Programmation C

Année Universitaire 2022/2023

Objectifs du cours

1. ***appréhender les composants d'un ordinateur***
2. ***Faire une introduction sur l'algorithmique***
3. ***Apprendre les concepts de base de :***
   - ***La programmation C ;***

4. ***Mettre en œuvre ces concepts pour :***
   - ***Analyser des problèmes simples ;***
   - ***Ecrire les programmes correspondants.***
   - ***Concevoir des systèmes micro programmés***

Plan du cours

1. **Généralités :**
   - **Programmation C.**
2. **La Programmation C :**
   - **les variables : Structuration et affectation des données ;**
   - **Entrées-sorties ;**
   - **Opérateurs ;**
   - **Structures de contrôle de flux ;**
   - **Tableaux ;**
   - **Fonctions ;**
   - **Gestion des fichiers ;**
   - **Structures ;….**

Généralités

Informatique : c'est la discipline du *traitement automatique de l'information*.

L'informatique nécessite un ensemble d'outils appelés Système informatique composé :

- **Le Hardware : architecture physique de l'ordinateur (composants, circuits électroniques, périphériques, …) ;**
- **Le Software : Logiciels intégrés qui font fonctionner l'ordinateur (environnements, systèmes d'exploitations, langage de programmation, …)**

Généralités

Système informatique

Matérielle

Hardware

Logicielle

Software

Le Hardware

Architecture Physique de l'Ordinateur

Qu'est ce qu'un ordinateur ?

Un ordinateur est un appareil électronique qui permet de traiter les informations de façon automatique.

Il existe plusieurs types d'ordinateur :

- **Personal Computer PC;**
- **Stations : SUN, Alpha, …**

Dans la suite, on s'intéresse uniquement au PC

Le Hardware

Architecture Physique de l'Ordinateur

Quels sont les constituants essentiels d'un ordinateur ?

Un ordinateur est constitué par :

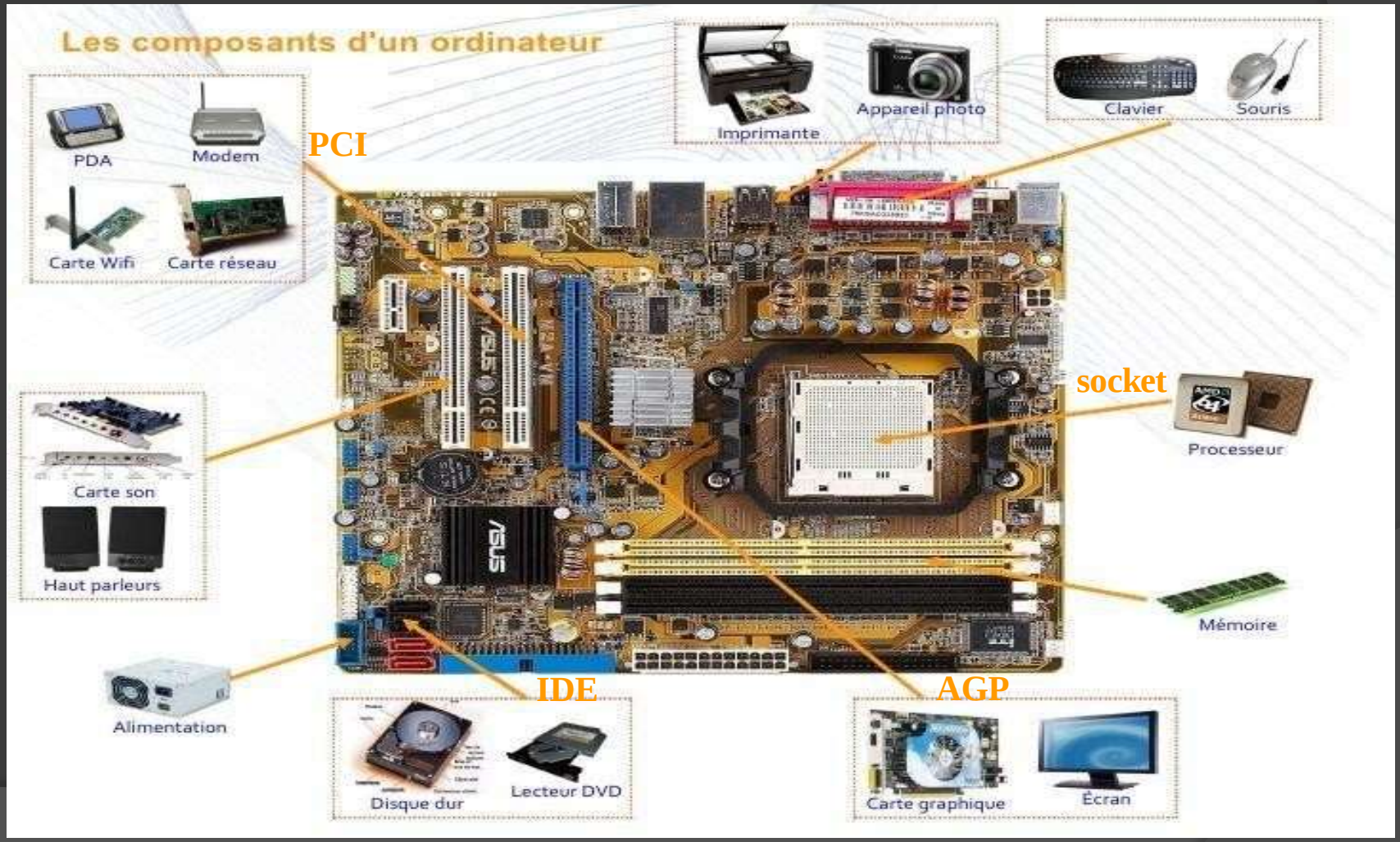
1. **Le microprocesseur ;**
2. **Les mémoires ;**
3. **Les bus ;**
4. **Les périphériques.**

Ces composants sont regroupés sur la carte mère du PC.

Le Hardware

Architecture Physique de l'Ordinateur

La Carte mère : L'élément principal de l'ordinateur,



Le Hardware

Architecture Physique de l'Ordinateur

1. Le processeur : appelé aussi microprocesseur ou UCT(unité centrale de traitement) en anglais central processing unit, CPU) est l'élément principal duquel découlent les performances de l'ordinateur.

Le choix du processeur : sa marque (AMD, INTEL) sa fréquence en Ghz, le nombre de cœur) déterminent la puissance de votre PC.

Le Hardware

Architecture Physique de l'Ordinateur

Générations d'ordinateurs de 1945 à nos jours

Première génération : (1945-1955)

✦ Tubes à vide et tableaux d'interrupteurs ✦

En septembre 1945, Presper Eckert et John William Mauchly achevèrent l'ENIAC (*Electronic Numerical Integrator and Computer*), gros calculateur entièrement électronique. Il avait été commandé en 1943 par l'armée américaine afin d'effectuer les calculs de balistique,

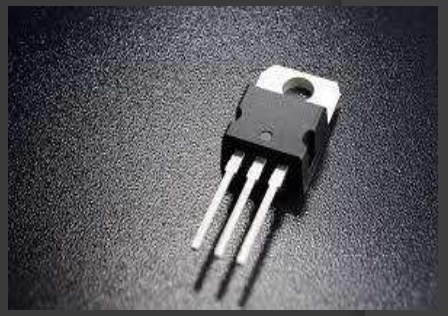
Le Hardware

Architecture Physique de l'Ordinateur

Générations d'ordinateurs de 1945 à nos jours

Deuxième génération : (1955-1965)

✦ Transistors et systèmes par lots ✦



La deuxième génération d'ordinateurs est basée sur l'invention du transistor en 1947. Cela permit de remplacer le fragile et encombrant tube électronique par un composant plus petit et fiable. Les ordinateurs composés de transistors sont considérés comme la deuxième génération et ont dominé l'informatique dans la fin des années 1950 et le début des années 1960.

