RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOSTEFA BENBOULAID BATNA 2

FACULTE DE TECHNOLOGIE

DEPARTEMENT SOCLE COMMUN EN SCIENCES ET TECHNOLOGIES

COURS INFORMATIQUE 1

Première année (S1) socle commun en sciences et technologies

Dr. Salhi Hicham

Maître de Conférences classe A

Introduction générale

L'informatique est une discipline qui englobe l'art, la technique, et la science. Elle utilise l'ordinateur comme outil pour définir des algorithmes, résoudre des problèmes complexes, et extraire des connaissances utiles à partir de données désorganisées. Elle offre une approche créative, pratique et scientifique pour aborder divers domaines et défis.

Le contenu de ce document pédagogique est abordé selon une approche classique. Dans le premier chapitre, on se limite à la présentation des concepts de base qui sont largement utilisés dans l'étude de l'informatique, talque systèmes de codage des informations, principe de fonctionnement d'un ordinateur et Partie matériel et système d'un ordinateur.

Dans le deuxième chapitre du document, l'accent est mis sur l'acquisition de connaissances fondamentales en informatique. Cela inclut la compréhension des algorithmes, de la programmation de base, et structures de données. Ces connaissances sont essentielles pour établir une base solide en informatique et préparent les apprenants à aborder des sujets plus avancés dans le domaine. Nous avons couvert divers aspects, parmi lesquels les points les plus significatifs : Représentation en organigramme, Structure d'un programme, démarche et analyse d'un problème, Structure des données et Les opérateurs.

Chapitre I : Introduction à l'informatique

I.1 Définition de l'informatique

L'informatique est le domaine d'étude et de pratique qui englobe la manipulation, la gestion, et le traitement de l'information à l'aide de l'ordinateur et de systèmes informatiques. Il englobe la conception, le développement, la programmation, l'analyse, et l'utilisation de logiciels et de matériel informatiques pour résoudre des problèmes, automatiser des tâches, stocker et récupérer des données, ainsi que pour la communication et la gestion de l'information. En essence, l'informatique consiste à traiter des données sous forme numérique en suivant des instructions logiques appelées algorithmes pour atteindre des objectifs spécifiques. Ce domaine interdisciplinaire est omniprésent dans la société moderne, touchant de nombreux aspects de la vie quotidienne, de la science à l'industrie, en passant par les loisirs et la recherche

I .2 Évolution de l'informatique et des ordinateurs

L'évolution de l'informatique et des ordinateurs est un voyage fascinant qui a traversé plusieurs décennies, marqué par des progrès significatifs en matière de technologie, d'architecture informatique et de puissance de calcul. Voici une synthèse des étapes clés de cette évolution :

- ***Les débuts (années 1940-1950) :*** Les premiers ordinateurs étaient des machines énormes et coûteuses, tels que l'ENIAC (Electronic Numerical Integrator and Computer) et l'UNIVAC (Universal Automatic Computer). Ils étaient principalement utilisés à des fins militaires et scientifiques, notamment pour des calculs complexes liés à la Seconde Guerre mondiale.
- ***L'ère des ordinateurs centraux (années 1950-1960) :*** Les ordinateurs centraux, également appelés mainframes, sont devenus populaires. Ils étaient utilisés par les entreprises et les institutions pour la gestion des données et des transactions. Le langage de programmation COBOL a été créé, ce qui a facilité le développement de logiciels d'entreprise.
- **L'ère de la mini-informatique (années 1960-1970) :** Les ordinateurs plus petits, appelés mini-ordinateurs, sont apparus. Ils étaient plus abordables et adaptés à un usage dans les laboratoires de recherche et les universités. L'émergence du système d'exploitation UNIX a eu un impact significatif sur le développement logiciel.
- ***L'ère de la micro-informatique (années 1970-1980) :*** L'avènement des microprocesseurs a permis la création des premiers ordinateurs personnels (PC). Des entreprises telles qu'Apple et IBM ont introduit des ordinateurs personnels, popularisant ainsi l'informatique domestique et professionnelle.
- **L'ère d'Internet et des ordinateurs personnels (années 1980-1990) :** La diffusion d'Internet a radicalement transformé la façon dont les gens communiquent et accèdent à l'information. Les ordinateurs personnels se sont généralisés, avec l'apparition de systèmes d'exploitation conviviaux tels que Windows.
- **L'ère de la mobilité (années 2000-2010):** L'avènement des smartphones et des tablettes a révolutionné la manière dont nous interagissons avec la technologie. Les processeurs plus

puissants et les réseaux sans fil ont ouvert la voie à de nouvelles applications et services mobiles.

- **L'ère du cloud computing (années 2010-présent) :** Le cloud computing a transformé la façon dont les données sont stockées et gérées, permettant un accès à distance aux ressources informatiques. L'intelligence artificielle, l'apprentissage automatique et l'analyse des données ont connu une croissance exponentielle.
- **L'avenir :** L'évolution de l'informatique se poursuit avec des avancées telles que la réalité virtuelle/augmentée, l'informatique quantique, l'automatisation avancée et l'Internet des objets (IoT). Les ordinateurs continuent de devenir plus puissants, compacts et intégrés dans notre vie quotidienne.

I .3 Les systèmes de codage des informations :

Les systèmes de codage des informations sont des méthodes ou des règles utilisées pour représenter, stocker, transmettre et décoder des données et de l'information. Ces systèmes sont essentiels dans le domaine de l'informatique, des télécommunications, de l'électronique et de nombreux autres domaines où la manipulation de l'information est cruciale.

I .3.1 Système de codage décimal :

Le système décimal est celui que nous utilisons au quotidien, basé sur dix chiffres (0-9). Il est utilisé pour représenter des nombres naturels en base dix.

Exemple

Soit un nombre décimal F = 2348. Ce nombre est la somme de 8 unités, 4 dizaines, 3 centaines et 2 milliers. Nous pouvons écrire

F = (2 x 1000) + (3 x 100) + (4 x 10) + (8 x 1)

$2348 = (2 \times 10^3) + (3 \times 10^2) + (4 \times 10^1) + (8 \times 10^0)$

10 représente la base et les puissances de 0 à 3 le rang de chaque chiffre. Quel que soit la base, le chiffre de droite est celui des unités. Celui de gauche est celui qui a le poids le plus élevé. Cette écriture s'appelle forme polynomiale.

I .3.2 Système de codage binaire :

Le système binaire est la base de tous les systèmes informatiques. Il utilise deux symboles, généralement 0 et 1, pour représenter l'information. Chaque chiffre binaire est appelé un "bit" (contraction de "binary digit").

Un octet est une unité d'information composée de 8 bits. Chaque bit peut prendre l'une des deux valeurs, 0 ou 1. En combinant ces 8 bits, on peut représenter un large éventail de valeurs numériques. Par exemple, un octet peut être utilisé pour stocker un caractère tel qu'une lettre ou un chiffre.

Les unités de mesure standardisées en informatique pour quantifier la capacité de stockage :

Un kilooctet (ko) équivaut à 2^{10} octets, soit 1024 octets.

Un mégaoctet (Mo) équivaut à 2^{20} octets, soit 1024 kilooctets.

Un gigaoctet (Go) équivaut à 2^{30} octets, soit 1024 mégaoctets.

Un téraoctet (To) équivaut à 2^{40} octets, soit 1024 gigaoctets.

Un pétaoctet (Po) équivaut à 2^{50} octets, soit 1024 téraoctets.

Conversion d'un nombre décimal en binaire :

La méthode de division pour convertir un nombre décimal en binaire est une approche relativement simple et systématique. Elle consiste à diviser progressivement le nombre décimal par 2 et à enregistrer les restes de chaque division jusqu'à ce que le quotient atteigne zéro. Voici une explication détaillée de cette méthode :

Étape 1 : Prendre le nombre décimal que vous souhaitez convertir en binaire. Par exemple, prenons le nombre décimal 19 que nous voulons convertir en binaire.

Étape 2 : Diviser le nombre décimal par 2 (la base du système binaire) et enregistrer le quotient (résultat de la division) et le reste (ce qui reste après la division) :

Dividende initial : 19

Quotient : 19 ÷ 2 = 9 et Reste = 1

Étape 3 : Prendre le quotient de l'étape précédente (dans notre exemple, 9) et le diviser à nouveau par 2. Enregistrez le nouveau quotient et le reste :

Quotient : 9 ÷ 2 = 4, Reste = 1

Étape 4 : Répétez cette division successive jusqu'à ce que le quotient atteigne zéro. Chaque étape consiste à diviser le quotient précédent par 2 et à enregistrer le quotient et le reste.

Quotient : 4 ÷ 2 = 2, Reste = 0

Quotient : 2 ÷ 2 = 1, Reste = 0

Quotient : 1 ÷ 2 = 0, Reste = 1

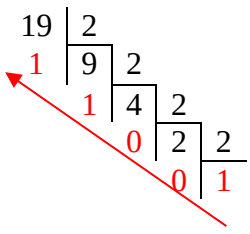
Étape 5 : Une fois que le quotient atteint zéro, vous avez terminé. Les restes enregistrés à chaque étape, lus de bas en haut, forment la représentation binaire du nombre décimal initial.

Dans notre exemple, la représentation binaire de 19 est 10011 (en lisant les restes de bas en haut).

$(19)_{10} = (10011)_2$

Remarque : on peut écrire aussi

$(19)_{10} = (00010011)_2$ on ajoute les 0 à gauche pour donner l'information sur 8 bits tel que le 0 à gauche ne modifié pas la valeur du code binaire.



Conversion d'un nombre décimal à virgule en binaire

La conversion d'un nombre décimal à virgule en binaire suit un processus similaire à celui de la conversion d'un nombre entier décimal en binaire, mais elle implique de prendre en compte la partie fractionnaire (virgule) du nombre. Voici une explication détaillée du processus de conversion d'un nombre décimal à virgule en binaire :

Étape 1 : Séparation de la partie entière et de la partie fractionnaire.

Prenez le nombre décimal à virgule que vous souhaitez convertir en binaire. Par exemple, considérons le nombre décimal 19,75.

Séparez la partie entière (19) de la partie fractionnaire (0,75).

Étape 2 : Conversion de la partie entière en binaire (nombre entier décimal en binaire).

Utilisez la méthode de division (comme expliqué précédemment) pour convertir la partie entière (19) en binaire. Dans cet exemple, la conversion donne $(19)_{10} = (10011)_2$

Étape 3 : Conversion de la partie fractionnaire en binaire (nombre fractionnaire décimal en binaire).

