UEO11 Cours/TDn°1 – Algorithmique : bases - 2011-2012

Table des matières

1-Introduction

L’informatique

C'est la Science du traitement de l’**infor**mation auto**matique**ment

Ensemble de techniques de collecte, de tri , de mise en mémoire, de transmission et de l’utilisation des informations traitées automatiquement à l’aide de programmes mis en œuvre sur ordinateur.

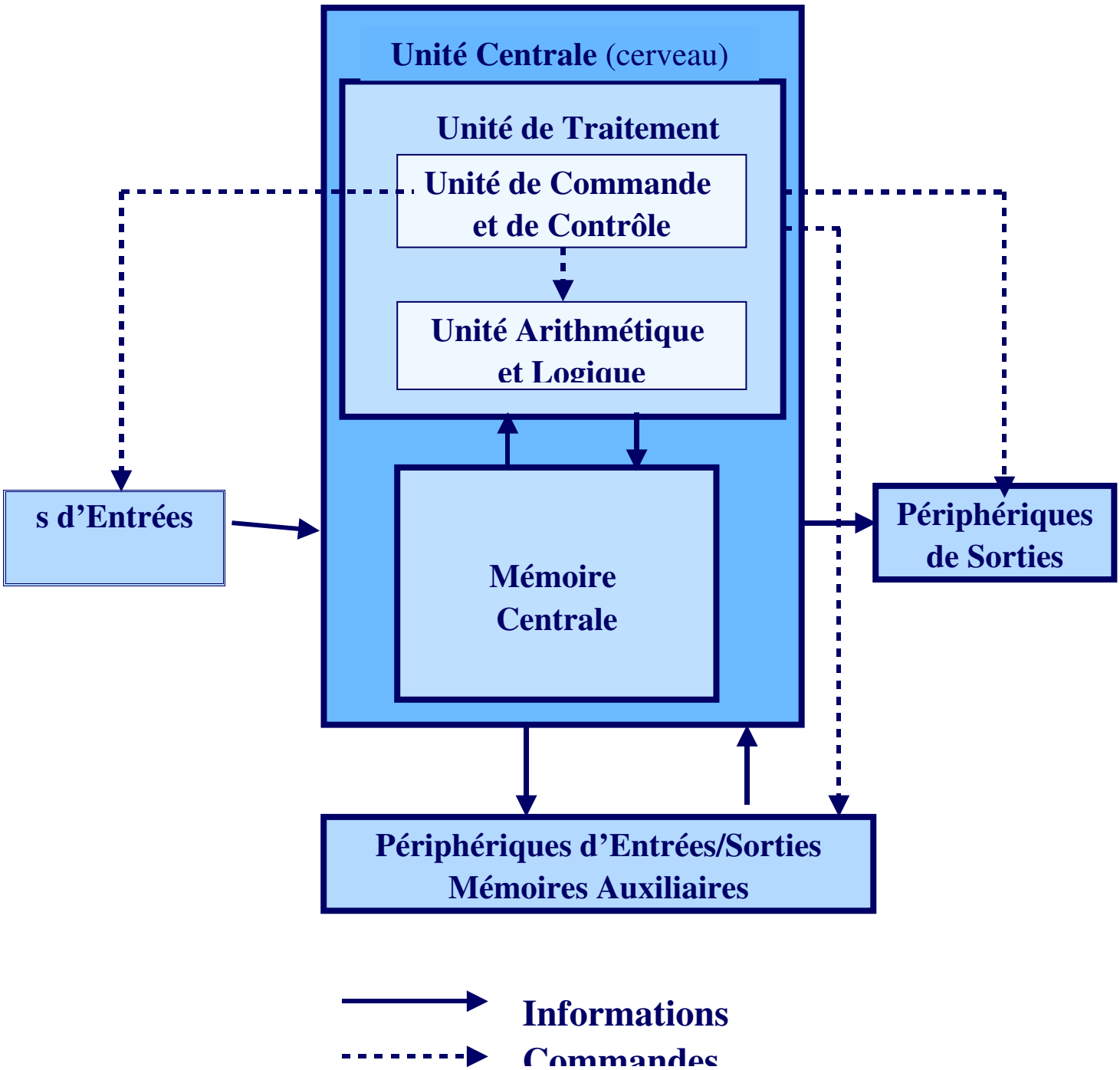
Exemple d’informations :

nombres entiers et réels codés en mémoire centrale. Caractères alphabétiques et caractères spéciaux….