En 1956, IBM sortit le premier système à base de disque dur, le Ramac 05 (Random Access Method of Accounting and Control). L'IBM 350 utilisait 50 disques de 24 pouces en métal, avec 100 pistes par face. Il pouvait enregistrer cinq mégaoctets de données et coûtait 10 000 $ par mégaoctet.

Le Hardware

Architecture Physique de l'Ordinateur

Générations d'ordinateurs de 1945 à nos jours

Troisième génération : (1965-1980) Premiers ordinateurs à circuits intégrés

✦ Circuits intégrés - Multiprogrammation - ✦

La troisième génération d'ordinateurs est celle des ordinateurs à circuit intégré qui ont été inventés par Jack Kilby en 1958. C'est à partir de cette date que l'utilisation de l'informatique a explosé.

Les premiers ordinateurs utilisant les circuits intégrés sont apparus en 1963. L'un de leurs premiers usages a été dans les systèmes embarqués, notamment par la NASA dans l'ordinateur de guidage d'Apollo,

Le Hardware

Architecture Physique de l'Ordinateur

Quatrième génération (1971 à la fin des années 1980)

✦ Évolution des ordinateurs personnels ✦

Le 15 novembre 1971, Intel dévoile le premier microprocesseur commercial, le 4004. Il a été développé pour Busicom, un constructeur japonais. Un microprocesseur regroupe la plupart des composants de calcul (horloge et mémoire mises à part pour des raisons techniques) sur un seul circuit. Couplé à un autre produit, la puce mémoire, le microprocesseur permet une diminution nouvelle des coûts. Le 4004 ne réalisait que 60 000 opérations par seconde

Le Hardware

Architecture Physique de l'Ordinateur

Quatrième génération (1971 à la fin des années 1980)
✦ Évolution des ordinateurs personnels ✦

En janvier 1973 est présenté le premier micro-ordinateur, le Micral conçu par François Gernelle de la société R2E dirigée par André Truong Trong Thi. Basé sur le premier microprocesseur, l'Intel 8008 8 bits, ses performances en font le plus petit ordinateur moderne de l'époque (500 kHz, mémoire RAM de 8 ko en version de base)

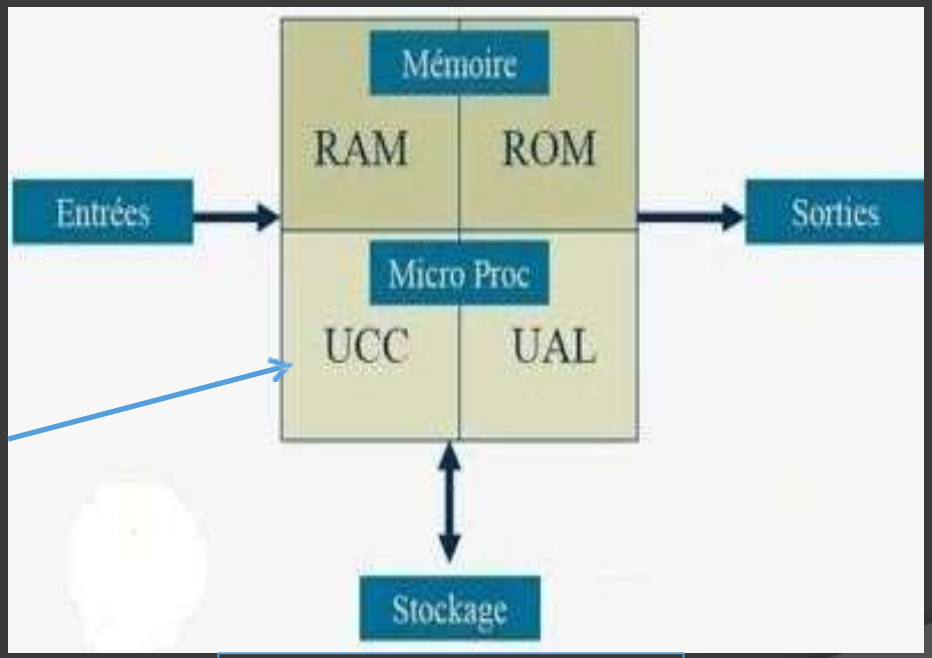
Le Hardware

Architecture Physique de l'Ordinateur

le processeur est le cœur du PC, chargé du traitement automatique de l'information, il doit dialoguer à sa périphérie à savoir :

mémoriser l'information

Réception des données



Fourniture des résultats

traitement Automatique de l'information

Les unités d'entrée, de sortie et de stockage sont appelés les périphériques externes.

enregistrement permanent de l'information

Le Hardware

Architecture Physique de l'Ordinateur

Afin de dialoguer avec son environnement, le processeur fournit des informations (signaux) pour faire les choix voulus.

Ces choix peuvent être :

- **Accès en lecture ou en écriture**
- **choix de la mémoire**
- **choix du périphérique concerné**
- **…..**

Ce choix est réalisé à l'aide des composants électroniques appelé chipset présenté comme jeu de circuits.

Le Hardware

Architecture Physique de l'Ordinateur

Le processeur est composé de :

1. **L'UAL : UnitéArithmétique et Logique qui calcule les opérations arithmétique et logiques sur des nombres entiers,**
2. **La FPU : Floating Point Unit qui traite les nombres flottants,**
3. **Le décaleur : qui décale les bits vers la gauche ou vers la droite ; c'est le spécialiste des divisions et multiplications par deux.**
4. **Les registres : petites mémoires rapides permettant le stockage des variables.**
5. **L'UCC : Unité de Contrôle et de Commande qui commande l'exécution de toutes les opérations à tous les niveaux (E/S, MC, UAL) et contrôle leurs déroulements**
6. **L'UGB : Unité de Gestion des Bus qui permet la gestion des informations entrantes et sortantes.**

Le Hardware

Architecture Physique de l'Ordinateur

Le Processeur dialogue avec le reste de l'ordinateur à travers le langage binaire.

Il ne comprend donc pas directement le code informatique que vous pourriez utiliser en suivant le cours de la programmation C.

Il existe un programme appelé compilateur (Dev-Cpp), qui convertit le langage de votre code en langage binaire.

Le Hardware

Architecture Physique de l'Ordinateur

2.La mémoire

On appelle " mémoire " tout composant électronique capable de stocker les données d'un programme ainsi que les résultats après exécution .

On distingue :

- **Les mémoires centrales (mémoires internes)**
- **Les mémoires secondaires (mémoires externes)**

Le Hardware

Architecture Physique de l'Ordinateur

1. **Mémoires centrales**

   - **La mémoire vive RAM ;**
   - **La mémoire morte ROM ;**
   - **La mémoire flash.**

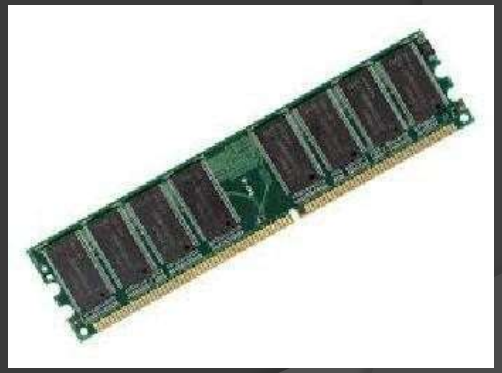
Le Hardware

Architecture Physique de l'Ordinateur

2.1 Mémoires centrales

- **La mémoire vive (RAM : Read & Access Memory) :**
  est une mémoire volatile (elle perd ses données lorsqu'on éteint l'ordinateur)

La RAM se présente sous forme de barrette caractérisée par son type (DDR, DDR2,..), sa capacité en octets(256, 512, 1000 Ko) et sa vitesse exprimée en fréquence ou en bande passante



La RAM est caractérisée aussi par sa rapidité d'échange avec le processeur lors du traitement de l'information.

