15$
On réalise alors ce que l'on appelle une affectation. La case-mémoire A s'appelle une variable informatique.
On peut réaliser par exemple des affectations plus complexes du type

$$A := A + 1$$

qui est une incrémentation ou

$$A := A - 1$$

qui est une décrémentation.
Attention

A	$:=$	$A + 1$
A gauche c'est le contenant ou le nom de la case − memoire ou variable		*A droite, c'est le contenu ici le contenu de A est incremente de* 1 *et stocke dans A*

Il est fort conseillé pour suivre l'évolution des cases-mémoires(ou variables informatiques) de faire un tableau d'exécution.
En SCILAB, l'affectation se note $A = 15$ à ne pas confondre avec le booléen $A == 15$

qui est vrai si la case A contient 15 et faux sinon.

4 Activité

Créer un algorithme qui fait l'ordinateur permuter le contenu de 2 cases-mémoires A et B remplies au clavier auparavant par un utilisateur

4.1

```
Algorithme ECHANGE1;
  Variables A,B,C de même type ;
début
  (*entrée des données *)
    lire au clavier le contenu de la case mémoire A;
    lire au clavier le contenu de la case mémoire B;
  (* traitement *)
    stocker dans la case C le contenu de A;
    stocker dans la case A le contenu de B;
    stocker dans la case B le contenu de C;
  (*sortie des résultats *)
    afficher à l'écran le contenu de la case mémoire A;
    afficher à l'écran le contenu de la case mémoire B;
fin.
```

Lorsqu'on programme pour éviter d'avoir un curseur nu à l'écran lors des entrées au clavier, **dorénavant, on précédera toujours les entrées (*lire A*) par un affichage sommaire et explicatif.**

4.2 Codage en AlgoBox

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     B EST_DU_TYPE NOMBRE
4     C EST_DU_TYPE NOMBRE
5   DEBUT_ALGORITHME
6     AFFICHER "Saisir au clavier le contenu de A :"
7     LIRE A
8     AFFICHER A
9     AFFICHER "Saisir au clavier le contenu de B :"
10    LIRE B
11    AFFICHER B
12    C PREND_LA_VALEUR A
13    A PREND_LA_VALEUR B
14    B PREND_LA_VALEUR C
15    AFFICHER "Le contenu de A est :"
16    AFFICHER A
17    AFFICHER "Le contenu de B est :"
18    AFFICHER B
19
20  FIN_ALGORITHME
```

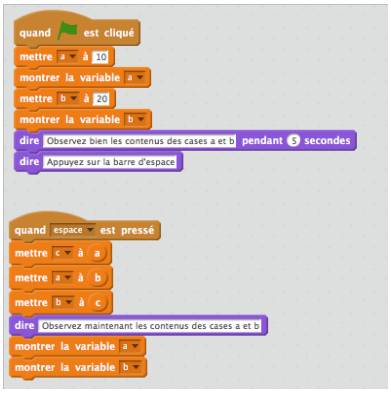
4.3 Codage enTurbo-Pascal

```
Program ECHANGE1;
  Var A,B,C :integer ;
begin
  (*entrée des données *)
    write(' Tapez au clavier le contenu de la case mémoire A = ');
    readln(A);
    write(' Tapez au clavier le contenu de la case mémoire B= ');
    readln(B);
  (* traitement *)
    C := A;
    A := B;
    B:= C;
  (*sortie des résultats *)
    writeln(' Le contenu de A est ', A);
    writeln(' Le contenu de B est ', B);
end.
```

	A	B	C	$ecran$
$lire(A)$	3			
$lire(B)$		8		
$C := A$			3	
$A := B$	8			
$B := C$		3		
$afficher(A)$				8
$afficher(B)$				3

On peut aussi permuter le contenu de deux cases contenant des noms ou des chaînes de caractères : Exemple en Turbo-Pascal, l'on saisit dans A : 'Ti Sonson' et dans B : 'Man Zosiane' à condition de déclarer A : *string* et B : *string*.
Le type **string** désignant le type chaîne de caractères.

4.4 Codage en Scratch



4.5 Codage en Scilab

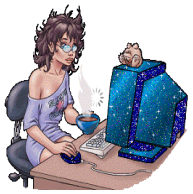
```
a=input("Entrez au clavier une valeur a = ")
b=input("Entrez au clavier une valeur b = ")
c =a
a=b
b=c
disp(a,"la nouvelle valeur de a est :")
disp(b,"la nouvelle valeur de b est :")
```

5 Activité

Faites tourner à la main l'algorithme suivant qui fait l'ordinateur permuter les contenus de deux cases A et B saisis au clavier par un utilisateur.
Cette méthode n'utilise que deux cases A et B.
Cet algorithme est donc plus économe en mémoire.

5.1