Le traitement de l’information consiste à partir d’une information de départ de produire par traitement une nouvelle information de sortie.

L’ordinateur

Un ordinateur est une machine qui permet d'effectuer des traitements sur des données à l'aide de programmes. Les données (les paramètres du problème, par exemple les notes des étudiants du cours d'informatique) ainsi que le programme (par exemple le calcul de la moyenne des notes) sont fournis à la machine par l'utilisateur au moyen de dispositifs de saisie (le clavier par exemple). Le résultat du traitement est recueilli à la sortie de l'ordinateur (l'écran, l’imprimante) sous forme de texte par exemple.


Unité Centrale (cerveau)
Unité de Traitement
Unité de Commande et de Contrôle
Unité Arithmétique et Logique
Mémoire Centrale
s d'Entrées
Périphériques de Sorties
Périphériques d'Entrées/Sorties Mémoires Auxiliaires
Informations
Commandes

2- Algorithmique

2.1 Définition : algorithmique

Critère algorithmique élémentaire

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

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

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

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

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

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

Opérations séquentielles

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

2.2 De l' algorithme au programme

Un peu d'histoire :

Le langage C peut être qualifié de bas niveau ou peu typé dans le sens où le langage est conçu pour permettre la manipulation directe des mots machine.

Le langage C a été utilisé pour rendre le système d'exploitation UNIX plus portable. Il a conservé de cela une très grande efficacité pour tout ce qui concerne le développement système.
Ainsi la majorité des grands systèmes d'exploitation entre les années 1975 et 1993 ont été programmés à l'aide de ce langage.

Le langage C est apparu au cours de l'année 1972 dans les Laboratoires Bell . Il était développé en même temps que Unix par **Dennis Ritchie** (tjours aux labos Bell) et **Ken Thompson (chez Google depuis 2006)**.

En 1983, l'**Institut national américain de normalisation** (ANSI) a formé un comité de normalisation du langage qui a abouti en 1989 à l'approbation de la norme dite **ANSI C** ou C89 (formellement ANSI X3.159-1989). En **1990**, cette norme a également été adoptée par l'**Organisation internationale de normalisation** (**C ISO**, formellement ISO/CEI 9899:1990). ANSI C est une évolution du C K&R qui reste extrêmement compatible. Elle reprend quelques idées de **C++**.

En **1999**, une nouvelle évolution du langage est normalisée par l'**ISO** : C99 (formellement ISO/CEI 9899:1999). Parmi les ajouts, on notera les tableaux dynamiques, ainsi que des fonctionnalités (types **complexes**, mot-clef « restrict », directives agissant sur la simplification des instructions arithmétiques) souhaitables pour les calculs numériques intensifs, domaine habituel de **FORTRAN**.

Le langage C reste un des langages les plus utilisés actuellement. Cela est dû au fait qu'il comporte des instructions et des structures de haut niveau tout en générant un code très rapide grâce à un compilateur très performant.

La rapidité des programmes écrits en C est en grande partie due au fait que le compilateur *présuppose* que le programmeur sait ce qu'il fait : il génère un code ne contenant pas de vérifications sur la validité des pointeurs, l'espace d'adressage, etc.

Ainsi, les programmes en C sont très compacts.