La vitesse de la RAM dépend du processeur et du chipset

Le Hardware

Architecture Physique de l'Ordinateur

2.1 Mémoires centrales

1.

- **La mémoire morte (ROM : Read Only Memory) :**

En opposition avec la RAM, le contenu de la ROM est stocké de manière permanente, même si l'ordinateur est éteint.

Elle contient le programme nécessaire au démarrage du PC juste après la mise sous tension. Ce programme est appelé le Bios.

Sa fonction principale est réduite au déclenchement du PC.

Le Hardware

Architecture Physique de l'Ordinateur

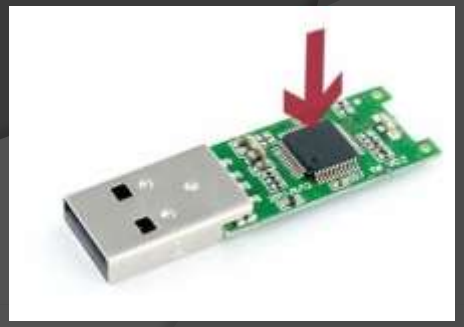
2.1 Mémoires centrales

- **La mémoire flash est un compromis entre la RAM et la ROM.**

La mémoire Flash est une mémoire non volatile, disponible en lecture et en écriture.

En contrepartie le temps d'accès des mémoires Flash est plus important que celui de la RAM.

La clé USB (flash disc) constituée principalement de mémoire flash, permet le stockage de données de manière amovible.



Le Hardware

Architecture Physique de l'Ordinateur

2.Les mémoires secondaires : appelées mémoires auxiliaires ou périphériques de stockage : sont des composants supplémentaires que l'on peut connecter à un ordinateur.

L'accès à ces mémoires se fait en lecture et/ou en écriture. Et le stockage se fait de manière permanente.

Il peut s'agir de :

- **Disques durs (internes et externes);**
- **CD-ROM, DVD ;**
- **USB ;**
- **…**

Le Hardware

Architecture Physique de l'Ordinateur

3. les Bus :

Il sont prévus pour connecter des cartes d'extension afin d'étendre ou de compléter les possibilités du PC.

Les bus possibles sur une carte mère sont :

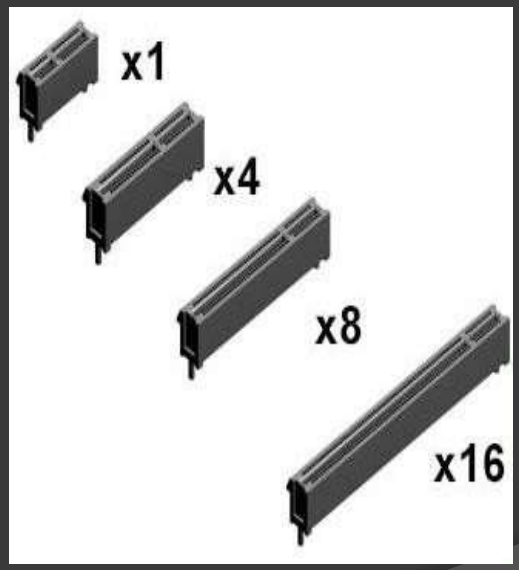
1.Bus AGP (Accelerated Graphic Port) conçu pour connecter une carte graphique;

2.Bus PCI (Peripheral Component Interconnected) symbolisés par des connecteurs blancs sont destinés à accueillir divers types de cartes (carte son, carte réseau, etc…)

Le Hardware

Architecture Physique de l'Ordinateur

3.3 Bus PCI-express : ont tendance à faire disparaitre les PCI et à remplacer les AGP; existe en 1x, 4x, 8x et 16x . Le 16x est dédié pour les cartes graphiques



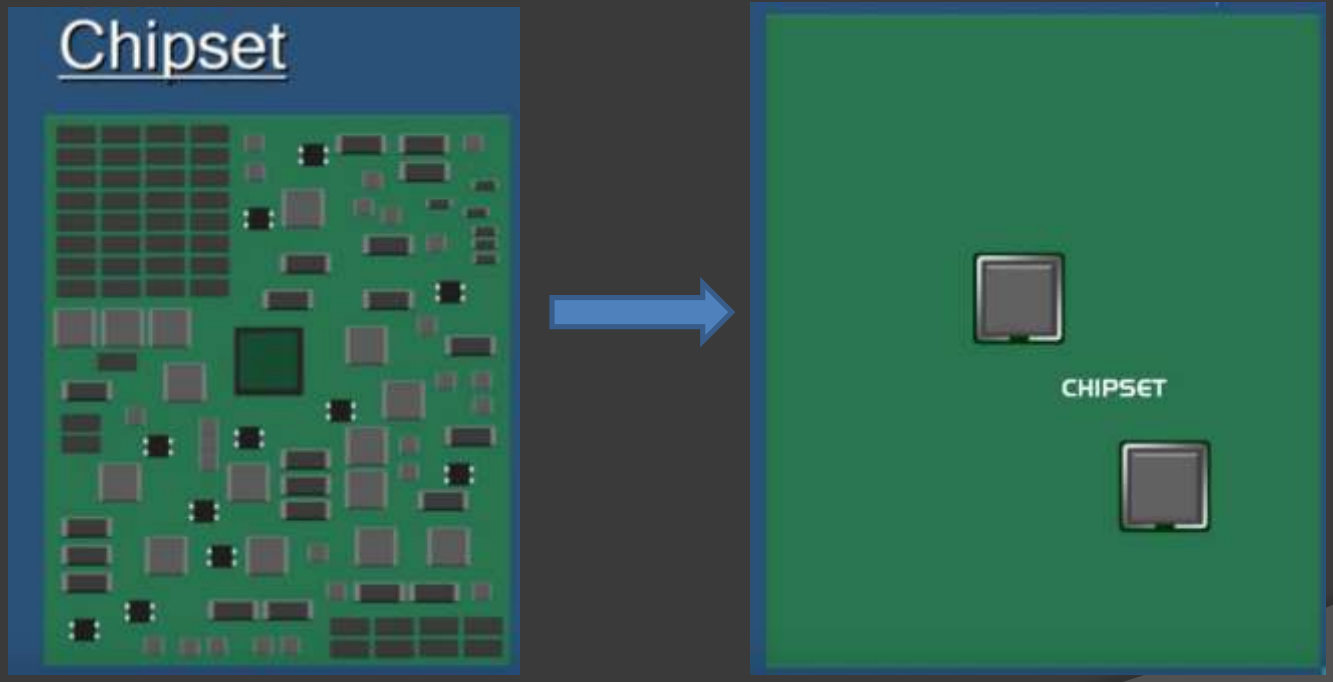
N.B. : avant de rajouter une carte d'extension dans votre PC, il faut savoir quels bus sont disponibles et de quels types

Le Hardware

Architecture Physique de l'Ordinateur

CHIPSET:

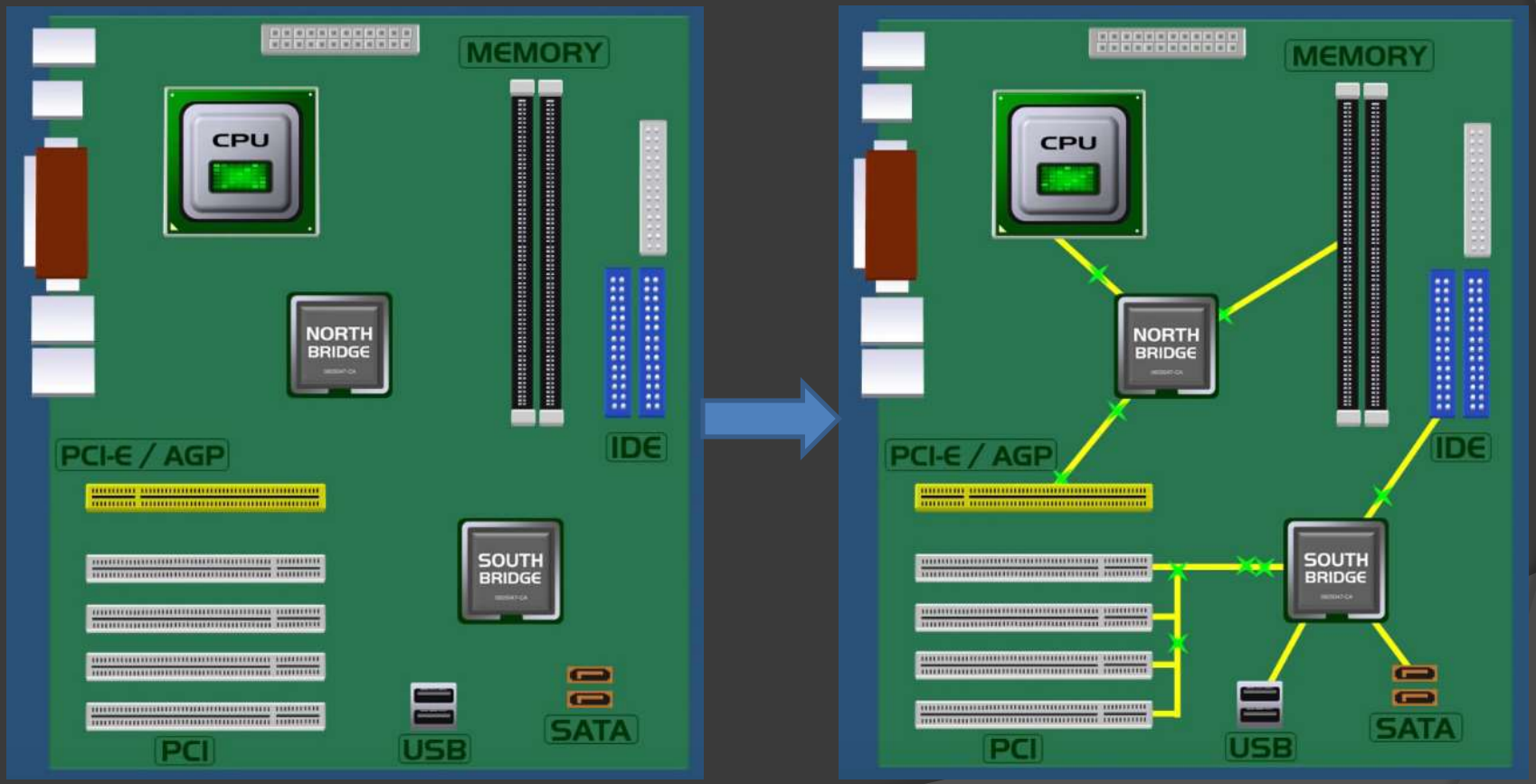
Son rôle est le contrôle du flux de donnée entre le CPU ,les bus ,la mémoire



Le Hardware

Architecture Physique de l'Ordinateur

CHIPSET: Vitesse des bus



Le Hardware

Architecture Physique de l'Ordinateur

4. Les Périphériques :

1. **Les cartes périphériques;**
2. **Les Périphériques internes :**
3. **Les Périphériques externes**

Le Hardware

Architecture Physique de l'Ordinateur

4. Les Périphériques :

1. Les cartes périphériques ou cartes d'extension :

Il s'agit des cartes insérables sur un bus adapté (PCI, AGP,….) ou sur une carte mère pour effectuer une ou plusieurs fonctions.

Les cartes les plus courantes sont :

- **Cartes graphiques ;**
- **Cartes d'acquisition analogique (vidéo, industrie,…..) ;**
- **Carte son pour générer les sons ;**
- **Carte réseau pour assurer la connexion entre plusieurs ordinateurs ;**
- **Carte d'interface (série, parallèle, disques)**
- **…….**

Le Hardware

Architecture Physique de l'Ordinateur

4.2 Périphériques internes :

Sont ceux qui sont intégrés dans le boitier et connectés directement à la carte mère tels que :

- **Les disques durs ;**
- **Les lecteurs CD/DVD ;**
- **Les graveurs .**

Ces Périphériques sont raccordés soit sur :

- **des ports IDE (Integrated Drive Electronics),**
- **des ports SATA**
- **des ports SCSI**
- **des ports USB**

Le Hardware

Architecture Physique de l'Ordinateur

3. Les Périphériques externes

Ils est possible de connecter une grande diversité de périphériques externes sur les ports d'entrée-sortie :

- **clavier ;**
- **souris ;**
- **appareil photo ;**
- **caméra numérique ;**
- **scanner ;**
- **écran ;**
- **imprimante ;**

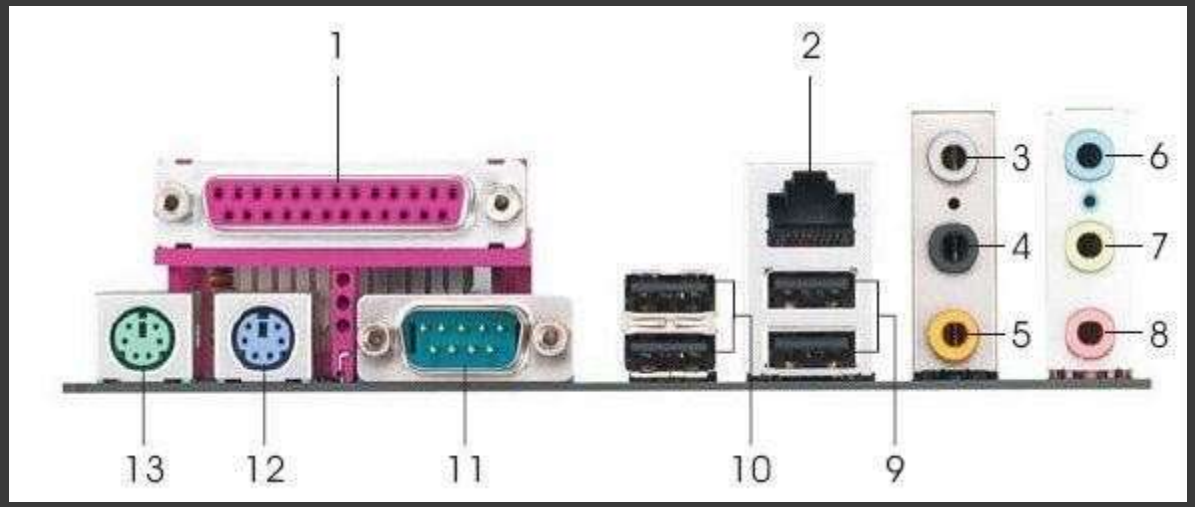
…

Le Hardware

Architecture Physique de l'Ordinateur

4.3 Les Périphériques externes

Exemple de connecteurs présents en partie arrière ou avant del'ordinateur sont:



- **Liaison parallèle (1)**
- **Liaison réseau (2)**
- **Liaisons audio (de 3 à 8)**
- **Liaisons USB (9 et 10)**
- **Liaisons pour souris et clavier (12 et 13)**

Le Software
La partie logiciel de l'Ordinateur

Les logiciels représentent un ensemble de programmes qui vont être exécutés par la machine pour réaliser des tâches précises.