Pour la partie fractionnaire, vous pouvez utiliser une méthode similaire à la multiplication par 2 répétée. Multipliez la partie fractionnaire par 2 et notez la partie entière du résultat.

Répétez cette opération pour la partie entière obtenue jusqu'à ce que la partie fractionnaire atteigne zéro ou que vous ayez obtenu suffisamment de chiffres binaires. Généralement, on s'arrête après un certain nombre de chiffres binaires pour représenter une précision donnée.

Exemple : Pour 0,75, multipliez par 2, obtenez 1,5 (1 est la partie entière) ; puis multipliez 0,5 par 2, obtenez 1 (1 est la partie entière) .

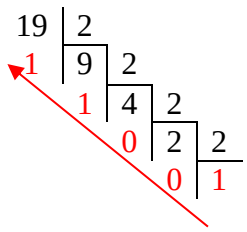
Enregistrer les parties entières successives pour obtenir la représentation binaire de la partie fractionnaire : 11.

Étape 4 : Combinaison de la partie entière et de la partie fractionnaire.

Prenez la représentation binaire de la partie entière (obtenue à l'étape 2) et la représentation binaire de la partie fractionnaire (obtenue à l'étape 3).

Ajoutez un point (virgule binaire) entre la partie entière et la partie fractionnaire pour obtenir la représentation binaire complète.

Conversion de la partie entière



Conversion de la partie fractionnaire

$0.75 \times 2 = 1.5$

$0.5 \times 2 = 1.0$

Résultat : $(19,75)_{10} = (10011,11)_2$.

Exemple :

$(0.15)_{10}= (?)_2$

$0.15\times2=0.3$

$0.3\times2=0.6$

$0.6\times2 = 1.2$

$0.2\times2 = 0.4$

$0.4\times2 = 0.8$

$0.8\times2 = 1.6$

$0.6\times2 = 1.2$

$0.2\times2 = 0.4$

$0.4\times2 = 0.8$

$0.8\times2 = 1.6$

$(0.15)_{10}=(0.00\underline{10011001})_2$

On dit que le nombre $(0.15)_{10}$ est cyclique dans la base 2 de période 1001.

Conversion binaire en décimal.

Il suffit de faire la somme des poids de chaque bit à 1, c'est-à-dire écrire le nombre sous forme polynomiale. Le nombre ci dessus est égal à

$(01000101)_2 = (1\times2^0 +1\times2^2 +1\times2^6)_{10} = (69)_{10}$

$(10011.11)_2= 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} = (19.75)_{10}$

I .3.3 Système de codage quaternaire :

Le système de codage quaternaire est un système de numération qui utilise une base de 4, ce qui signifie qu'il utilise quatre symboles différents pour représenter des valeurs numériques. Contrairement au système binaire (base 2), qui utilise 0 et 1, le système quaternaire utilise quatre symboles distincts. Chaque position dans un nombre quaternaire peut prendre l'une de ces quatre valeurs (0, 1, 2 et 3).

Exemples :

- ***Conversion* quaternaire *en décimal.***

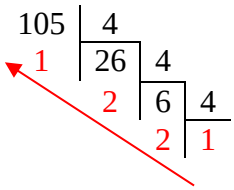
$(2331)_4= 2\times4^3 + 3\times4^2 + 3\times4^1 + 1\times4^0 = (189)_{10}$

$(130.21)_4= 1\times4^2 + 3\times4^1 +1\times4^0+ 2\times4^{-1} + 1\times4^{-2} = (29.5625)_{10}$

- ***Conversion d'un nombre décimal en* quaternaire.**

$(105,125)_{10} = (?)_4$

Conversion de la partie entière



Conversion de la partie fractionnaire

$0.125 \times 4 = 0.5$

$0.5 \times 4 = 2.0$

$(105,125)_{10} = (1221.02)_4$

I .3.4 Système de codage octal :

Le système de codage octal est un système de numération qui utilise une base de 8, ce qui signifie qu'il utilise huit symboles différents pour représenter des valeurs numériques. Contrairement au base 2 et 4, le système octal utilise huit symboles distincts. Chaque position dans un nombre octal peut prendre l'une de ces huit valeurs (les chiffres décimaux de 0 à 7.).

Exemples :

- ***Conversion* octal *en décimal.***

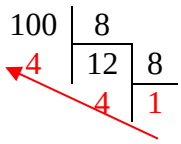
$(144.4)_8 = 1\times8^2 + 4\times8^1 + 4\times8^0 + 4\times8^{-1} = (100,5)_{10}$

$(1274.632)_8 = 1\times8^3 + 2\times8^2 + 7\times8^1 + 4\times8^0 + 6\times8^{-1} + 3\times8^{-2} + 2\times8^{-3} = (700.8007813)_{10}$

- ***Conversion d'un nombre décimal en* octal.**

$(100,5)_{10} = (\ ?\ )_8$

Conversion de la partie entière



Conversion de la partie fractionnaire

$0.5 \times 8 = 4.00$

$(100,5)_{10} = (144.4)_8$

I .3.5 Système de codage hexadécimal :

Le système de codage hexadécimal, est un système de numération qui utilise une base de 16, ce qui signifie qu'il utilise seize symboles différents pour représenter des valeurs numériques. Le système hexadécimal est largement utilisé en informatique et en électronique pour représenter des données binaires de manière plus concise et plus lisible pour les humains. Pour écrire les nombres en base 16 nous devons disposer de 16 chiffres, pour les dix premiers, nous utilisons les chiffres de la base 10, pour le suivant nous utiliserons des lettres de l'alphabet.

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Exemples :

- ***Conversion* hexadécimal *en décimal.***

$(B2F)_{16} = B\times16^2 + 2\times16^1 + A\times16^0 = (12\times256) + (2\times16) + (15\times1) = (3119)_{10}$

- ***Conversion d'un nombre décimal en* hexadécimal.**

$(3119,15)_{16} = (\ ?\ )_{16}$

Conversion de la partie entière

3119	16	
F	194	16
	2	B

Conversion de la partie fractionnaire

$0.15 \times 16 = 2.4$

$0.4\times 16 = 6.4$

$0.4\times 16 = 6.4$

$(3119,15)_{16} = (B2F.2\underline{66})_{16}$

I .3.6 Correspondance entre binaire et hexadécimal:

La correspondance entre les nombres binaires et hexadécimaux est essentielle en informatique, car le système hexadécimal permet de représenter des valeurs binaires de manière plus concise et plus lisible pour les humains. Dans le système hexadécimal, chaque chiffre hexadécimal (0-9 et A-F) équivaut à un groupe de quatre bits dans le système binaire. Les correspondances sont les suivantes :

hexadécimal	binaire	hexadécimal	binaire
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Ainsi, chaque chiffre hexadécimal correspond à un groupe de quatre bits binaires. Pour convertir un nombre binaire en hexadécimal, il vous suffit de regrouper les bits par groupes de quatre, de gauche à droite, et de leur attribuer la valeur hexadécimale correspondante.

Exemples :

$(110110100111)_2 = (1101\ 1010\ 0111)_2 = (DA7)_{16}$

Inversement, pour convertir un nombre hexadécimal en binaire, attribuez à chaque chiffre hexadécimal sa valeur binaire correspondante.

Exemple :

$(B2E)_{16} = (1011\ 0010\ 1110)_2$

En general, lorsque les bases B1 et B2 sont des puissances de 2, vous pouvez effectuer une conversion directe entre ces bases en utilisant la base 2 (binaire) en tant qu'intermédiaire. Cela simplifie le processus de conversion, car chaque chiffre de la base source correspond directement à une combinaison de bits dans la base cible. Cela est particulièrement utile dans les systèmes informatiques où les puissances de 2 sont couramment utilisées pour la representation des données.

Base quaternaire (base 4) :

La base quaternaire 4 = 22. Chaque chiffre quaternaire peut être directement converti en binaire en utilisant 2 bits. Par exemple, le chiffre quaternaire 3 correspond au binaire 11, car 3 en base 4 équivaut à 11 en base 2.

Base octale (base 8) :

La base octale $8 = 2^3$. Chaque chiffre octal peut être converti en binaire en utilisant 3 bits. Par exemple, le chiffre octal 5 correspond au binaire 101, car 5 en base 8 équivaut à 101 en base2.

Base hexadécimale (base 16) :

La base hexadécimale $16 = 2^4$. Chaque chiffre hexadécimal peut être converti en binaire en utilisant 4 bits. Par exemple, le chiffre hexadécimal A correspond au binaire 1010, car A en base 16 équivaut à 1010 en base 2.

Exemples :

Ecrire les nombres suivants en quaternaire(4), octal(8), hexadécimal(16). (1100101011000110 1.0001)2 ; (111010011011.00111)2 ;

$(01\ 10\ 01\ 01\ 01\ 10\ 00\ 11\ 01.00\ 01)_2 = (121112031.01)_4$

$(11\ 001\ 010\ 110\ 001\ 101.000\ 100)_2 = (312615.04)_8$

$(1\ 1001\ 0101\ 1000\ 1101.0001)_2 = (1958D.1)_{16}$

$(11\ 10\ 10\ 01\ 10\ 11.00\ 11\ 10)_2 = (322123.032)_4$

$(111\ 010\ 011\ 011.001\ 110)_2 = (7233.16)_8$

$(1110\ 1001\ 1011.0011\ 1000)_2 = (E9B.38)_{16}$

I .3.7 Les nombres signés:

Les nombres signés en binaire sont des représentations binaires de nombres qui peuvent être positifs ou négatifs. Pour représenter des nombres signés en binaire, différentes conventions ont été établies, et les deux méthodes les plus couramment utilisées sont la notation en complément à deux et la notation en complément à un.

- ***Notation en complément à un(CP1) :***

La notation en complément à un est moins courante que la notation en complément à deux, mais elle est utilisée dans certains systèmes. Voici comment cela fonctionne :

Les nombres positifs sont représentés de la manière habituelle en binaire, où le bit le plus significatif est un "0".

Les nombres négatifs sont obtenus en prenant le complément à un du nombre positif correspondant, c'est-à-dire en inversant tous les bits.

Exemple :

Pour représenter -3 en notation en complément à un, commencez par représenter 3 en binaire (0011) et inversez tous les bits pour obtenir 1100.

- ***Notation en complément à deux (CP2) :***

La notation en complément à deux est la méthode la plus répandue pour représenter des nombres signés en binaire. Voici comment cela fonctionne :

Les nombres positifs sont représentés de la manière habituelle en binaire, où le bit le plus significatif (le bit le plus à gauche) est un "0".

