1. **Architecture de Von Neumann**
2. **L’algorithmique**
3. **Structure de données**

1. Architecture de Von Neumann

Un **ordinateur** est un ensemble de circuits électroniques permettant de manipuler des informations qu'on appelle des données et capable de faire "tourner" des programmes, c'est-à-dire une suite ou séquence d'instructions programmées à l'avance et qu'il va dérouler du début à la fin dans le but d'obtenir des résultats. Pour comprendre comment un ordinateur peut dérouler un programme, il faut étudier un peu plus en détail son fonctionnement.

C'est **Von Neumann** qui a défini en 1944 l'architecture des ordinateurs modernes encore largement utilisée aujourd'hui (avec des variantes cependant). L'architecture de Von Neumann décompose l'ordinateur en quatre parties distinctes:

- L'Unité Arithmétique et Logique
- La mémoire
- L'Unité de Contrôle (UC)
- Les Entrées/Sorties E/S.

1. **L’Unité Arithmétique et Logique UAL** (ALU en anglais) est l’organe de l’ordinateur qui exécute les calculs: additions, soustractions, multiplications, divisions, modulos, gestion des signes (positif, négatif), opérations logiques (booléenne), comparaisons, parfois rotations et décalages de valeurs (toujours dans le cadre d’une logique booléenne). Il existe des UAL spécialisées dans les nombres à virgule flottante, d’autres dans des traitements complexes comme les logarithmes, les inversions, les racines, les vecteurs, les calculs trigonométriques, etc. Certaines documentations lui rajoutent quelques registres (petites cases mémoires intégrées à l’UAL) et lui donnent le nom de processeur (CPU).

2. **L‘Unité de Contrôle UC** (CU en anglais), à ne pas confondre avec Unité Centrale, contrôle le séquençage des opérations, autrement dit le déroulement du programme. Elle prend ses instructions dans la mémoire et donne ses ordres à l’UAL. Les résultats retournés peuvent influer sur le séquençage. L’UC passe alors à l’instruction suivante ou à une autre instruction telle que le programme lui ordonne d’effectuer.

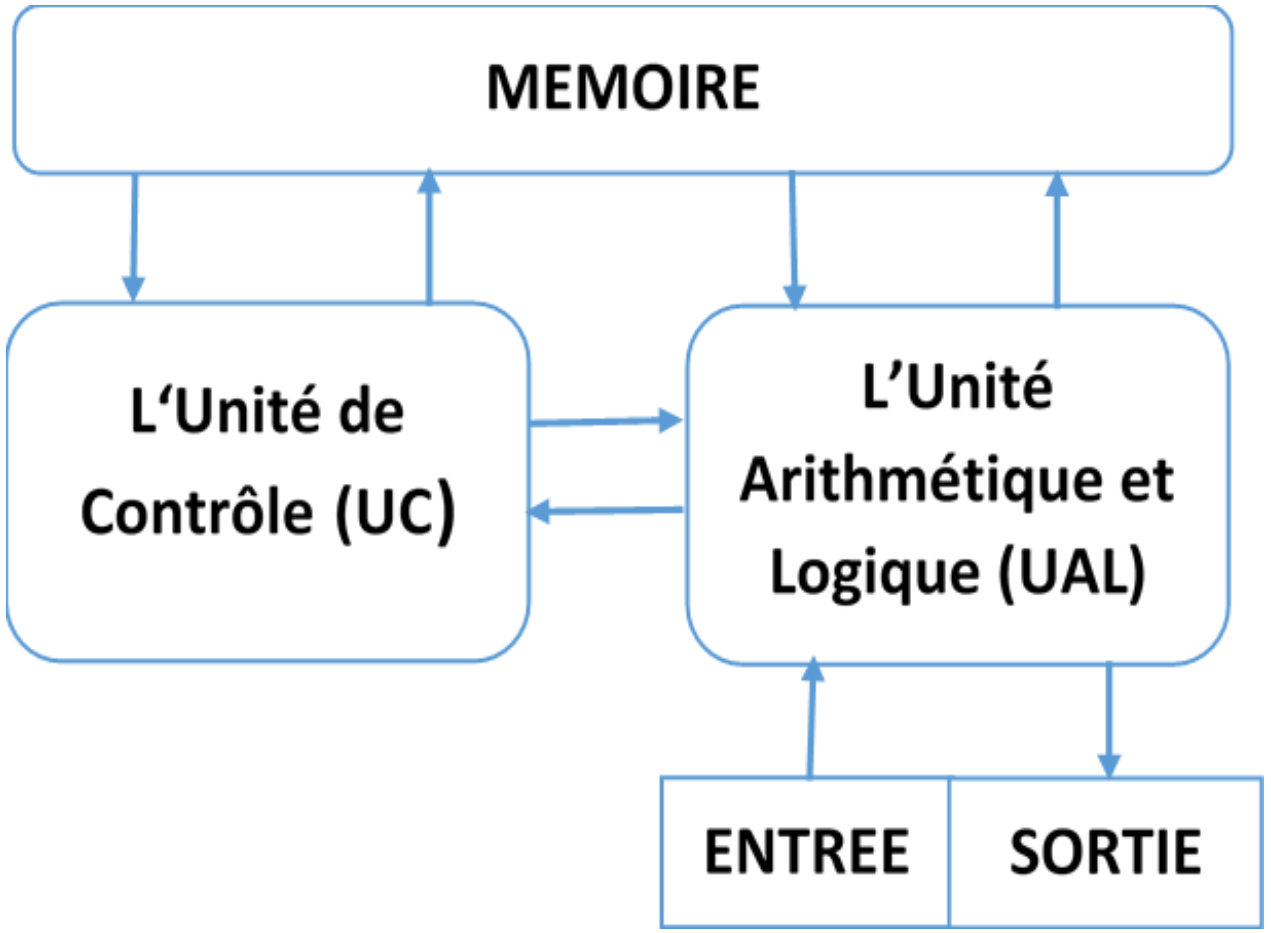
3. **La mémoire** peut être décrite comme une suite de petites cases numérotées, chaque case pouvant contenir une information. Cette information peut être une instruction ou un morceau d'instruction du programme (une instruction peut occuper plusieurs cases) ou une donnée (nombre, caractère, ou morceau de ceux-ci). C'est l'UC qui a comme rôle central de contrôler l'accès à la mémoire pour le programme et les données. **Chaque numéro de case est appelé une adresse**. Pour accéder à la mémoire, il suffit de connaître son adresse. Les instructions du programme pour l'UC et les données pour l'UAL sont placées dans des zones différentes de la même mémoire physique.