De plus, une des caractéristiques du C est qu'il est un langage « faiblement typé » : les types de données qu'il manipule sont très restreints, et proches de la représentation interne par le processeur : par exemple, le type 'Chaîne de caractères' n'existe pas en C. A l'inverse, comparer un entier et un caractère a un sens en C car un caractère est bien représenté en interne par le processeur par une valeur de type entier (le code ASCII ou le code EBCDIC).

Définition : programme

Un programme est le codage d'un algorithme afin que l'ordinateur puisse exécuter les actions décrites dans l'algorithme. Ce dernier doit être écrit dans un langage « **compréhensible** » par l'ordinateur dit langage de programmation (Assembleur (micropro), C, Fortran, Pascal, Cobol …).

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

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

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

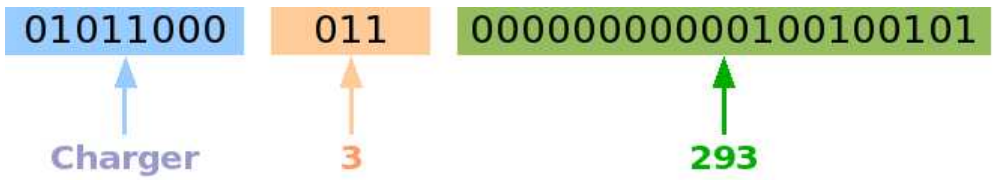


Schéma : Exemple d'instruction en langage machine
(Charger le nombre 293 dans le registre 3)

Ce type de code binaire est le seul que la machine puisse directement comprendre et donc réellement exécuter.

Tout programme écrit dans un langage évolué devra par conséquent être d'abord traduit en code-machine avant d'être exécuté.

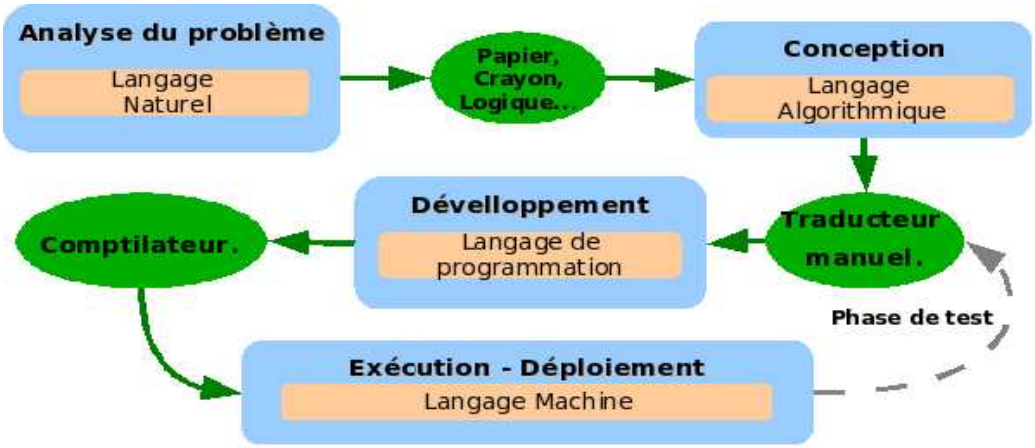


Schéma 3.2 : Cycle de vie d'un logiciel

2.3 Caractéristiques des algorithmes :

Elles permettent de formaliser le problème et de déterminer et de perfectionner les mécanismes nécessaires à sa résolution.

Plusieurs propriétés sont inhérentes à un algorithme donné et permettent de définir son utilité :

- - **La calculabilité des algorithmes** (convergence de l'algorithme)
- - **La complexité des algorithmes** (nombre d'opérations nécessaires)
- - **L'efficacité des algorithmes** (vitesse des algo: raisonnable) : temps d'exécution et taux d'occupation de la mémoire centrale

2.3.1 Méthodologie simple

Suivre la démarche suivante :

1. *Définir* clairement le problème (Analyse de l'énoncé, associer aux éléments de cet énoncé des actions simples)
2. Chercher une *méthode de résolution* (formules…)
3. Définir les *entrées* nécessaires et les *résultats* obtenus
4. Écrire l'algorithme (**langage algorithmique**)