Les nombres négatifs sont obtenus en prenant le complément à un (inversion de tous les bits 0 en 1 et 1 en 0) du nombre positif correspondant, puis en ajoutant 1 au résultat.

Exemple :

Pour représenter -3 en notation en complément à deux, commencez par représenter 3 en binaire (0011), inversez tous les bits (1100), puis ajoutez 1 pour obtenir 1101.

La notation en complément à deux est généralement préférée, car elle a l'avantage de représenter un seul zéro pour zéro positif et négatif, ce qui évite les problèmes de doublons.

De plus, elle permet des opérations arithmétiques simples sur les nombres signés.

Les nombres signés en binaire sont essentiels en informatique pour la représentation des nombres négatifs, et ils sont couramment utilisés dans les processeurs d'ordinateurs et les systèmes de stockage de données.

Le plus grand nombre signé sur 8 bits est +127 (01111111) Le plus petit nombre signé sur 8 bits est -128 (10000000)

-128à+127=>256combinaisons(2^8)

Ce qui s'applique sur 8 bits s'applique aussi sur 4, 16, ... bits.

- ***Notifications : L'addition binaire :***

L'addition de deux bits binaire se réalise selon la spécification suivante :

bit1		Bit2		résultat	retenue
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

L'addition binaire de deux nombres s'effectue bit à bit de droite à gauche, en reportant les retenues

I .3.8 Le décimal codé binaire (code DCB):

Le terme "décimal codé binaire" (DCB), en anglais "binary-coded decimal" (BCD), fait référence à une méthode de représentation de nombres décimaux en utilisant des chiffres binaires. Le but du

codage DCB est de permettre une représentation binaire de nombres décimaux de manière plus directe. Voici comment cela fonctionne :

Chiffres décimaux : Les chiffres décimaux, de 0 à 9, sont représentés en binaire. Chaque chiffre décimal est codé en utilisant quatre bits binaires. Par exemple, le chiffre décimal 0 est représenté en binaire comme 0000, le chiffre 1 est 0001, le chiffre 2 est 0010, et ainsi de suite jusqu'à 9 qui est représenté comme 1001.

Représentation des nombres : Pour représenter un nombre décimal en utilisant le codage DCB, chaque chiffre décimal est converti en sa représentation binaire à quatre bits, puis les chiffres binaires sont concaténés pour former le nombre complet.

Exemple:

Pour représenter le nombre décimal 723 en DCB, vous auriez:

7 en DCB : 0111

2 en DCB : 0010

3 en DCB : 0011

Donc, $(723)_{10}$ =$(0111\ 0010\ 0011)_{DCB}$.

Le principal avantage du codage DCB est qu'il permet une représentation efficace des nombres décimaux en utilisant des chiffres binaires. Cela peut être particulièrement utile dans les systèmes où la manipulation des chiffres décimaux est nécessaire, tout en conservant les avantages de la représentation binaire pour le stockage et la manipulation des données.

Le codage DCB est couramment utilisé dans les systèmes électroniques, tels que les calculatrices, les afficheurs à sept segments et d'autres dispositifs où les chiffres décimaux doivent être affichés ou traités en utilisant des composants électroniques qui fonctionnent en binaire.

I .3.9 Le codage ASCII:

Le codage ASCII (American Standard Code for Information Interchange) est un système de codage caractère-à-binaire qui attribue un code numérique unique à chaque caractère imprimable, tel que les lettres, les chiffres, la ponctuation, les symboles spéciaux et les commandes de contrôle dans le domaine de l'informatique et des communications. Le codage ASCII est l'un des systèmes de codage de caractères les plus couramment utilisés et a une longue histoire remontant aux premiers jours de l'informatique.

Voici quelques points clés à retenir sur le codage ASCII :

- ***Représentation des caractères :*** Le codage ASCII attribue un nombre unique à chaque caractère. Par exemple, la lettre majuscule "A" est représentée par le nombre 65 en ASCII, tandis que la lettre minuscule "a" est représentée par le nombre 97.
- ***7 bits originaux et extension à 8 bits :*** L'ensemble de caractères ASCII original utilisait 7 bits pour représenter chaque caractère, permettant ainsi 128 caractères différents (de 0 à 127). Plus tard, une extension à 8 bits appelée "ASCII étendu" a été développée, ce qui a permis de

représenter un ensemble plus large de caractères, notamment des caractères accentués, des symboles et des caractères spécifiques à certaines langues.

- ***Caractères de contrôle :*** En plus des caractères imprimables, le codage ASCII inclut également des caractères de contrôle, tels que le retour à la ligne, la tabulation, le saut de page, etc. Ces caractères de contrôle sont utilisés pour formater le texte et contrôler les dispositifs de sortie.
- ***Compatibilité universelle :*** L'un des avantages majeurs de l'ASCII est sa compatibilité universelle. La plupart des ordinateurs, systèmes d'exploitation, langages de programmation et périphériques comprennent et utilisent le codage ASCII.
- ***ASCII étendu :*** Bien que l'ASCII original ait été conçu pour l'anglais américain, l'ASCII étendu a inclus des caractères spécifiques à d'autres langues et des symboles utiles dans divers domaines, ce qui a permis son utilisation internationale.
- ***Codage binaire :*** Chaque caractère ASCII est représenté par une séquence binaire de 7 ou 8 bits, ce qui permet une représentation binaire directe des caractères dans les systèmes informatiques.

I .3.10 Exercices d'application:

Exercice 1

Convertir en nombres décimaux les nombres binaires suivants : 110, 1100, 100101110.

Solution:

$(110)_2=2^2+2^1=6$

$(1100)_2=2^3+2^2+=12$

$(100101111)_2 = 2^8+2^5+2^3+2^2+2^1+2^0 = 303$

Exercice 2

Convertir en nombres binaires les nombres décimaux suivants : 43, 51, 128, 131, 202.

Solution:

$43=(101011)_2$

$55=(110111)_2$

$128= (10000000)_2$

$202 = (11001010)_2$

Exercice 3

Convertir en nombres binaires puis en nombres décimaux les nombres hexadécimaux suivants : 12, DADA et 5F3.

Solution:

Pour la conversion hexadécimal→binaire :

$(12)_{16}= (0001\ 0010)_2$

$(DADA)_{16} = (1101\ 1010\ 1101\ 1010)_2$

$(5F3)_{16} = (0101\ 1111\ 0011)_2$

Pour la conversion hexadécimal→décimal :

$(12)_{16} = 1\times161 +2\times160 = 18$

$(DADA)_{16} = 13\times163 +10\times162 +13\times161 +10\times160 = 56026$

$(5F3)_{16} = 5\times162 +15\times161 +3\times160 = 1523$

Exercice 4

Faire les conversions suivantes :

Base X à base 10 $(231)_4$ = (.....)$_{10}$;$(1523)_8$ = (.....)$_{10}$ (BAF F); 16= (.....)10

(22.01)$_4$ = (.....)$_{10}$; $(152.44)_8$ = (.....)$_{10}$; $(10B.7)_{16}$= (.....)$_{10}$

Base 10 à base X $(53)_{10}$ = (.....)$_4$; $(142)_{10}$ = (.....)$_2$; $(253)_{10}$ = (.....)$_{16}$

$(148,8)_{10}$= (.....)$_{16}$; $(312.3)_{10}$= (.....)$_4$;$(7.875)_{10}$= (.....)$_8$

Solution:

Faire les conversions suivantes : on utilise l'écriture polynomiale pour trouver le résultat

Base X à base 10 $(231)_4 = (45)_{10}$; $(1523)_8 = (851)_{10}$; $(BAF\ F)_{16}= (47871)_{10}$

(22.01)$_4$ = $(10.0625)_{10}$; $(152.44)_8 = (106.5625)_{10}$; $(10B.7)_{16}= (267.4375)_{10}$

Base 10 à base X $(53)_{10} = (311)_4$; $(142)_{10} = (10001110\ )_2$; $(253)_{10} = (FD)_{16}$

$(148,8)_{10}= (98.\ 12\ 12\ldots)_{16}$; $(312.3)_{10}= (10320\ .\ 1\ 03\ 03\ldots)_4$; $(7.875)_{10}= (7.7)_8$

Exercice 5

Effectuer les conversions suivantes en utilisant la base 2 comme base intermédiaire : a. $(673)_8$ vers l'hexadécimal.$(673)_8$=(.....)$_2$=(.....)$_{16}$

b. $(E7C)1_6$ vers l'octal.$(E7C)_{16}$=(.....)$_2$=(.....)$_8$

c. Ecrire les nombres suivants en quaternaire(4), octal(8), hexadécimal(16). 111010100001100101.101 ; 11001010110001101.0001 ; 111010011011.00111

Solution:

Effectuer les conversions suivantes en utilisant la base 2 comme base intermédiaire :

a. $(673)_8$ vers l'hexadécimal. $(673)_8$=$(110\ \ 111\ 011)_2$=$(1\ \ 1011\ \ 1011)_2$=$(1BB)_{16}$

b. $(E7C)_{16}$ vers l'octal.$(E7C)_{16}$=$(1110\ 0111\ 1100)_2$=$(111\ 001\ 111\ 100)_2$=$(7174)_8$

c. Ecrire les nombres suivants en quaternaire(4), octal(8), hexadécimal(16).

$(111010100001100101.101)_2$; $(11001010110001101.0001)_2$; $(111010011011.00111)_2$;

$(11\ 10\ 10\ 10\ 00\ 01\ 10\ 01\ 01.10\ 10)_2 = (322201211.22)_4$

$(111\ 010\ 100\ 001\ 100\ 101.101)_2 = (724145.5)_8$

$(11\ 1010\ 1000\ 0110\ 0101.1010)_2 = (3A865.A)_{16}$

$(1\ 10\ 01\ 01\ 01\ 10\ 00\ 11\ 01.00\ 01)_2 = (121112031.01)_4$

$(11\ 001\ 010\ 110\ 001\ 101.000\ 100)_2 = (312615.04)_8$

$(1\ 1001\ 0101\ 1000\ 1101.0001)_2 = (1958D.1)_{16}$

$(11\ 10\ 10\ 01\ 10\ 11.00\ 11\ 10)_2 = (322123.032)_4$

$(111\ 010\ 011\ 011.001\ 110)_2 = (7233.16)_8$

$(1110\ 1001\ 1011.0011\ 1000)_2 = (E9B.38)_{16}$

Exercice 6

Effectuer les transcodages suivants :

$(\ 5\ 7\ 6)_{10} = (\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ )_{DCB}$

$(\ 9\ 9\ )_{10} = (\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ )_{DCB}$

$(\ 1000\ 0011\ 0110)_{DCB} = (\ \ldots\ldots\ldots\ldots\ldots\ )_{10}$

Solution:

$(\ 5\ 7\ 6)_{10} = (\ 0101\ 0111\ 0110\ )_{DCB}$

$(\ 9\ 9\ )_{10} = (\ 1001\ 1001\ )_{DCB}$

$(\ 1000\ 0011\ 0110)_{DCB} = (\ 836\ )_{10}$

Exercice 7

1-Donnez la représentation de $(-34)_{10}$ en Complément à 2:

Sur 8 bits : $(-34)_{10}$=(.....)ca2

Sur 10 bits : $(-34)_{10}$=(.....)ca2

2-Peut-on représenter ce nombre en ca2 sur 6 bits (justifier votre réponse)?