Certains logiciels s'exécutent automatiquement, d'autres s'exécutent à volonté.

Il existe, principalement, deux types de logiciels :

1. **logiciels de base ;**
2. **logiciels d'application.**

Le Software
La partie logiciel de l'Ordinateur

1. **Les logiciels de base ou Système d'Exploitation :**
   sont des logiciels qui s'exécutent automatiquement lors du démarrage du PC.

   Le système d'exploitation est donc un logiciel qui permet de créer un environnement pour assurer la gestion de l'ordinateur et de ses périphériques

Les systèmes d'exploitation les plus connus sont :

- **MS-DOS,**
- **Windows,**
- **Linux et Unix.**

Le Software
La partie logiciel de l'Ordinateur

2. Les logiciels d'application :

1. **logiciels de programmation permettant le stockage des données de manière structurée pour les exploiter après.**

   tels que:
   - **Les gestionnaires de base de données (Access , SQL server, …),**
   - **Les compilateurs (Dev-C++, java ,python, …),**
   - **Les logiciels de dessin (Autocad, Adobe Illustrator….).**

2. **logiciels destinés à des utilisateurs qui ne sont pas nécessairement informaticiens tels que :**
   - **Jeux ;**
   - **World, Excel ;**
   - **Power point ;**
   - **Antivirus…**

Algorithmique

Généralités sur l'algorithmique

L'algorithmique désigne la science qui étudie les algorithmes et leurs applications en informatique.

Algorithme = Algorithmi nom du mathématicien arabe Al-Khawarizmi (780 – 850 après J.C.).

L'écriture algorithmique est universelle, elle ne dépend ni :

- **du langage dans lequel il sera implanté,**
- **de la machine qui exécutera le programme correspondant.**

Généralités sur l'algorithmique

Un algorithme permet de décrire dans l'ordre et en détail les différentes étapes à suivre pour résoudre un problème donné.

Exemple :

Appeler un ami depuis une cabine téléphonique.

1. **décrocher l'appareil ;**
2. **insérer les pièces de monnaie ;**
3. **composer le numéro ;**
4. **communiquer ;**
5. **raccrocher.**

Généralités sur l'algorithmique

Une bonne connaissance de l'algorithmique permet d'écrire des programmes exacts et efficaces.

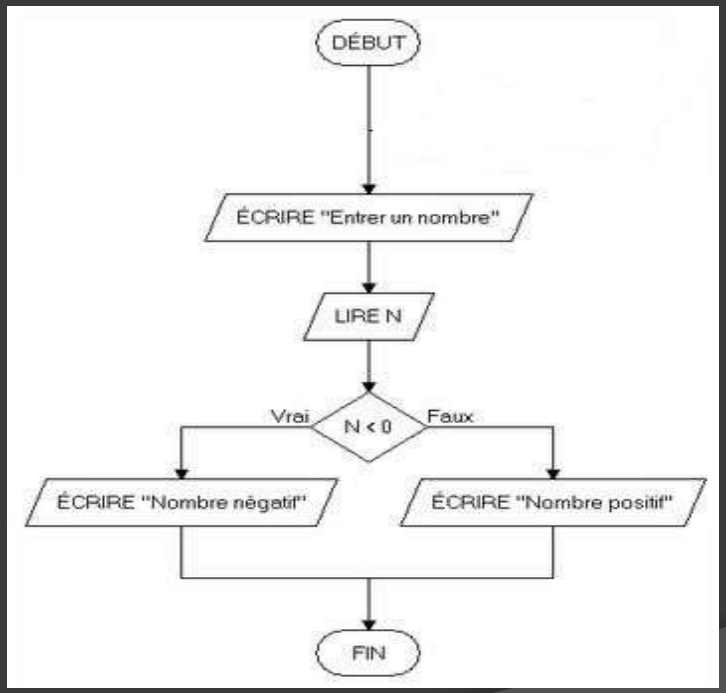
Il existe deux façons pour représenter un algorithme :

- **L'organigramme ;**
- **Le pseudo-code.**

Généralités sur l'algorithmique

L'organigramme :

c'est une représentation graphique (rectangle, losanges, …)utile pour décrire le fonctionnement d'instructions composant l'algorithme.



Exemple d'organigramme

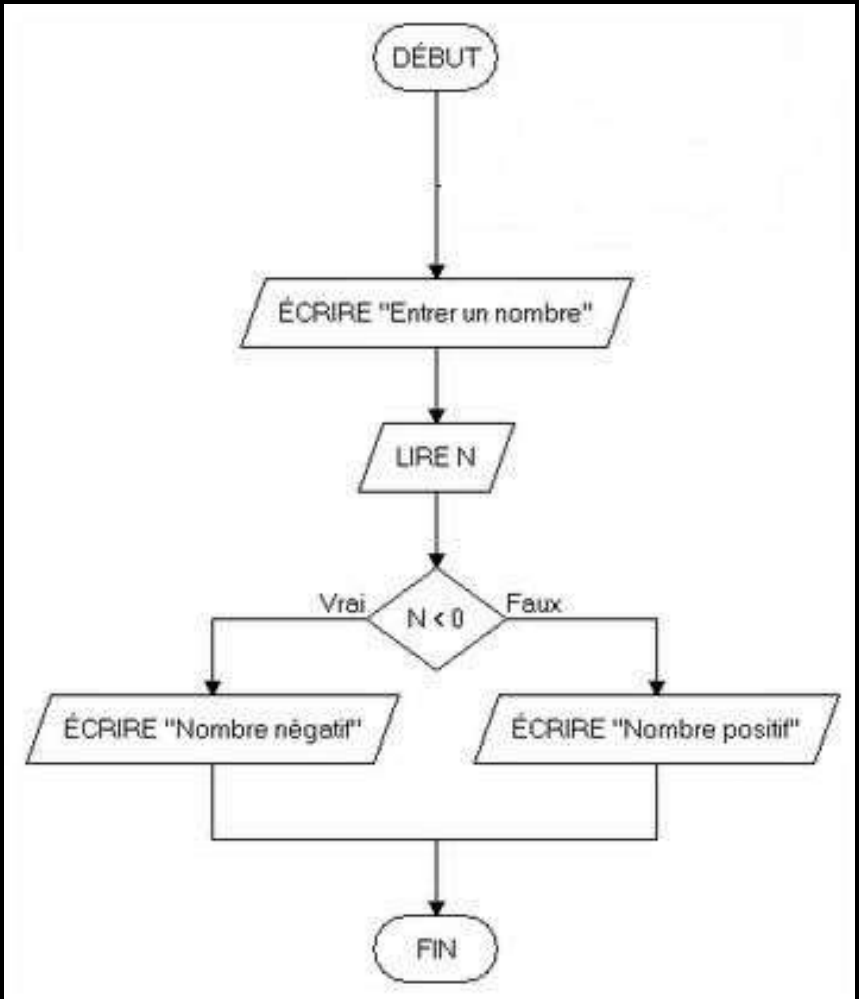
L'organigramme offre une vue d'ensemble de l'algorithme.