Si le problème est trop complexe, réaliser une **analyse méthodique descendante** :

- **décomposer** en sous-problèmes,
- appliquer, pour chacun, la méthodologie simple

Décomposer un problème revient à en fournir une *description de + en + détaillée* jusqu'à obtenir un *ensemble d'opérations élémentaires* traductibles en langage de programmation

Exemple: Ecrire un algorithme calculant la circonférence d'un cercle de rayon 12

1) Définir clairement l'énoncé : relever les actions de cet énoncé (=> calculant) ainsi que les attributs associés à ces actions (=> circonférence, cercle, r=12)
2) Chercher une méthode de résolution : à partir des éléments de l'énoncé, définir la marche à suivre pour résoudre le problème (rappel de la formule de calcul de la circonférence d'un cercle C = 2*PI*R)
3) Définir les entrées nécessaires et les résultats obtenus (préciser la nature de ces entrées et sorties)

Entrées	***Sortie***
• PI (réel) (<!> *facultatif* <!>) • R rayon du cercle (entier ou réel)	Circonférence du cercle (réel)

Reste l'écriture de l'algorithme : nous allons aborder les éléments et formalismes que vous devez posséder pour réaliser cette écriture.

2.4 Langage algorithmique

Peut se définir comme une boite à outil avec un mode d'emploi permettant d'ajouter de la rigueur et de la lisibilité à un traitement donné.

On distingue plusieurs catégories d'éléments :

2.4.1 Données :

Elements introduits dans la méthode par l'utilisateur (au clavier ou dans un fichier). Ils sont désignés par un **identificateur** et possèdent un **type** clairement définies ((facultatif) peuvent également être des méthodes)

2.4.2 Constantes :

Element dont la valeur, utilisée dans le programme, ne change pas lors de l'exécution (par ex. PI). Tout comme la donnée, cet élément est désigné par un identificateur qui lui est propre et possède un type clairement défini qui ne varie pas tout au long du programme.

2.4.3 Variable:

Element dont la valeur, utilisée dans le programme, peut changer lors de l'exécution. Cette variable, elle aussi désignée par un identificateur, se référe à un emplacement précis en mémoire centrale

De façon plus générale, il est indispensable de:

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

2.4.4 Entrées / Sorties

* *L'échange de données entre programme et utilisateur*

(On peut éventuellement s'aider d'un schéma)

Lire : pour recevoir de l'information de l'extérieur

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

Écrire : pour fournir de l'information à l'extérieur

- **Écrire (x)** : la valeur contenue dans la variable x sera affichée

Exemple : Écrire ('Bonjour') : Bonjour sera écrit à l'écran

2.4.5 Opérateurs

* *L'affectation* :

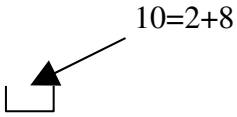
Opération qui consiste à attribuer une valeur donnée à une variable.donnée. Cette valeur peut être une donnée simple ou le résultat d'une opération ou d'une méthode.

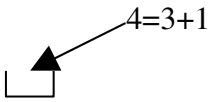
Se note : variable **<-** valeur

Exemples:

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

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



i <- i+1 (i est incrémenté de 1) i 

Representer les variables par ces petites boites avec leur nom, leur type pour tous les algos et programmes qui suivent et expliquer que chaque boite occupe un espace mémoire dont la taille depend du type de la variable.

2 classes d'opérateurs : arithmétiques et logiques

** Opérateurs arithmétiques*

Opérateurs classiques que l'on retrouve sur l'espace des réels :
- addition « + »
- soustraction « - »
- multiplication « * »
- division « / »
- modulo (reste de la div euclidienne) « % »

Attention toutefois aux types : (division : (entier)/(entier) => (entier) (3/2 => 1))

** Opérateurs logiques*

Opérateurs logiques normaux : ET, OU, NON + opérateurs de comparaison (=, >, <, >=, <=).