3-Donner les 4 nombres entiers à la suite de $(7FC)_{16}$

4-Soit le nombre décimal $X= 4a^5 + 2a^3 + a + 5$, tel que a est un entier (a>5). Exprimer X en base a

Solution:

1-Sur 8 bits : $(-34)_{10}$=(11011110)ca2

Sur 10 bits : $(-34)_{10}$=(1111011110)ca2

2-Non on ne peut pas car sur 6 bits on peut coder 26 valeurs de -2^5 à 2^5-1 c'est-à-dire les valeurs de [-32,+31]

13- 7FD , 7FE , 7FF , 800

24- X=(402015)a

I .4. Principe de fonctionnement d'un ordinateur:

Le principe de fonctionnement d'un ordinateur est basé sur l'exécution d'instructions en langage binaire (0 et 1) pour effectuer diverses tâches, notamment le traitement de données, le stockage d'informations, la communication et l'affichage de résultats. Voici un aperçu des principaux composants et du fonctionnement général d'un ordinateur :

- ***Unité centrale de traitement (CPU) :*** La CPU est le cerveau de l'ordinateur. Elle exécute les instructions en lisant des programmes stockés en mémoire. La CPU comprend une unité de contrôle, qui supervise l'exécution des instructions, et une unité arithmétique et logique (ALU) qui effectue des opérations mathématiques et logiques.
- ***Mémoire centrale :*** La mémoire centrale de l'ordinateur, généralement appelée RAM (Random Access Memory), est l'endroit où les données et les programmes en cours d'exécution sont temporairement stockés. La RAM est volatile, ce qui signifie que son contenu est effacé lorsque l'ordinateur est éteint.
- ***Unité de stockage :*** L'ordinateur dispose également d'unités de stockage non volatiles, telles que des disques durs et des disques SSD, où les données sont stockées à plus long terme. Ces dispositifs permettent de conserver des programmes, des fichiers et des données même lorsque l'ordinateur est éteint.
- ***Entrées et sorties (E/S) :*** Les périphériques d'entrée (comme le clavier et la souris) permettent à l'utilisateur de communiquer avec l'ordinateur. Les périphériques de sortie (comme l'écran et l'imprimante) affichent ou impriment les résultats des opérations de l'ordinateur.
- Bus : Les bus sont des voies de communication qui relient tous les composants de l'ordinateur, permettant le transfert de données entre eux. Il existe généralement plusieurs types de bus, y compris le bus de données, le bus d'adresse et le bus de contrôle.
- ***Système d'exploitation :*** Le système d'exploitation (comme Windows, macOS ou Linux) est un logiciel qui gère les ressources matérielles de l'ordinateur, coordonne les programmes et fournit une interface utilisateur. Il permet également d'effectuer des opérations telles que le démarrage, l'arrêt et la gestion des fichiers.

Le fonctionnement de l'ordinateur repose sur l'exécution de programmes, qui sont des séquences d'instructions en langage binaire stockées en mémoire. Lorsqu'un programme est exécuté, la CPU récupère les instructions en mémoire, les décode, les exécute et stocke les résultats dans la mémoire. Ce processus se répète continuellement, permettant à l'ordinateur de réaliser des tâches complexes.

L'ordinateur fonctionne en suivant un modèle de cycle d'instructions de base : récupération, décodage, exécution et stockage des résultats. Ce modèle permet à l'ordinateur de traiter des données, de réaliser des calculs, d'interagir avec l'utilisateur et d'accomplir une variété de tâches, en fonction des programmes et des instructions fournies.

I. 5-Partie matériel d'un ordinateur:

La partie matérielle d'un ordinateur comprend tous les composants physiques qui sont nécessaires au fonctionnement de la machine. Ces composants sont responsables du traitement des données, du stockage de l'information et de l'interaction avec les utilisateurs. Voici les principaux éléments de la partie matérielle d'un ordinateur :

- ***Unité centrale de traitement (CPU):*** La CPU, souvent appelée processeur, est le cerveau de l'ordinateur. Elle exécute des instructions et effectue des opérations mathématiques et logiques. La CPU peut avoir plusieurs cœurs pour traiter des tâches en parallèle.
- ***Mémoire vive (RAM):*** La RAM est une mémoire volatile qui stocke temporairement les données et les programmes en cours d'utilisation. Elle permet un accès rapide aux informations nécessaires pour l'exécution des programmes.
- ***Unité de stockage:*** L'unité de stockage comprend les disques durs, les disques SSD (Solid State Drive) et les lecteurs de CD/DVD. Ces dispositifs permettent de stocker des programmes, des fichiers et des données sur une base permanente ou semi-permanente.
- ***Carte mère:*** La carte mère est le circuit principal de l'ordinateur. Elle relie tous les composants matériels et permet la communication entre eux. La carte mère comprend également des ports pour connecter des périphériques tels que le clavier, la souris et l'écran.
- ***Carte graphique:*** La carte graphique (ou GPU, Graphics Processing Unit) est responsable du traitement des données graphiques et de l'affichage de l'image sur l'écran. Elle est essentielle pour les applications graphiques, les jeux vidéo et les tâches nécessitant un affichage haute résolution.
- ***Alimentation:*** L'alimentation électrique fournit l'énergie nécessaire au fonctionnement de l'ordinateur. Elle convertit l'électricité en tension et en courant adaptés aux besoins de chaque composant.
- ***Refroidissement:*** Les systèmes de refroidissement, tels que les ventilateurs et les dissipateurs thermiques, évacuent la chaleur générée par la CPU, la carte graphique et d'autres composants afin de maintenir des températures de fonctionnement sûres.
- ***Périphériques d'entrée et de sortie:*** Les périphériques d'entrée, tels que le clavier et la souris, permettent à l'utilisateur de communiquer avec l'ordinateur. Les périphériques de sortie, tels que l'écran et l'imprimante, affichent ou impriment les résultats.
- ***Cartes d'extension:*** Ces cartes, telles que les cartes réseau, les cartes son et les cartes d'extension de ports, permettent d'ajouter des fonctionnalités supplémentaires à l'ordinateur.
- ***Boîtier:*** Le boîtier de l'ordinateur abrite tous les composants matériels et protège l'intérieur de la poussière et des dommages. Il offre également des ports d'entrée/sortie pour connecter des périphériques.

- ***Bus:*** Les bus de données, d'adresse et de contrôle sont des voies de communication qui permettent le transfert de données entre les composants matériels.
- ***Horloge:*** L'horloge de l'ordinateur génère des impulsions à intervalles réguliers pour synchroniser les opérations de la CPU.

La combinaison de ces composants matériels, interconnectés et fonctionnant en tandem, permet à un ordinateur de traiter des données, d'exécuter des programmes, de stocker des informations, d'afficher des résultats et d'interagir avec les utilisateurs. La partie matérielle est complétée par le système d'exploitation et les logiciels applicatifs, qui exploitent ces ressources matérielles pour réaliser une variété de tâches.

I. 6 Partie système d'un ordinateur:

La partie système d'un ordinateur fait référence à l'ensemble du logiciel et des composants qui permettent à l'ordinateur de fonctionner. Cette partie inclut le système d'exploitation, les pilotes, les utilitaires système, les logiciels de base et d'autres composants essentiels pour le bon fonctionnement de l'ordinateur. Voici un aperçu des principaux éléments de la partie système d'un ordinateur :

- ***Système d'exploitation (SE) :*** Le système d'exploitation est le logiciel de base qui gère les ressources matérielles de l'ordinateur. Il coordonne l'exécution des programmes, la gestion de la mémoire, la gestion des fichiers, l'interaction avec les périphériques, la sécurité et d'autres tâches essentielles. Les exemples de systèmes d'exploitation incluent Windows, macOS, Linux, Android, et d'autres.
- ***Pilotes de périphériques :*** Les pilotes sont des logiciels qui permettent au système d'exploitation de communiquer avec les périphériques matériels tels que l'imprimante, la carte graphique, le clavier, la souris, etc. Chaque périphérique a généralement besoin de son propre pilote pour fonctionner correctement.
- ***Utilitaires système :*** Les utilitaires système sont des programmes conçus pour maintenir et optimiser les performances de l'ordinateur. Ils incluent des outils de défragmentation de disque, d'optimisation de la mémoire, de surveillance des performances, de nettoyage du registre, etc.
- ***Gestionnaire de fichiers :*** Le gestionnaire de fichiers est un composant du système d'exploitation qui permet aux utilisateurs de créer, d'organiser, de déplacer, de copier et de supprimer des fichiers et des dossiers sur le disque dur ou d'autres dispositifs de stockage.
- ***Système de fichiers :*** Le système de fichiers est une structure qui définit comment les données sont organisées et stockées sur le disque dur. Il gère l'accès aux fichiers et dossiers, ainsi que la façon dont les données sont stockées et indexées.

- ***Bibliothèques système :*** Les bibliothèques système sont des collections de fichiers et de fonctions partagées utilisées par les programmes et le système d'exploitation. Elles facilitent la réutilisation du code et la compatibilité entre les applications.
- ***Services système :*** Les services système sont des programmes qui s'exécutent en arrière-plan pour effectuer des tâches telles que la gestion du réseau, la planification des tâches, la sécurité et d'autres fonctions essentielles.
- ***Environnements de développement :*** Ces environnements fournissent aux développeurs des outils pour créer des logiciels, des applications et des pilotes qui s'exécuteront sur l'ordinateur. Les environnements de développement incluent des compilateurs, des débogueurs et d'autres outils de programmation.
- ***Mises à jour et correctifs :*** Les mises à jour du système d'exploitation, les correctifs de sécurité et les mises à jour de pilotes sont essentiels pour maintenir la sécurité, la stabilité et les performances de l'ordinateur.
- ***Configuration système :*** Les paramètres de configuration du système permettent aux utilisateurs de personnaliser le comportement de leur ordinateur, notamment la résolution de l'écran, les paramètres de réseau, les options de sécurité, etc.
- ***Système de gestion des utilisateurs et des comptes :*** Les systèmes d'exploitation offrent des mécanismes pour créer des comptes d'utilisateurs, gérer les autorisations d'accès et garantir la sécurité du système.