Généralités sur l'algorithmique

Exemple :

Les formes d'instructions de base d'un algorithme

L'organigramme est quasiment abandonné aujourd'hui



1- Marque le début et la fin

2- lecture / écriture :

3- test vrai ou faux :

Généralités sur l'algorithmique

Le pseudo-code : c'est une représentation textuelle .
il peut être facilement traduit en n'importe quel langage de programmation.

Exemple : Résoudre une équation du second degré

Le pseudo-code est plus pratique pour écrire un algorithme.
C'est une représentation largement utilisée

Généralités sur l'algorithmique

La structure générale d'un algorithme :

1. **Titre du Problème**
2. **Déclaration des variables : réservation de la mémoire**
3. **Début**
4. **Préparation du traitement : données nécessaires à la résolution du problème**
5. **Traitement : résolution pas à pas**
6. **Edition des résultats**
7. **FIN**

Généralités sur l'algorithmique

Exemple : Résoudre une équation du second degré

1-Titre : résoudre ax^2+bx+c;

2 Variables

a, b, c, delta, x1, x2 : réels

3 Début

4 Préparation du traitement

a←1, b←3, c←10

5 Traitement :

delta←(b*b)-(4*a*c)

si delta>0 ou nulle

x1←(-b + racine(delta))/(2*a)

x2←(-b - racine(delta))/(2*a)

6 Edition des résultats afficher (x1,x2)

Programmation C

Généralités sur la programmation

La programmation n'est que la traduction d'un algorithme en un langage simple que l'ordinateur peut transformer.

Le résultat donne un programme, qui peut ensuite être exécuté, pour effectuer les tâches souhaitées.

Il existe de nombreux langages dans lesquels on peut écrire des programmes :

- **C /C++ ;**
- **Basic ;**
- **Fortran ;**
- **Pascal ;**
- **Java ;**
- **Python …**

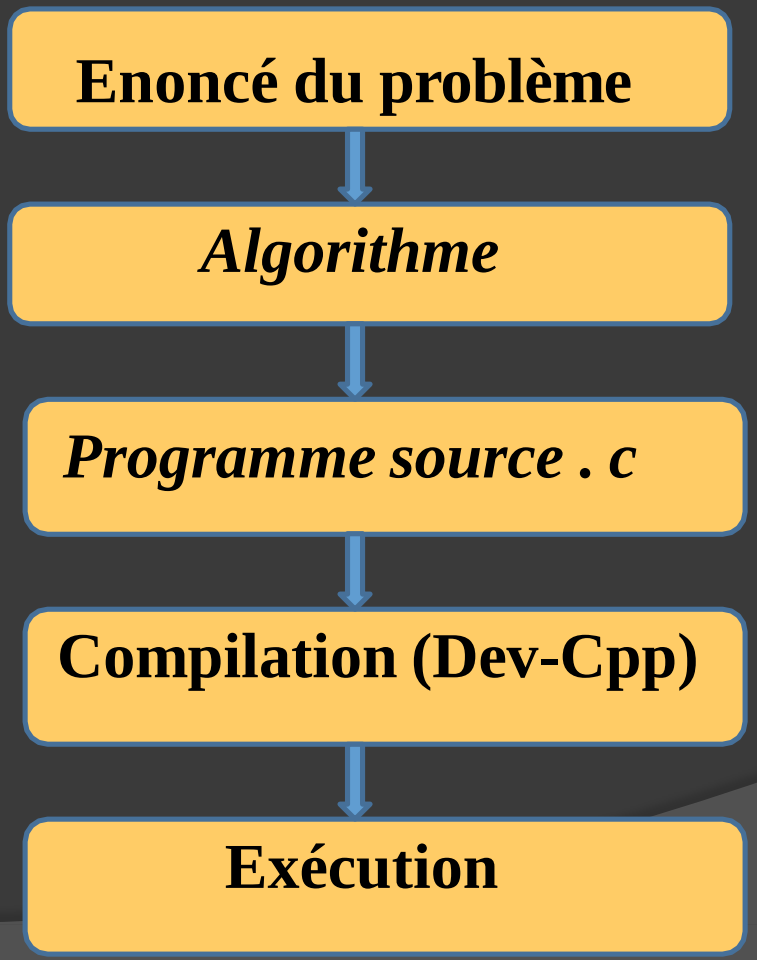
Généralités sur la programmation

Pourquoi le langage C ?

- **Le C est l'un des langages les plus répandus et les plus utilisés ;**
- **Le C est extrêmement performant car très "proche" des instructions du processeur ;**
- **Le C est très rapide en exécution ;**
- **Le C est très pratique pour les débutants ;**
- **Le C s'adresse aux scientifiques : (calcul mathématique, modélisation physique, pilotage automatique…).**

Différentes phases pour l'élaboration d'un programme

La réalisation d'un programme passe par les étapes suivantes :



Structure d'un programme en C

Un code source doit suivre la structure suivante :

1. **Déclaration des directive d'inclusion "# include " ;**
2. **Le programme doit être constitué par une ou plusieurs fonctions dont la fonction principale "main" ;**
3. **Chaque fonction est constituée par une ou plusieurs instructions. La fin d'une instruction est marquée par un point virgule (;).**
4. **Les instructions d'une même fonctions sont comprises entre deux accolades {…}.**
5. **Toutes les variables, utilisées dans le code source, doivent être déclarées avec un type.**
6. **L'appel des fonctions "getch ()" ou system ("pause") permet de figer l'écran d'affichage.**

Généralités sur la programmation

Structure générale d'un programme en C (code source)

1Déclaration des directives d'inclusion ;
2Déclaration des fonctions ;
3-{
4- Déclaration des variables ;

5- Instructions.

6- }

```
1  # include <stdio.h>
2  main()
3-{
4-   float   a, b, c, delta, x1, x2;
        a=1, b=3, c=10;
       delta=(b*b)-(4*a*c) ;
5-     if delta>0;
        x1=(-b+ sqrt (delta))/(2*a);
        x2=(-b- sqrt (delta))/(2*a);
        printf("  %f %f " ,x1,x2);
       getch();
6-}
```

Code source (.c) → Compilateur → Langage binaire

Dev-C++

Traduction d'un algorithme en langage C

Algorithme

```
Titre : Affichage
Début

    Ecrire("bonjour tout le monde" )


    Fin
```

Programme (.c)

```
# include <stdio.h>

 main()

{
    printf ("le double est %d :", d);


    getch();

}
```

Les variables en C

Les variables en langage C

Dans les langages de programmation, une variable sert à stocker une valeur.

Toute variable doit être déclarée avant d'être utilisée.

Elle est caractérisée par :

- **un nom (identificateur) ;**
- **un type (entier, réel, caractère, chaîne de caractères, …) ;**
- **une valeur (4, 3.14, 'A'…).**

Les variables en langage C
Noms des variables

Le nom d'une variable doit :

- **commencer par une lettre alphabétique Exemple : A1 et non pas 1A ;**

- **être constitué uniquement de lettres, de chiffres et du soulignement " _"**

  Exemple : GA_2017/18 et non pas GA-2017/18 ;

- **être différent des mots clés du langage C :**

  Exemple : prix et non pas int.

Les variables en langage C
Noms des variables

Les mots clés du langage C :

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	inline	long	register	restrict
return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned
void	volatile	_Bool	complexe	Class
Imaginary	while			

Conseil : pour la lisibilité du programme, choisir des noms significatifs qui décrivent les données manipulées (**Exemple:** somme, moyenne ect.).