4. **Les Entrées/Sorties E/S** (I/O en anglais) permettent de communiquer avec le monde extérieur et donc vous : ce peut être un clavier pour entrer les données, et un écran pour afficher les résultats. Il permet à l'ordinateur d'être interactif.

Pour résumer : **l'architecture de Von Neumann** est simple à comprendre et répartit les fonctionnalités d'un ordinateur en quatre entités logiques. Ces entités logiques se retrouvent pour deux d'entre elles (UC et UAL) dans le microprocesseur. Les autres et les composants additionnels se retrouvent sur la carte mère ou sur les cartes d'extension (la mémoire n'est plus soudée sur une carte mère mais fournie sous forme de carte additionnelle appelée barrette, le contrôleur E/S graphique est sur une carte graphique additionnelle reliée à un bus PCI, AGP ou PCI/E). Les programmes sont exécutés par le microprocesseur qui est aidé (dans le sens où celui-ci accède aux fonctions proposées) par divers contrôleurs.

Note : les microprocesseurs actuels sont très complexes. Il n'est pas rare de trouver au sein de ceux-ci plusieurs UAL pour accélérer les traitements. De même on trouve souvent une mémoire intermédiaire appelée mémoire cache ou antémémoire, celle-ci étant souvent spécialisée : une mémoire cache pour les instructions et une autre pour les données.

Architecture de Von Neumann



Architecture de Von Neumann

L’algorithmique

Algorithme

Un algorithme est une séquence d'***instruction***s ***non ambiguës*** pour résoudre un problème, c'est-à-dire pour obtenir un résultat requis pour toute entrée légitime dans un laps de temps fini. Cette définition peut être illustrée par un schéma simple (figure 1.1).

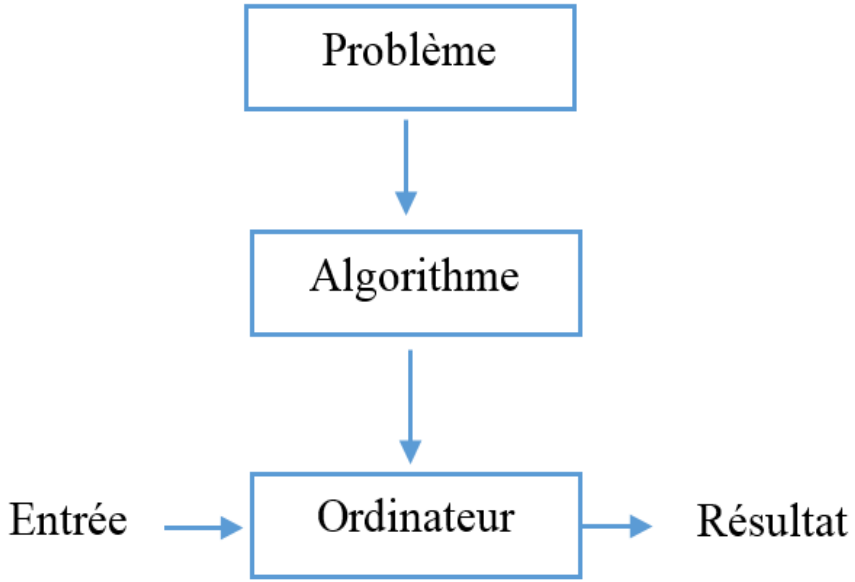


Figure 1.1 La notion d’algorithme

Un algorithme doit respecter les règles suivantes :

1. Il est defini sans ambiguïté

2. Il se termine après un nombre fini operations

3. Toutes les opérations en doivent pouvoir être effectuées par un homme utilisant des moyens manuels

4. Il manipule des objets qui doivent être définis de façon très précise.

L'**algorithmique**, l'art d'écrire des algorithmes, permet de se focaliser sur la procédure de résolution du problème sans avoir à se soucier des spécificités d'un langage particulier.