La partie système est essentielle pour le fonctionnement de l'ordinateur, car elle assure la coordination entre les composants matériels et les applications logicielles. Elle permet également aux utilisateurs de configurer leur ordinateur, de gérer leurs fichiers, de garantir la sécurité et de maintenir la performance de la machine.

Chapitre II : Notions d'algorithme et de programme

II.1 Concept d'un algorithme :

Un algorithme est un ensemble fini et bien défini d'instructions ou de règles logiques qui sont suivies dans un ordre précis pour accomplir une tâche ou résoudre un problème spécifique. Les algorithmes sont des séquences d'étapes clairement définies et non ambiguës qui permettent de transformer une entrée donnée en une sortie souhaitée. Les algorithmes sont largement utilisés dans le domaine de l'informatique, des mathématiques, de la science, de l'ingénierie et d'autres disciplines pour automatiser des processus et résoudre des problèmes. Voici les principaux concepts associés aux algorithmes :

Définition claire : Un algorithme doit avoir une définition précise et non ambiguë. Chaque étape de l'algorithme doit être clairement comprise et décrite.

Entrées et sorties : Un algorithme reçoit des données en entrée, les traite conformément aux instructions définies et produit une sortie. Les données d'entrée et de sortie sont des éléments clés de la conception de l'algorithme.

Séquence d'étapes : Un algorithme consiste en une séquence d'étapes ordonnées. Chaque étape indique ce qui doit être fait à un moment donné.

Répétition (boucles) : Les algorithmes peuvent inclure des structures de répétition (boucles) pour effectuer la même séquence d'actions plusieurs fois en fonction de certaines conditions.

Sélection (instructions conditionnelles) : Les algorithmes peuvent inclure des instructions conditionnelles pour prendre des décisions en fonction de certaines conditions. Par exemple, "si une condition est vraie, effectuer une action A ; sinon, effectuer une action B."

Structure modulaire : Les algorithmes peuvent être divisés en modules ou sous-algorithmes, ce qui facilite la réutilisation du code et la maintenance.

Optimisation : La conception d'un algorithme peut viser à améliorer l'efficacité, c'est-à-dire à minimiser le temps, l'espace mémoire ou les ressources nécessaires pour accomplir la tâche.

Correction : Un algorithme doit être correct, c'est-à-dire qu'il doit produire les résultats attendus pour toutes les entrées valides. La correction d'un algorithme est vérifiée à l'aide de tests et de preuves formelles.

Complexité : L'analyse de la complexité d'un algorithme permet de déterminer combien de temps ou de ressources il faudra pour le faire fonctionner. Cela peut être exprimé en termes de notation "O" (par exemple, O(n) pour une complexité linéaire).

Adaptabilité : Les algorithmes doivent être adaptables pour résoudre différents problèmes ou pour fonctionner dans des environnements variés.

Langage indépendant : Un algorithme peut être conçu en utilisant des concepts et des instructions logiques, et n'est pas spécifiquement lié à un langage de programmation particulier.

Les algorithmes sont au cœur de la résolution de problèmes informatiques et de nombreuses autres applications. Ils sont utilisés pour trier des listes, rechercher des éléments, effectuer des calculs

mathématiques, gérer des bases de données, créer des graphiques, optimiser des itinéraires, et bien plus encore. La conception d'algorithmes efficaces est une compétence essentielle dans le domaine de l'informatique et de l'ingénierie.

II. 2 Représentation en organigramme :

La représentation en organigramme est un moyen visuel de représenter un algorithme ou un processus en utilisant des symboles, des formes et des flèches pour montrer la séquence d'étapes, les décisions, les boucles, les entrées et les sorties. Voici les symboles de base couramment utilisés dans un organigramme :

Terminaison (oval) : Ce symbole indique le début ou la fin de l'algorithme ou du processus. Il est généralement utilisé pour délimiter le début et la fin d'un organigramme.

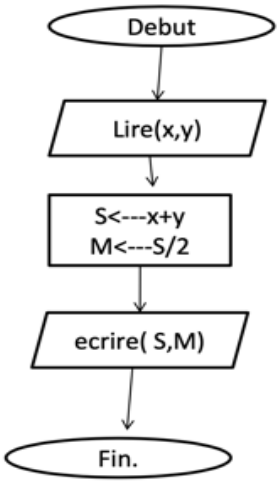
Processus (rectangle) : Le symbole en forme de rectangle représente une étape ou une action à effectuer. C'est là que vous décrivez l'action ou l'instruction à suivre.

Décision (losange) : Le losange est utilisé pour représenter une condition ou une décision à prendre. Une flèche sort de la décision et pointe vers l'une des deux options possibles, généralement indiquées sous la forme "Vrai" et "Faux".

Entrée/Sortie (parallélogramme) : Le parallélogramme est utilisé pour indiquer une entrée (comme la saisie de données) ou une sortie (comme l'affichage de résultats).

Connecteur (cercle vide) : Les connecteurs servent à connecter différentes parties de l'organigramme lorsqu'il y a trop de lignes de liaison. Ils sont généralement numérotés pour montrer la séquence des étapes.

Flèches (flèches simples ou doubles) : Les flèches indiquent la direction de flux entre les étapes, les décisions, les entrées et les sorties.



II. 3 Structure d'un programme :

II. 3.1 Structure d'un algorithme :

L'algorithme doit avoir une structure bien définie. Cette structure doit comporter :

- L'en-tête qui comprend le nom de l'algorithme pour identifier l'algorithme.
- Les déclarations des variables et des constantes.
- Le corps de l'algorithme qui contient les instructions.

Toutes les instructions doivent situer entre le mot ***Début*** et le mot ***Fin***, et chaque instruction doit comporter un point-virgule à la fin.

Algorithme nom_Algorithme ; } Entête

Déclaration des constantes ;
Déclaration des variables ; } Partie déclaration

Debut
    <instruction 1>;
    <instruction 2>; } Le corps
    <instruction 3>;
    Fin.

II. 3.2 Structure d'un programme :

En Pascal, un programme est généralement organisé en plusieurs parties, chacune ayant un rôle spécifique:

- ***En-tête du programme :*** Cette section contient des informations sur le programme, telles que le nom de l'auteur, la date de création, une brève description du programme, etc. Elle est généralement placée en début de fichier pour la documentation.
- ***uses :*** Cette partie indique les bibliothèques ou les modules que vous allez utiliser dans votre programme. Vous pouvez inclure des bibliothèques standard de Pascal ou des unités personnalisées.
- ***const :*** Vous pouvez déclarer des constantes qui seront utilisées dans tout le programme. Les constantes sont des valeurs immuables.
- ***var*** : Vous déclarez ici des variables globales, c'est-à-dire des variables accessibles dans tout le programme.
- ***begin....end*** : C'est la partie principale du programme où vous écrivez le code qui sera exécuté. Vous pouvez inclure des instructions de contrôle de flux (comme des boucles et des conditions) et des appels de fonctions ou de procédures ici.

Program nom_program;

uses (liste des modules utilisés) ;

Var (Déclaration des variables);

Const (Déclaration des constantes);

Begin

<instruction 1>;

<instruction 2>;

<instruction 3>;

end.

Exemple :

Ecrire un programme en Pascal qui affiche "Bonjour" à l'écran :

```
program Bonjour;
begin
writeln('Bonjour');
end.
```

II.4 Structure des données :

II.4.1 Les Constantes et Les variables :

Les Constantes : Les constantes en Pascal sont des valeurs immuables qui ne peuvent pas être modifiées une fois qu'elles sont définies. Elles sont utiles pour stocker des valeurs qui ne changent pas pendant l'exécution du programme. Pour déclarer une constante en Pascal, utilisez le mot-clé **const** suivi du nom de la constante, du signe égal =, et de la valeur constante.

Exemple

```
program Constantes;
const PI= 3.14159;
begin
  writeln('La valeur de PI est : ', PI);
end.
```

Les variables :

Les variables en Pascal sont utilisées pour stocker des données qui peuvent être modifiées au cours de l'exécution du programme. Vous devez déclarer le type de la variable lors de sa création. Pour déclarer une variable en Pascal, utilisez le mot-clé var suivi du nom de la variable et de son type de données

II.4.2 *Les types de données de bases*:

Les variables et les constantes peuvent avoir cinq types de base :

- ***Type entier :*** un type numérique qui représente l'ensemble des entiers naturels et relatifs, tels que : 0, 45, -10,...
  Mot clé : **entier** en Pascal **integer.**
- ***Type réel:*** un autre type numérique qui représente les nombres réels, telsque:0.5, -3.67, 1.5e+5 ,... Mot clé : **réel** en Pascal **real.**
- ***Type caractère :*** représente tous les caractères alphanumériques tels que : ′a′, ′B′, ′*′, ′9′, ′@′, ′ ′,...
  Mot clé : **car** en Pascal ***char.***
- ***Type chaînes de caractères:*** concerne des chaines de caractères telsque des mots ou des phrases : ″informatique″, ″la section B″,...
  Mot clé : ***chaîne*** en Pascal ***string***.
- ***Type booléen :*** ce type ne peut prendre que deux états : vrai ou faux
  Mot clé : booléen en Pascal **boolean.**

Exemple :

```
program ExempleVariables;
var
  C: integer;
  Nom: string;
begin
  C := 0;
  Nom := 'Hicham';
writeln('C = ', C);
writeln('Nom : ', Nom);
end.
```

II.5 Les opérateurs :

En algorithmique, les opérateurs sont des symboles spéciaux utilisés pour effectuer diverses opérations sur les données.

Les opérateurs les plus couramment utilisés en algorithmique :

- ***L'opérateur d'affectation***: représenté par le symbole «← » (:=), qui confère une valeur à une variable ou à une constante. (affectation de la valeur à la variable (ou à la constante)).