Les variables en langage C
Types des variables

Le type d'une variable indique :

- **le nombre d'octets qu'elle occupe en mémoire,**
- **l'ensemble des valeurs qu'elle peut prendre.**

Les différents groupes de types :

- **types simples caractérisés par une seule valeur :**
  1. **type de données entières ;**
  2. **type à virgule flottante ;**
  3. **type pointeur ;**
  4. **type booléen.**
- **types complexes caractérisés par un ensemble de valeurs :**
  1. **tableaux ;**
  2. **structures ;**
  3. **classes.**

Les variables en langage C

Dans les langages de programmation une variable sert à stocker une valeur.

Toute variable doit être déclarée avant d'être utilisée.

déclarée

Elle est caractérisée par :

- **un nom (Identificateur) ;**
- **un type (entier, réel, caractère, chaîne de caractères, …) ;**
- **une valeur (4, 3.14, 'A'…).**

Les variables en langage C
Valeurs affectées

L'affectation est l'action qui consiste à donner une valeur à une variable.

En C, on utilise l'opérateur d'affectation "=" :

variable = valeur

Exemple 1 :

```
int i ;
i = 3 ;
int j = 5;
float   x = 9.5 , z = 0.9;
```

Les variables en langage C
Valeurs affectées

Remarques très importantes :

Remarque 1 :

En informatique, l'opérateur affectation "=" est différent de "l'égal" mathématique car l'affectation n'est pas commutative :

En mathématique : x = 3 est équivalent à 3 = x ;

En informatique : l'opérateur affectation met l'opérande de droite dans l'opérande de gauche :

x = 3 veut dire "mettre" la valeur 3 dans la variable x,

3 = x est illicite, car il veut dire "mettre" la variable x dans la valeur 3, ce qui est absurde ! .

Les variables en langage C
Valeurs affectées

Remarque 2 :

En mathématique : l'écriture x = x+3 est absurde ;

En informatique : x = x+3 veut dire que la nouvelle valeur de x est égale à l'ancienne valeur de x plus 3.

Exemple:

```
float  x=10 ;
x = x + 3;          // La nouvelle valeur de x est 13.00
x = x * 5 :         // x est 65.00
x = x / 10 :        // x est 6.500
```

Entrées-Sorties en C

Les entrées-sorties en C

Définition :

Les entrées-sorties désignent l'ensembles des fonctions qui permettent de saisir des valeurs à partir du clavier (les entrées), et de les afficher sur l'écran (les sorties).

Les entrées-sorties en langage C

En langage C :

- **Pour saisir au clavier, on utilise la fonction scanf.**
- **Pour afficher sur l'écran, on utilise la fonction printf.**

Pour utiliser ces fonctions , il faut inclure l'en-tête <stdio.h> en utilisant la directive d'inclusion :

```
# include <stdio.h>
```

Les sorties en langage C

Pour afficher le contenu d'une variable x de forme et de format précis, on utilise la fonction printf de syntaxe suivante :

printf(" %k", x) **k : clé de formatage**

- **Si x est un char, alors k=c :** **printf ("%c",x)**
- **Si x est un int, alors k=d :** **printf ("%d",x)**
- **Si x est un float, alors k=f :** **printf ("%f",x)**

Les sorties en langage C

Remarque 1 :

Pour afficher une phrase (par exemple : bonjour tout le monde):

```
printf ("bonjour tout le monde")
```

Remarque 2 :

Il est possible d'afficher plusieurs variables, en utilisant une seule instruction printf :

```
printf ("%d    %f", i, x)
```

affiche les valeurs de deux variables :
i (de type int) et x (de type float).

Les sorties en langage C

Remarque 3 :

On peut afficher une phrase contenant des variables en utilisant une seule instruction printf :

```
printf(" le double de %d est %d ", i, 2*i);
```

Remarque 4 :

Il existe des séquences d'échappement particulières utilisant des caractères spéciaux, tels que :

- **\n : passage à la ligne suivante ;**
- **\t : tabulation horizontale ;**
- **\a : émission d'un signal sonore.**

```
printf(" le double de %d  \t  %d \n ", i, 2*i);
printf(" le double de %f \t %f \a", x, 2*x);
```

Les entrées en langage C

Pour saisir des valeurs ou des caractères à partir du clavier, on utilise la fonction scanf de syntaxe :

```
scanf(" %k", &x)
```

- **Si x est un int, alors k=d : scanf("%d" ,&x)**
- **Si x est un float, alors k=f : scanf("%f" ,&x)**
- **Si x est un char, alors k=c : scanf("%c" ,&x)**

Remarque:

On peut saisir plusieurs valeurs et les affecter aux variables considérées, en faisant appel à la fonction scanf une seule fois.

```
scanf("%f  %d",&x,&y)     // x de type float et y de type int.
```

N.B. on n'utilise pas les séquences d'échappement (\n,\t,…) avec la fonction scanf

Les opérateurs

Les opérateurs

Les opérateurs sont des symboles (+, -, %, ++, <, …) qui permettent d'effectuer différentes opérations.

Les opérateurs sont très nombreux, on peut citer :

- **Les opérateurs arithmétiques ;**
- **Les opérateurs de comparaison ;**
- **Les opérateurs logiques ;**
- **Les opérateurs de bit ;**
- **Les opérateurs d'affectation ;**
- **L'opérateur conditionnel ;**
- **L'opérateur séquentiel**
- **L'opérateur de taille ;**
- **L'opérateur de cast ;**
- **…**

Opérateurs arithmétiques

Ces opérateurs permettent d'effectuer des calculs élémentaires sur leurs opérandes.

En plus des opérateurs connus : +, -, * et /, il existe aussi :

- **l'opérateur "-" (signe -) qui permet de changer le signe de l'opérande (l'opérateur signe est unaire).**
- **l'opérateur modulo % qui représente le reste entier d'une division de deux entiers ;**

Opérateurs arithmétiques

Exemple :

```
int x, y, z1,z2;
x = 3    y = 5 ;

z1= (x % y)
z2= (y % x)
```

Opérateurs de comparaison

Appelés aussi opérateurs relationnels, ils permettent de comparer les valeurs de leurs opérandes :
Le résultat de la comparaison est de type booléen c'est-à-dire soit 1 (vrai) soit 0 (faux).

- **x == y : vaut 1 si x est égal à y (vaut 0 sinon).**
- **x != y : vaut 1 si x est différent de y (vaut 0 sinon).**
- **x <= y : vaut 1 si x est inférieur ou égal à y (vaut 0 sinon).**
- **x >=y : vaut 1 si x est supérieur ou égal à y (vaut 0 sinon).**
- **x < y : vaut 1 si x est inférieur à y (vaut 0 sinon).**
- **x > y : vaut 1 si x est supérieur à y (vaut 0 sinon).**

Opérateurs de comparaison

Exemple :

```
int  x, y z1,z2,z3,z4;
x = 3  ,  y = 5  ;

z1 = (x==y);
z2 = (x < y);
z3 = (x!=y);
z4 = (x<=y);
```

Opérateurs logiques

Les opérateurs logiques effectuent les opérations de logique.

Trois opérateurs sont proposés par C :

- **L'opérateur logique "&&" représente la connexion ET.**
  Le résultat vaut 1 si les deux opérandes sont vraies et vaut 0 dans les autres cas ;

- **L'opérateur logique "||" représente la connexion OU.**
  Le résultat vaut 0 si les deux opérandes sont fausses et 1 dans les autres cas ;

- **L'opérateur logique "!" représente la connexion Non.**
  Le résultat vaut 1 si l'opérande est fausse et vaut 0 si l'opérande est vraie.

Opérateurs logiques

Remarque :

Le résultat des opérations logiques est de type booléen soit 1 (vrai) soit 0 (faux).