Les objets

Les **objets d'entrée** : données fournies a l'algorithme

Les **objets de sortie** : résultats produits par l'algorithme

Les **objets internes** : ce sont les objets servant aux manipulations internes de l'algorithme

Le traitement d'un objet concerne la valeur de cet objet.

Constante

Un objet dont la valeur ne peut pas être modifiée dans un algorithme est une constante.

Variable

Un objet dont la valeur peut être modifiée dans un algorithme est une variable.

Les Caractéristiques d’un objet

Un objet est parfaitement défini si nous connaissons ces trois caractéristiques :

- ✓ son **identificateur** (nom de l'objet) : il est représente par une vie quelconque de caractères alphanumériques commençant obligatoirement par une lettre.
- ✓ sa **valeur** (constantes ou variables)
- ✓ son **type** : le type d'un objet est défini par un ensemble de valeurs constantes et par l'ensemble des opérations que nous pouvons appliquer a ces constantes.
  - Reel ;
  - Entier
  - Booleen
  - Caracteres
  - Chaine de caractères

Tous les objets utilisés dans un algorithme doivent être déclares pour cela nous déterminons quels sont les objets de valeurs constantes et variables.

Les constantes :

CONST nom_objet = valeur

Les variables :

VAR nom_objet : type

Remarque :

- si plusieurs variables sont de même type nous pouvons les regrouper sur une même ligne ;
- les constants sont déclares en premier ;
- les termes réserves CONST et VAR ne figure qu'une seule fois dans l'algorithme ;

Exemples :

```
CONST max = 50
      min  = 10
VAR jour, mois, année : chaine de caractères
```

Formalisme d'un algorithme

Un algorithme sera defini par :

- un nom ;
- déclarations des variables et des constantes ;
- les actions constituant le traitement a exécute seront délimitées par les termes : DEBUT et FIN ;

Remarque : afin de permettre une plus grande visibilité, il faudra utiliser des commentaires délimités par les sigles
/*commentaires*/.

```
Programme nom_pg
/*Declarations des constantes*/
CONST nom_const = valeur
VAR nom_var : type
 Debut
   | Action 1
   | Action 2
   | etc.
 Fin
```

Comment trouver un algorithme ?

Question à see poser :

- ✓ quelles sont les données dont on dispose ?
- ✓ quelles sont les résultats que l'on doit obtenir ?

Comment obtenir ces resultats :

- ✓ appliquer les règles de gestion ;
- ✓ traite à plusieurs exemples significatifs (attention aux cas particuliers) ;
- ✓ écrire les actions nécessaires pour le traitement.

La validité d'un algorithme

Pour être valable un algorithme doit répondre aux critères suivants :

- ✓ le résultat donne doit être le résultat attendu, c'est-a-dire que le programme doit donner le bon resultat dans tous les cas de figures, il faut donc procéder a des tests par l'intermediaire de différents essais ;
- ✓ l'algorithme doit s'arreter une fois sa tache accomplie ;
- ✓ le résultat doit être donne dans un temps acceptable ;
- ✓ l'algorithme doit gérer au mieux la mémoire de l'ordinateur.

Structure de données

Definition

Dans un sense general, **une structure de données est toute représentation de données et ses operations associées**. Meme un nombre entier ou un nombre à virgule flottant memorisé dans l'ordinateur peut etre vu comme une structure de donnees simple. **Le plus souvent, une structure de données est une organization ou une structuration pour une collection d'éléments de donnees**. Une liste triée de nombres entiers memoriser dans un tableau à simple dimension est un exemple de telle organization ou structuration.

Comment choisir une structure de données ?

Il est evident qu'on écrit des programmes pour résourdre des problèmes. Cependant, il est crucial de garder cette evidence a l'esprit lors du choix d'une structure de données pour resourdre un problem particulier.

Pour choisir une structure de données pour résourdre un problem, on doit suivre les étapes suivantes:

1. Analyser le problème à résourdre pour determiner les opérations de bases qui doivent etre prises en compte. Exemple d'operations de bases:
   - L' ajout d'un élément de donnée dans la structure de données
   - La suppression d'un élément de données de la structure de données
   - La recherché d'un élément de donné specifié
2. Quantifier les contraintes de ressources pour chaque operations de base.

3. Choisir la structure de données qui correspond au mieux à ces exigences