Syntaxe

identificateur_1← identificateur_2 ;(Affecte à la variable 1 le contenu de la variable 2)

identificateur_1 ←expression; (Affecte à la variable 1 le résultat de l'expression)

Exemples

où A, B et C sont de type real, i, j de type integer et S de type string (chaîne de caractères)

A←B{ assigne à A la valeur de B } ;

A←i{ assigne à A la valeur de i } ;

Remarques:

Ne pas confondre ':= ' (affectation) avec ' = ' (comparaison des variables dans un test) - Ne pas inverser les identificateurs ' A:= B ' et ' B := A ' donnent des résultats différents.

- ***Opérateurs arithmétiques :***

Addition (+) : Utilisé pour additionner deux valeurs.

Soustraction (-) : Utilisé pour soustraire la deuxième valeur de la première.

Multiplication (*) : Utilisé pour multiplier deux valeurs.

Division (/) : Utilisé pour diviser la première valeur par la deuxième.

Modulo (%) : Utilisé pour obtenir le reste de la division entière de la première valeur par la deuxième.

- ***Opérateurs de comparaison :***

Égal (=) : Vérifie si deux valeurs sont égales.

Différent (<>) : Vérifie si deux valeurs sont différentes.

Inférieur (<) : Vérifie si la première valeur est inférieure à la deuxième.

Inférieur ou égal (<=) : Vérifie si la première valeur est inférieure ou égale à la deuxième.

Supérieur (>) : Vérifie si la première valeur est supérieure à la deuxième.

Supérieur ou égal (>=) : Vérifie si la première valeur est supérieure ou égale à la deuxième.

- ***Opérateurs logiques :***

ET en Pascal **and:** Renvoie TRUE si les deux opérandes sont TRUE.

OU en Pascal **or:** Renvoie TRUE si au moins l'une des opérandes est TRUE.

NON en Pascal **not:** Inverse la valeur d'une expression booléenne (TRUE devient FALSE, et vice versa

II.6 Les opérations d'entrée/sortie:

En algorithmique, les opérations d'entrée/sortie (E/S) sont essentielles pour permettre à un algorithme de communiquer avec l'utilisateur ou avec d'autres composants du système. Les opérations d'entrée permettent de recueillir des données, tandis que les opérations de sortie servent à afficher des informations ou à enregistrer des résultats.

II.6.1 Opérations d'entrée (Input) en pseudo-code *:*

il n'existe pas de fonctions ou de procédures spécifiques pour effectuer des opérations d'entrée. Vous pouvez simplement indiquer comment les données seront obtenues (par exemple, depuis le clavier ou un fichier)

Exemple:

Écrire "Entrez votre nom :"

Lire NomUtilisateur

En Pascal, vous pouvez utiliser la fonction **readln** pour lire des données entrées par l'utilisateur depuis le clavier.

II.6.2 Opérations de sortie (Output):

Pour effectuer des opérations de sortie, vous pouvez utiliser l'instruction **Écrire** ou **Afficher**, suivi du texte ou des données que vous souhaitez afficher.

En Pascal, pour afficher des informations à l'utilisateur ou enregistrer des résultats, vous pouvez utiliser la fonction **writeln** pour écrire du texte suivi de variables ou de constantes.

II.6.3 Exercices d'application:

Exercice 1

Ecrire un programme en Pascal qui affiche le message "Bonjour" à l'écran

Solution :

```
program Ex1;
USES crt ;
begin
  writeln('Bonjour');
  writeln('Le résultat de 5 + 3 est : ', 5 + 3);
Readkey();
end.
```

Exercice 2

ecrire un programme en Pascal qui demande à l'utilisateur de saisir son nom, puis affiche le message "Bonjour, Nom de l'utilisateur"

Solution :

```
program LectureClavier;
USES crt;
var NomUtilisateur: string;
begin
  writeln('Entrez votre nom : ');
  readln(NomUtilisateur);
  writeln('Bonjour, ', NomUtilisateur, ' !');
```

```
Readkey();
end.
```

Exercice 3

ecrire un programme en Pascal qui calcule la valeur de B en utilisant la formule B = 2 * A, avec A ayant la valeur de 300.

Solution :

```
Program affectation ;
USES crt;
Var A, B :integer ;
Begin
A:=300;
B:=A*2 ;
Writeln('A=',A) ;
Writeln('B=',B) ;
Readkey();
End.
```

Exercice 4

ecrire un programme en Pascal qui calcule la somme de deux variables, A et B, et affiche le résulta.

Solution :

```
Program sum ;
USES crt;
Var A, B,S: integer ;
Begin
Writeln('A='); Readln(A);
Writeln('B='); Readln(B);
S:= A+B;
Writeln('S=',S) ;
Readkey();
End.
```

Exercice 5

ecrire un programme simple en Pascal qui calculer la surface d'un disque.

Solution :

```
Program Surface_disque;
USES crt;
Var R, Sur : real ;
Const Pi = 3.14 ;
Begin
 Writeln ('Donner le rayon du disque : ') ;
Readln (R) ;
Sur:= (R ^ 2) * Pi ;
Writeln ('La surface du disque est : ', Sur) ;
Readkey():
Fin.
```

II.7 Les structures de contrôle conditionnel :

Les structures conditionnelles, ne permettent d'exécuter certaines instructions, que sous certaines conditions. Une condition (expression conditionnelle ou logique) est évaluée, c'est à dire qu'elle est jugée vraie ou fausse. Si elle est vraie, un traitement (une ou plusieurs instructions) est réalisé; si la condition est fausse, une autre instruction va être exécutée, et ensuite le programme va continuer normalement.

II.7.1 Structures conditionnelles simples :

Syntaxe	**En Pascal**

Si <condition> **Alors**
Instructions 1
 	
Finsi
	IF condition(s) **THEN**
	Begin
	Instructions 1 **;**
	End ;

Exemple:

```
program ExempleIf;
var  nombre: integer;
begin
 writeln('Entrez un nombre : ');
 readln(nombre);
  if nombre > 0 then
   writeln ('Le nombre est positif');
end.
```

II.7.2 Structures *alternatives* :

Dans le déroulement d'un algorithme, on doit souvent choisir entre deux actions, suivant une condition concernant la valeur de certaines données. La structure alternative va permettre d'effectuer des choix.

Syntaxe

Si <condition> Al***ors***

Instructions 1
Instructions 2
Sinon

Instructions 1
Instructions 2
Finsi

En Pascal

IF condition(s) **THEN**
Begin
Instructions 1 **;**
Instructions 2 **;**
End

Else

Begin
Instructions 1 **;**
Instructions 2 **;**
End ;

Exemple:

```
program ExempleIfElse;
var age: integer;
begin
  writeln('Quel est votre âge ? ');
  readln(age);
  if age >= 18 then
    writeln('Vous êtes majeur.')
  else
    writeln('Vous êtes mineur.');
end.
```

II.7.3 Structures conditionnelles imbriquées :

Un test est une instruction qui permet d'effectuer un traitement différent selon qu'une condition est vérifiée ou non. Le test imbriqué est une généralisation de la structure de contrôle conditionnelle, lorsque le nombre de traitements différents est plus grand que deux**.**

Syntaxe

```
SI condition1 ALORS
debut
   traitement1 ;
fin
SINON
debut
   SI condition2 ALORS
   debut
   Traitement2 ;
   fin
   SINON
   debut
        SI condition3 ALORS
        debut
        Traitement3 ;
        fin
        SINON
        debut
             SI condition4 ALORS
             debut
             Traitement4 ;
             fin SINON
                 ..................

             FIN
   FIN
FIN
```

En Pascal

```
IF condition1 THEN
begin
    Trait1;
 end
ELSE
begin
   IF condition2 THEN
   Begin
   Trait2 ;
   end
   ELSE
   begin
      IF condition3 THEN
      begin
      Trait3 ;
      end
      ELSE
        begin
           IF condition 4 THEN
           begin
           Trait 4 ;
            ELSE
           ..................
          End ;
    End ;
End ;
```

Exemple:

```
program ExempleElseIf;
var   note: integer;
begin
 writeln ('Entrez une note : ');
readln(note);
 if note >= 90 then
 writeln('A')
else if note >= 80 then
 writeln('B')
else if note >= 70 then
```

writeln('C')

else if note >= 60 **then**

writeln('D')

else

writeln('E');

end.

II.7.4 structure alternative multiple:

La structure choix permet de choisir le traitement à effectuer en fonction de la valeur ou de l'intervalle de valeurs d'une variable ou d'une expression. Cette structure permet de remplacer avantageusement une succession de structures SI ... ALORS.

Syntaxe	**En Pascal**
choix expression **dans** valeur1 : trait1 ; valeur2 : trait2 ; valeur5, valeur8 : trait3 ; valeur10..valeur30 : trait4 ; valeur n-1 : trait n-1 ; Sinon trait n ; **FINchoix**	**CASE** expression **OF** **Valeur1 : trait1 ;** **valeur2 : trait2 ;** **valeur 5, valeur 8 : trait3 ; valeur 10..valeur 30 : trait4 ; ...** **valeur n-1 : trait n-1 ;** **Else trait n ;** **END ;**

Exemple:

program CaseExample;

var day: **integer;**

begin

writeln('Enter a number (1-7) to represent a day of the week: '); **readln**(day);

case day **of**

1: **writeln**('Sunday');

2: **writeln**('Monday');

3: **writeln**('Tuesday');

4: **writeln**('Wednesday');

5: **writeln**('Thursday');

6: **writeln**('Friday');

7: **writeln**('Saturday')

else

writeln('Invalid input. Please enter a number from 1 to 7.');

end; end.

II.7.5 Exercices d'application:

Exercice 1

Ecrire un programme Pascal qui permet de tester si un nombre entier donne par l'utilisateur est pair ou non.

Solution:

```
PROGRAM pair_impair;
USES crt ;
VAR  n :integer ;
Begin
  Readln(n);
  If (n mod 2 <> 0) then
write(n, ' est impair');
  If (n mod 2 = 0) then
write(n, ' est pair');
End.
```

Exercice 2

Ecrire un programme Pascal qui permet d'afficher le minimum de trois valeurs r´eelles distinctes donn´ees par l'utilisateur.

Solution:

```
PROGRAM Min;
USES crt ;
VAR  a,b,c :integer ;
Begin
  Read(a,b,c);
  If (a < b)and (a < c)then write('Le minimum est: ',a) ;
If (b < a)and (b < c) then write('Le minimum est: ',b);
  If (c < a)and (c < b) then write('Le minimum est: ',c);
End.
```

Exercice 3

Ecrire un programme Pascal qui demande à l'user d'entrer la note est qui affiche la mention comme suite :

Faible ; si note<=10. Passable ; si 10<=note<=12. A.Bien ; si 12<=note<=14.