```
Algorithme ECHANGE2;
  Variables A,B de même type ;
début
  (*entrée des données *)
    lire au clavier le contenu de la case mémoire A;
    lire au clavier le contenu de la case mémoire B;
  (* traitement *)
    stocker dans la case A le contenu de A + B;
    stocker dans la case B le contenu de A - B;
    stocker dans la case A le contenu de A - B;
  (*sortie des résultats *)
    afficher à l'écran le contenu de la case mémoire A;
    afficher à l'écran le contenu de la case mémoire B;
fin.
```



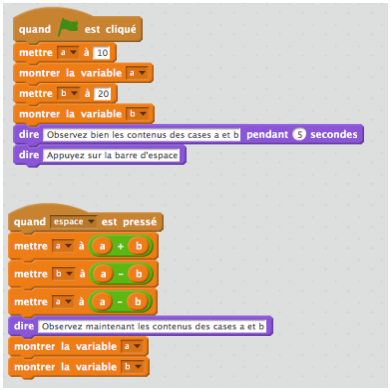
5.2 Codage en Algobox

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     B EST_DU_TYPE NOMBRE
4   DEBUT_ALGORITHME
5     AFFICHER "Saisir au clavier le contenu de A :"
6     LIRE A
7     AFFICHER A
8     AFFICHER "Saisir au clavier le contenu de B :"
9    LIRE B
10    AFFICHER B
11    A PREND_LA_VALEUR A + B
12    B PREND_LA_VALEUR A - B
13    A PREND_LA_VALEUR A - B
14    AFFICHER "Le contenu de A est :"
15    AFFICHER A
16    AFFICHER "Le contenu de B est :"
17    AFFICHER B
18     FIN_ALGORITHME
```

5.3 Codage en Turbo-Pascal

```
Program ECHANGE2
  Var A,B :integer ;
begin
  (*entrée des données *)
    write(' Tapez au clavier le contenu de la case mémoire A = ');
    readln(A);
    write(' Tapez au clavier le contenu de la case mémoire B= ');
    readln(B);
  (* traitement *)
    A := A + B;
    B := A - B;
    A := A - B;
  (*sortie des résultats *)
    writeln(' Le contenu de A est ', A);
    writeln(' Le contenu de B est ', B);
end.
```

5.4 Codage en Scratch



5.5 Codage en Scilab

```
a=input("Entrez au clavier une valeur a = ")
b=input("Entrez au clavier une valeur b = ")
a = a+b
b = a-b
a = a-b
disp(a,"la nouvelle valeur de a est :")
disp(b,"la nouvelle valeur de b est :")
```

	A	B	$ecran$
$lire(A)$	3		
$lire(B)$		8	
$A := A + B$	11		
$B := A - B$		3	
$A := A - B$	8		
$affiche r(A)$		8	
$affiche r(B)$			3

On peut suivre de façon formelle l'évolution du prédicat suivant $\{A = a \ et \ B = b\}$ après chaque affectation :
début
$\{A = a \ et \ B = b\}$
$A := A + B$;
$\{A = a + b \ et \ B = b\}$
$B := A - B$;
$\{A = a \ et \ B = a + b - b = a\}$
$A := A - B$;
$\{A = a + b - a = b \ et \ B = a\}$
fin
Attention , ce deuxième algorithme bien que plus économe en mémoire n'est pas conseillé car il n'est pas très fiable.
Il suffit d'imaginer les effets des erreurs d'arrondis lors des calculs $A+B$; $A-B$ pour un contenu de B qui soit négligeable devant le contenu de A.
L'affectation est l'instruction qui amène les pires ennuis.

Voici un exemple d'effets surprenants dûs aux calculs approchés :

Instructions	Calcul formel	Arithmétique décimale à 2 chiffres	Arithmétique décimale à 6 chiffres
$lire(C)$	$C = 4/3$	$C = 1,33$	$C = 1,333333$
$X := 14 - 3 * C$	$X = 10$	$X = 14 - 3 \times 1,33 = 10,01$	$X = 14 - 3 \times 1,333333 = 10,000001$
$Y := 100 * (X - 10)$	$Y = 0$	$Y = 1$	$Y = 0,0001$
$affiche r(Y)$			

L'augmentation de la précision des calculs décimaux peut amener des erreurs d'arrondi.

Exemple sous Maple :

ψ^{40} est évalué directement en passant par le logarithme alors que $\frac{228826127}{2}$ et $-102334155\sqrt{52}$

```
> restart;
> psi := (1 - sqrt(5))/2;
```

$$\psi := \frac{1}{2} - \frac{1}{2}\sqrt{5}$$

```
> psi^40;
```

$$\left(\frac{1}{2} - \frac{1}{2}\sqrt{5}\right)^{40}$$

```
> expand(%);
```

$$\frac{228826127}{2} - \frac{102334155}{2}\sqrt{5}$$

```
> evalf(%);
```

$$0.$$

```
> evalf(%%%);
```

$$4.370130127\,10^{-9}$$

sont évalués par **evalf** à la précision prévue sous Maple.