Exemples :

int x =3;

1- r=((x<7) && (x>2))

2- r= ((x>7) || (x<2))

3- r=!(x>0)

Opérateurs de bits

Les opérateurs de bits permettent de travailler directement sur le code binaire d'une valeur.

Six opérateurs sont proposés par C :

- **L'opérateur de bit "&" représente le ET.**
- **L'opérateur de bit "|" représente le OU inclusif .**
- **L'opérateur de bit "^" représente le OU exclusif .**
- **L'opérateur de bit "~" représente le complément à 1.**
- **L'opérateur de bit "<<" représente le décalage à gauche.**
- **L'opérateur de bit ">>" représente le décalage à droite**

Remarque :

Ces opérateurs ne fonctionnent que sur les types entiers

Opérateurs d'affectation

Huit opérateurs disponibles en C :

- **l'opérateur "++" désigne l'incrémentation de 1 ;**
- **l'opérateur "--" désigne la décrémentation de 1 ;**
- **l'opérateur "=" désigne l'affectation simple ;**
- **l'opérateur "+=" désigne l'affectation avec addition ;**
- **l'opérateur "-=" désigne l'affectation avec soustraction ;**
- **l'opérateur "*=" désigne l'affectation avec multiplication ;**
- **l'opérateur "/=" désigne l'affectation avec division ;**
- **l'opérateur "%=" désigne l'affectation avec opération de reste.**

Opérateurs d'affectation

Exemple :

float x =3;

x++;

x--;

x-=5 ➔ x = x-5

x*=5 ➔ x = x*5

x /= 5 ➔ x = x/5

x % =2 ➔ x = x % 2

Opérateurs d'affectation

Remarque sur les préfixes et post-fixes

Dans une opération qui contient en même temps l'opérateur "=" et l'un des opérateurs : ++, --, le résultat obtenu diffère suivant que l'opérateur (++ ou --)est préfixe ou postfixe, c'est-à-dire suivant que l'opérateur est avant ou après l'opérande.

conclusion

Le résultat de l'opération y =x++ est différent de celui de y=++x.

Dans y = x++ : la variable x est affectée à la variable y, ensuite x est incrémenté, soit :

```
y=x ;
x++ ;
```

ensuite

Opérateurs d'affectation

Dans y = ++x : on commence par faire augmenter la valeur de x. Ensuite, cette nouvelle valeur est affectée à y, soit :

```
x++ ;
y = x ;
```

Exemple :

```
float  x=3;
float y;
 y = x++ ;       //  y = 3 et  x  =  4

float x=3;
float y;
 y = ++x ;        // y = 4 et  x = 4
```

Opérateur conditionnel

L'opérateur " ? : " utilise trois opérandes , sous la forme suivante :

opérande 1 ? opérande 2 : opérande 3

si l'opérande 1 est vraie, le résultat de l'opération prend la valeur de l'opérande 2 si non elle prend la valeur de l'opérande 3.

Exemple :

```
int x =10, y =5, r;
L'expression   r=(x!=y) ? x : y                 //r = x ; r =10
alors que l'expression r=(x<=y) ? x : y         //r= y ; r =5.
```

Opérateur séquentiel

L'opérateur séquentiel ou opérateur virgule "," permet de rassembler de manière syntaxique deux expressions en une seule.

Exemple :

L'instruction :

```
int x  ;
int y ;
```

est équivalente à :

```
int x , y ;
```

Opérateur de dimension

L'opérateur de dimension "sizeof" calcule l'occupation mémoire en octets requise par une variable ou par un type de données.

Exemple :

```
1  int x ;
   printf(" %d" , sizeof(x));
2  printf(" %d" , sizeof(int));
```

Dans les deux cas, le résultat affiché est 4 qui indique la mémoire occupée par un int.

Priorités des opérateurs

Exemple

float x=5, y=10, z=2;

1- x + y/z =

2- ++x * y =

3- x * -- y =

4- x *= ++y

Structures de contrôle
de flux en C

Structure de contrôle de flux en C

Les structures de contrôle de flux sont des commandes qui contrôlent l'ordre dans lequel les différentes instructions d'un programme sont exécutées

1. **Les structuresalternatives ou de test :**
   rendent l'exécution de certaines instructions dépendantes d'une certaine condition.
2. **Les structures répétitives ou itérations ou boucles :**
   répètent certaines phases de traitement.
3. **Les structures de branchement :**
   fournissent des possibilités diverses de branchement inconditionnels.

Structure de contrôle de flux

Structures alternatives (ou detest)

Les structures alternatives permettent d'exécuter certaines instructions sous certaines conditions.
Le langage c propose trois instructions :

- **if (Si)**
- **if-else (Si-sinon)**
- **switch (Selon)**

Structure de contrôle de flux
Instructions alternatives

Instruction if (si)

Il ne permet d'exécuter des instructions que si une certaine condition est satisfaite.

La syntaxe de l'instruction "if" est :

```
if( Test vrai ) instruction ;
```

ou encore :

```
if( Test vrai   )
{
        bloc d'instructions;
 }
```

Structure de contrôle de flux
Instructions alternatives

Exemple 1 :

```
float x;
 scanf(" %f",&x);
 if(x==0) printf("x est  nul") ;
```

Exemple 2 :

```
if (x<0)
    {
        printf("x est négatif") ;
        printf(" donner une valeur positive ") ;
        scanf ("%d",&x);
     }
```

Structure de contrôle de flux
Instructions alternatives

Test avec alternative if- else (si-sinon)

L'instruction if- else permet d'exécuter des instructions si une certaine condition est satisfaite et d'exécuter d'autres instructions sinon.

La syntaxe de l'instruction if- else est :

```
if ( Test vrai )   instruction 1 ;
else instruction 2 ;
```

ou encore :

```
if ( Test vrai )
     {        bloc 1 d'instructions      }
else
     {        bloc 2 d'instructions      }
```

Structure de contrôle de flux

Instructions alternatives

Test avec alternative if-else

Exemple :

```
#include <stdio.h>
main( )
{
     int  x, y ;
      printf("donner x et y " );
      scanf ("%d%d", &x, &y);
     if (x>y) printf("x est plus grand que y ") ;
     else printf(" y est plus grand que x " ) ;
}
```

Structure de contrôle de flux

Instructions alternatives

Test avec alternative if else-if else (si sinon-si sinon)

La branche else peut contenir d'autres instructions conditionnelles.

la syntaxe est :

```
if( Test1 vrai ) instruction 1 ;
else if ( Test2 vrai )   instruction 2 ;
else    instruction3;
```

Ou

```
if( Test1 vrai ) {   bloc 1 d'instructions ;  }
else if ( Test2 vrai )   { bloc 2 d'instructions ;  }
else   {  bloc3 instructions;          }
```

Structure de contrôle de flux

Instructions alternatives

Test avec alternative if else-if else

Exemple :

```
#include <stdio.h>
main( )
{
    int x, y ;
    printf("donner x et y " );
    scanf ("%d%d", &x, &y);

    if (x>y) printf("x est plus grand que y ") ;
    else if (x<y) printf(" y est plus grand que x" ) ;
    else printf (" x égale à y" )
}
```

Structure de contrôle de flux

Instructions alternatives

Remarques :

- **La condition doit être entre parenthèse ;**
- **Il ne faut jamais mettre un point virgule après la condition ;**
- **S'il n'y a qu'une seule instruction, les accolades ne sont pas nécessaires ;**
- **Il est possible de définir plusieurs conditions en utilisant les opérateurs logiques: && et ||.**

Exemples :

```
if((i > 0) && (i % 2==0 )) printf (" %d",i);

if((x < 0) || (y < 0)) printf (" x*y est négatif");
```

SUITE PARTIE (2)