Bien ; si 14<=note<=16. T. Bien; si 16<=note<=18. Excellent ; si 18<=note<=20 .

Solution:

```
PROGRAM mention;
USES crt ;
VAR  note:Real ;
Begin
Writeln (‘entrer la note :’) ;Readln (note);
if (note <10) then
Writeln ('faible')
Else  if (note <14) then
Writeln ('A. Bi en')
Else  if(note <16) then
Writeln ('Bien')
Else    if(note <18) Then
Writeln ('T. Bien')
Else
Writeln ('Excellent');
End.
```

Exercice 4 :

Ecrire un programme Pascal permettant de résoudre une équation du second degré.

Solution:

```
PROGRAM second_degré;
USES crt ;
VAR  a, b, c, delta:Real ;
Begin
Writeln (‘saisissez les valeurs a, b et c’) ;Readln (a, b, c);
If (a=0) then Writeln (‘équation du premier degré’)
Else
delta=b^2 - 4*a*c;
if (delta>0) then
    begin
      Writeln (‘les solutions de l’équation sont’);
       Writeln (-b-sqrt(delta))/(2*a), (-b+sqrt(delta))/(2*a));
     end
       Else if (d=0) then  Writeln (-b/ (2a))
          Else Writeln (‘pas de solutions réelles’);
 Readkey();  End.
```

Exercice 5:

Ecrire un programme en Pascal pour gérer différentes alternatives d'un menu.

Solution:

```
program CaseMenu;
Uses crt ;
var choix: integer;
begin
  writeln('Menu :');
  writeln('1. Option 1');
  writeln('2. Option 2');
  writeln('3. Option 3');
  writeln('4. Quitter');
  writeln('Faites votre choix : ');
  readln(choix);
  case choix of
    1: writeln('Vous avez choisi l' 'Option 1');
    2: writeln('Vous avez choisi l' 'Option 2');
    3: writeln('Vous avez choisi l' 'Option 3');
    4: writeln('Vous avez choisi de quitter le menu');
  else
    writeln('Choix non valide');
  end;
Readkey();
end.
```

Exercices supplémentaire

Exercice 1

Ecrire un algorithme puis la traduction en Pascal d'un programme intitulé Cylindre, qui calcule et affiche le volume d'un cylindre après saisie son rayon R et sa hauteur H.

Exercice 2

Ecrire un algorithme puis la traduction en Pascal d'un programme intitulé Permut, qui fait la permutation de deux variables A et B.

Exercice 3

Proposer une marche à suivre qui fait, une permutation circulaire à droite, des valeurs de trois variables A, B et C. Par exemple : à partir de (A, B, C) = (10, 25, 4), on passe à (A, B, C) = (4, 10, 25).

Exercice 4

Proposer une marche à suivre qui fait la permutation de deux variables entieres X et Y, sans faire appel à aucune variable intermédiaire.

```
Algorithme permut ;
Var x,y :entier ;
Début
   Lire (X) ;
   Lire (Y) ;
   X ←X+Y ;
   Y ←X-Y ;
   X ←X-Y ;
   Ecrire (X, "",y) ;
 Fin .
```

Exercice 5

Ecrire un algorithme puis la traduction en Pascal d'un programme, qui convertit en heures, minutes et secondes, une durée T donnée en secondes. Il affiche le résultat sous la forme digitale comme celle d'une montre électronique (hh : mn : ss).

Exercice 6

On sait qu'avec un réservoir de L litres, une voiture a parcouru Y km. Ecrire un algorithme puis la traduction en Pascal d'un programme, qui fait lire les données nécessaires et fait calculer et afficher le taux de consommation aux 100 km de cette voiture.

Exercice 7

Soit l'unité d'enseignement méthodologie contenant les matières suivantes: tpphys (coefficient=1, crédit=2); tpchim (coefficient=1, crédit=2); inf (coefficient=2, crédit=4); methred (coefficient=1, crédit=1).

moyenneunité= (notetpphys+notetpchim+noteinf*2+notemethred)/5. Un étudiant obtient 9 crédits si sa moyenneunité >=10, sinon il cumule(يجمع) le nombre de crédits des matières où il a eu une note supérieure ou égale à 10.

Compléter le programme suivant qui lit les notes obtenues par un étudiant dans toute les matières de l'unité et calcule le nombre de crédits qu'il a obtenu.

```
   Program cumulcredits;
Uses crt;
Var moyenneunité, notetpphys, notetpchim, noteinf, notemethred:……;
     credit:……;
  Begin
     Read(……);
```

```
    Read(……);
    Read(……);
    Read(……);
    Credit:=……..;
    moyenneunité:=……..……..…..;
    If moyenneunité…… then credit:=……..…….…….
    Else
        begin
          If notetpphys ……..then credit:=……..;
          If notetpchim ……..then credit:=……..;
          If noteinf ……..……..then credit:=…….……..;
          If notemethred ……...then credit:=……..……..;
        End;
   Writeln('credits cumulés est:',……);
End.
```

Exercice 8

Compléter le programme qui demande une date sous la forme de 2 nombres entiers (numéro du jour et numéro du mois) et affiche la saison (ex : 12/02 : hiver). On supposera que l'hiver correspond aux mois de décembre, janvier et février, le printemps aux mois de mars, avril et mai, l'été aux mois juin, juillet et aout, enfin l'automne aux mois septembre, octobre et novembre).

```
Program saison ;
 Uses crt ;
 var      jour, mois : ……. ;
begin
 writeln('Quel est le jour ?') ;
 read( …….) ;
 writeln('Quel est le mois (entre 1 et 12)') ;
 read(……..) ;
 if (……...…..…….……..…….. ) then writeln('…….')
 else
 begin
   if (……..…..…….. ) then writeln('…….')
   else
   begin
     if (……...…..……...…….…….. ) then writeln('…….')
     else     writeln ('…….. ') ;
```

```
    end;
end;
end.
```

Exercice 9

Un magasin de reprographie facture une photocopie à 5 DA pour les dix premières, 3 DA pour les vingt suivantes et 2 DA au-delà. compléter le programme qui demande à l'utilisateur le nombre de photocopies et qui affiche la facture correspondante.

```
Program facture ;
 Uses crt ;
 Var n,f:……. ;
Begin
  Read(……);
  If ……..…… then …….....……
  Else
 if …… then ……
  Else ……..;
 Writeln('la facture est de:', …….);
End.
```

II.8 Les structures de contrôle répétitives:

Une structure de répétition, également appelée structure itérative, implique la répétition d'une séquence d'actions dans un ordre spécifique, un nombre précis de fois ou un nombre indéterminé de fois. On peut également se référer à une structure itérative comme une boucle.

Cependant, il existe deux scénarios distincts :

- Lorsque le nombre de répétitions est prédéterminé à l'avance, nous parlons de boucles itératives.
- Lorsque le nombre de répétitions n'est pas connu à l'avance ou peut varier, nous faisons référence aux boucles conditionnelles.

II.8.1 Boucle For (Pour):

En Pascal, la boucle **FOR** est une structure de contrôle répétitive qui permet d'itérer sur une séquence de nombres ou d'effectuer un ensemble d'instructions un nombre spécifique de fois

Syntaxe	**En Pascal**

pour variable← valeur de_départ **à** valeur_de_fin
faire
Instructions1;
Instructions2;
…..
Fin pour;
	for variable **:=** valeur_de_départ **to** valeur_de_fin **do**
	begin
	Instructions1;
	Instructions2;
	…..
	end;

Avec:

variable : Il s'agit d'une variable qui sert de compteur ou de variable de contrôle pour la boucle. La variable est initialement définie à valeur_de_départ.

valeur_de_départ : C'est la valeur initiale de la variable de contrôle. La boucle commence avec cette valeur.

valeur_de_fin : C'est la valeur à laquelle la variable de contrôle doit atteindre pour que la boucle se termine. La boucle continuera d'itérer tant que la variable de contrôle est inférieure ou égale à valeur_de_fin.

do : Indique le début du bloc d'instructions qui sera répété.

Instructions : Les instructions entre les mots-clés begin et end sont exécutées à chaque itération de la boucle.

Exemple:

```
program BoucleForExemple;
Uses crt ;
var i: integer;
begin
  for i := 1 to 6 do
  begin
    writeln('Valeur de i : ', i);
   end;
Readkey();
end.
```

Dans cet exemple, la boucle "for" itère sur les valeurs de 1 à 6 pour la variable i. À chaque itération, elle affiche la valeur de i. Le résultat sera :

Valeur de i : 1. Valeur de i : 2. Valeur de i : 3.

Valeur de i : 4. Valeur de i : 5. Valeur de i : 6.

II.8.2 Boucle While (Tan-que):

La boucle **while** est une structure de contrôle répétitive qui permet d'exécuter un bloc d'instructions tant qu'une condition spécifiée est vraie.

Syntaxe	**En Pascal**
Tant-que condition **faire**	**while** condition **do**
debut	**begin**
Instructions1;	Instructions1;
Instructions2;	Instructions2;
…..	…..
FinTant-que;	**end;**

Avec:

Condition: est une expression booléenne. Tant que cette condition est vraie, la boucle "while" continue d'exécuter les instructions à l'intérieur de son bloc.

do: indique le début du bloc d'instructions qui sera répété.

Instructions : Les instructions situées entre les mots-clés begin et end sont exécutées à chaque itération de la boucle, tant que la condition est vraie.

Exemple:

```
program BoucleWhileExemple;
Uses crt ;
var  j: integer;
begin
 j := 1;
while j <= 6 do
 begin
writeln('Compteur : ', j);
 j := j+ 1;
end;
Readkey();
end.
```

Dans cet exemple, la boucle "while" itère tant que la variable j est inférieure ou égale à 6. À chaque itération, elle affiche la valeur du compteur et incrémente le compteur de 1. Le résultat sera:

Compteur : 1. Compteur : 2. Compteur : 3.

Compteur : 4. Compteur : 5. Compteur : 6.

II.8.3 Boucle Repeat (Répéter):

La boucle Repeat est une structure de contrôle répétitive qui permet d'exécuter un bloc d'instructions au moins une fois, puis de continuer à l'exécuter tant qu'une condition spécifiée est vraie.

Syntaxe	**En Pascal**
Répéter	**Repeat**
Instructions1;	Instructions1;
Instructions2;	Instructions2;
…..	…..
jusqu'à condition;	**until** condition;

Avec:

Instructions : Les instructions à l'intérieur du bloc "repeat" et "until" sont exécutées au moins une fois, puis répétées tant que la condition spécifiée est vraie.

Condition: (exprimée après le mot-clé "until") est une expression booléenne. Tant que cette condition est vraie, la boucle continue d'exécuter les instructions du bloc "repeat".

Exemple:

```
program BoucleRepeatExemple;
Uses crt ;
var  i: integer;
begin
i := 1;
 repeat
  writeln('Compteur : ', i);
     i := i + 1;
 until i > 6;
Readkey();
end.
```

Dans cet exemple, la boucle "repeat" exécute le bloc d'instructions au moins une fois, puis répète ce bloc tant que la variable i est inférieure ou égale à 6. À chaque itération, elle affiche la valeur du compteur et incrémente le compteur de 1. Le résultat sera:

Compteur : 1. Compteur : 2. Compteur : 3.
Compteur : 4. Compteur : 5. Compteur : 6.

II.8.4 Exercices d'application:

Exercice 1

Utiliser la boucle **for** pour calculer la somme des entiers de 1 à n.

Solution:

```
program Some_n;
Uses crt ;
var n, som: integer;
begin
  writeln('Entrez un nombre entier n : ');
  readln(n);
som := 0;
  for i := 1 to n do
  begin
    som := som + i;
  end;
writeln('La somme est ', som);
Readkey() ;
end.
```

Exercice 2

Utiliser la boucle **while** pour trouver le premier multiple de 7 supérieur à 100.

Solution:

```
program PremierMultipleDe7;
Uses crt ;
var       nombre: Integer;
begin
  nombre := 100;
  while nombre mod 7 <> 0 do
    nombre := nombre + 1;
  writeln('Le premier multiple de 7 supérieur à 100 est : ', nombre);
Readkey() ;
end.
```

Exercice 3

Utiliser la boucle **Repeat** pour demander à l'utilisateur de saisir un mot de passe correct.

Solution:

```
program mot_de_passe;
Uses crt ;
var motDePasse, saisie: string;

begin
  motDePasse := 'secret';
  repeat
    writeln('Entrez le mot de passe : ');
    readln(saisie);
  until saisie = motDePasse;
  writeln('Mot de passe correct. Accès autorisé.');
Readkey() ;
end.
```

Exercice 4

Soit la suite u définie par $U_1 = 1$ et pour tout $n \in N^*$, $U_{n+1} = U_n + 1/n$. Ecrire un programme en Pascal Qui calcule et affiche le terme U_{30}

Solution:

```
Program suite ;
Uses crt ;
Var i :integer;
U :real;
Begin
U:=1;
For i:=1 to 29 do
U:= U+(1/i) ;
Writeln('le terme U30 est:', u) ;
Readkey() ;
End.
```

Exercice 5

Ecrire un programme qui demande un réel x et un entier naturel n positif ou nul, puis qui calcule x^n.

Solution:

```
PROGRAM puissance;
Uses crt;
VAR x,p : real ;
 n,k : integer ;
```

```
BEGIN
WRITE ('Donner x et n'); readln(x); readln(n);
p:=1;
IF (n>=0) THEN
  begin
    FOR i:=1 TO n DO
     p:=p*x;
    WRITELN (x:1:2,' ^',n,'=',p:1:2);
   end
   else writeln(' valeur de n doit être positive ou nul');
 Readkey() ;
END.
```

Exercice 6

Écrivez un programme en Pascal qui demande à l'utilisateur de saisir un entier n et renvoie le résultat de "n !" (n factoriel) ,utilise d'abord la boucle "for" puis la boucle "while".

Solution:

1.Pour la boucle For:

```
PROGRAM fact_FOR;
Uses crt;
VAR n,k,fact : INTEGER;
BEGIN
WRITELN('Entrer un entier n positif') ;
 READLN(n) ;
fact:= 1 ;
FOR k:=1 TO n DO
fact:=fact*k ;
 WRITELN(n, '!= ' , fact);
Readkey() ;
END.
```

2.Pour la boucle While:

```
PROGRAM fact_while;
Uses crt;
VAR n,i,fact : INTEGER;
BEGIN
WRITELN('Entrer un entier n positif') ;
 READLN(n) ;
```

```
fact:= 1 ;
i := 1;
   while i <= n do
   begin
     fact := fact * i;
     i := i + 1;
   end;
WRITELN(n, '!= ' , fact);
Readkey() ;
END.
```

II.9 Les fonctions plus utilisées en Pascal :

Il existe de nombreuses fonctions standard intégrées qui vous permettent de réaliser diverses opérations, telles que la manipulation de chaînes de caractères, la gestion des dates et heures, l'interaction avec l'utilisateur, etc. Voici quelques-unes des fonctions standard couramment utilisées en Pascal :

writeln / write : Ces fonctions sont utilisées pour afficher du texte à l'écran. writeln ajoute automatiquement un saut de ligne à la fin, tandis que write n'ajoute pas de saut de ligne.

read / readln : Ces fonctions sont utilisées pour lire des données à partir de l'entrée utilisateur. readln lit une ligne entière et la stocke dans une variable, tandis que read lit jusqu'au prochain espace ou retour chariot.

length : Cette fonction permet de déterminer la longueur d'une chaîne de caractères.

copy : Cette fonction permet de copier une partie d'une chaîne de caractères dans une nouvelle chaîne.

pos : Cette fonction renvoie la position d'une sous-chaîne à l'intérieur d'une chaîne donnée.

uppercase / lowercase : Ces fonctions permettent de convertir une chaîne de caractères en majuscules ou en minuscules, respectivement.

strToInt / strToFloat : Ces fonctions permettent de convertir une chaîne de caractères en un nombre entier ou en un nombre à virgule flottante.

intToStr / floatToStr : Ces fonctions permettent de convertir un nombre entier ou un nombre à virgule flottante en une chaîne de caractères.

random : Cette fonction génère un nombre aléatoire dans une plage donnée.

date / time : Ces fonctions renvoient la date et l'heure actuelles.

now : Cette fonction renvoie la date et l'heure actuelles sous forme de valeur DateTime.

abs : Cette fonction renvoie la valeur absolue d'un nombre.

round / trunc : Ces fonctions permettent d'arrondir un nombre à la valeur entière la plus proche, en utilisant différentes méthodes.

sqrt : Cette fonction calcule la racine carrée d'un nombre.

SQR: Cette fonction est utilisée pour calculer le carré d'un nombre x. Elle renvoie le résultat du carré de x.

sin / cos / tan : Ces fonctions renvoient les valeurs trigonométriques du nombre spécifié (en radians).

Arctan: Cette fonction est utilisée pour calculer l'arc tangente (ou inverse de la tangente) d'un nombre x. L'arc tangente renvoie l'angle en radians dont la tangente est égale à x.

Round: Cette fonction est utilisée pour arrondir un nombre réel x à la valeur entière la plus proche. Si la partie décimale de x est égale ou supérieure à 0,5, alors Round(x) renverra la valeur entière supérieure. Si la partie décimale est inférieure à 0,5, alors Round(x) renverra la valeur entière inférieure.

Trunc: Cette fonction est utilisée pour tronquer (ou "tronquer vers zéro") un nombre réel x. Cela signifie que Trunc(x) renvoie la partie entière de x en supprimant la partie décimale. Contrairement à la fonction Round qui arrondit à l'entier le plus proche, Trunc ne fait pas d'arrondi et se contente de supprimer la partie décimale.

Exp: Cette fonction est utilisée pour calculer l'exponentielle d'un nombre x. L'exponentielle d'un nombre x est égale à e^x, où e est la base de logarithme naturel, une constante mathématique d'environ 2,71828.

Ln: Cette fonction est utilisée pour calculer le logarithme naturel (ou logarithme népérien) d'un nombre x. Le logarithme naturel d'un nombre x est noté comme ln(x).

Ces fonctions standard font partie de la bibliothèque standard de Pascal et sont largement utilisées dans le développement de programmes en Pascal pour effectuer différentes tâches.

Exercices supplémentaire

Exercice 1

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.

Exercice 2

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message :

« Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

Exercice 3

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Exercice 4

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de 7 : 7 x 1 = 7

7 x 2 = 14

7 x 3 = 21 …

7 x 10 = 70.

Exercice 5

Ecrire un algorithme qui demande un nombre positif de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer : 1 + 2 + 3 + 4 + 5 = 15

NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Exercice 6

Ecrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle. NB : la factorielle de 8, notée 8 !, vaut 1 x 2 x 3 x 4 x 5 x 6 x 7 x 8

Exercice 7

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres :

Entrez le nombre numéro 1 : 12

Entrez le nombre numéro 2 : 14 etc.

Entrez le nombre numéro 20 : 6

Le plus grand de ces nombres est : 14

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre : C'était le nombre numéro 2.

Bibliographie

1-Cours et exercices corriger d'algorithmique, J.Tulliand Ed Vuibert Fev 2010.

2- Architecture et technologie des ordinateurs 4eme édition, P. ZANELLA, DUNOS 1991.

3- Introduction à l'algorithmique, T. CARMEN, DUNOD, 1994.

4- Analyse et mise en œuvre arithmétique des ordinateurs, M. LAPORTE, E. TECHNIP, 1974.

5- Exercices et problèmes d'algorithmique, B. BAYNAT, DUNOD, 1991.

6- Support de cours en Informatique o2, Algorithmique et programmation, T. Nateche, USTO 2017.

7- Support de cours en Informatique 01, Algorithmique et programmation, MADANI FOUATIH, USTO 2019.

8- Support de cours en Informatique 01, Algorithmique et programmation, R. BENGHEZAL,2020.

9- John Paul Mueller et Luca Massaron, Les algorithmes pour les Nuls grand format, 2017.

10- Charles E. Leiserson, Clifford Stein et Thomas H. Cormen, Algorithmique: cours avec 957 exercices et 158 problèmes, 2017.

11- Thomas H. Cormen, Algorithmes: Notions de base, 2013.

12- Structures conditionnelles [if], K. ZAMPIERI, Unisciel Algoprog, 2018.

13- Algorithmes D.E Knuth CSLI Publications 2011.

14- Support de cours en Informatique 01, Algorithmique et programmation, Bekkouche Souad,2018.

15- Support de cours en Informatique I & II, ALGORITHMIQUE I & II, Le ̈ila BOUSSAAD,2018.

16- INITIATION A TURBO PASCAL, J.P. CHEVAUX ,2003.

17- Algorithmique et programmation en Pascal, Djelloul Bouchiha, , Editions Universitaires Européennes, 2020.