2.4.6 Structures

** Structure globale d'un algorithme*

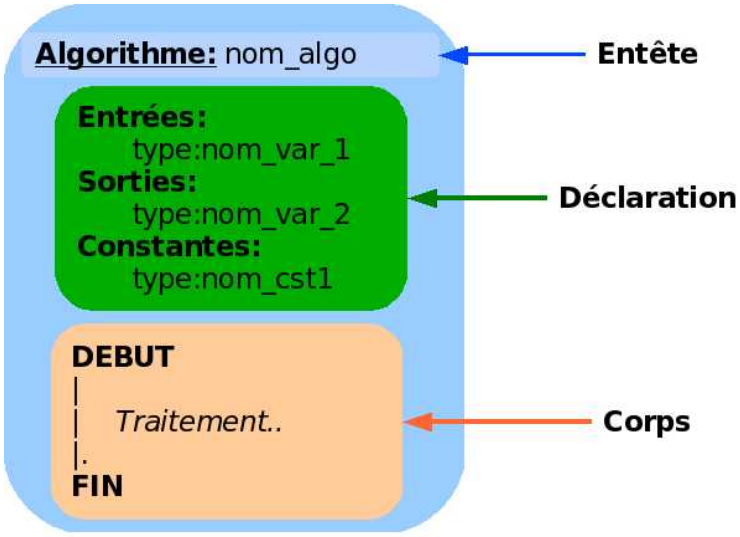


Schéma 3.3 : Formalisme d'écriture d'un algorithme

Exemple :

Retour sur le calcul de la circonférence d'un cercle de rayon 12

Voici l'énoncé exacte: Ecrire un algorithme qui calcule et affiche la circonférence d'un cercle de r=12 (r est donnée par l'utilisateur)

Rappel des entrées et sorties :

Entrées	***Sortie***
• PI (réel) (<!> *facultatif* <!>) • R rayon du cercle (entier ou réel)	Circonférence du cercle (réel)

Voici quelques questions bien utiles pour réaliser l'algorithme correspondant à partir de l'énoncé

Que doit réaliser l'ordinateur?
Quels sont les éléments que l'utilisateur doit faire rentrer dans l'ordinateur?
Quels sont les éléments à afficher?

Les réponses sont donc à chercher dans l'énoncé comme on le voit ci-dessous

Que doit réaliser l'ordinateur? (correspond aux actions à réaliser par l'ordinateur)
Quels sont les éléments que l'utilisateur doit faire rentrer dans l'ordinateur (correspond aux variables d'entrée)
Quels sont les éléments à afficher? (correspond aux variables de sortie)

Ecrire un algorithme qui *calcule* et *affiche* la **circonférence d'un cercle** de r=12 (r est donné par l'utilisateur)

Solution 1: solution simple

algorithme	Programme C (uniquement pour les SE (Sciences Exactes))

DEBUT
Écrire (2*3.14159*12)
FIN
	#include <stdio.h>
	main ()
	{
	  printf(" %0.2f\n ", 2*3.14159*12) ;
	}

Solution 2: Notion de variable x

algorithme	Programme C (uniquement pour les SE)

Variables :
  Réel : x
 	
DEBUT
  Ecrire ('tapez la valeur du rayon')
  Lire (x)
  Écrire (2*3.14159*x)
FIN
	#include <stdio.h>
	main ()
	{
	  float x ;
	  printf("tapez la valeur du rayon\n") ;
	  scanf(" %f ",&x) ;
	  printf(" %0.2f\n ", 2*3.14159*x) ;
	}

Solution 3: Utilisation de trois variables réelles

algorithme	Programme C (SE uniquement)

Variables :
  Réel : rayon, Pi,Perimetre
DEBUT
  Pi <- 3.14159
  Écrire ('Entrer la valeur du rayon:')
  Lire (rayon)
  Perimetre <- 2*Pi*rayon
  Écrire ('la circonférence du cercle est',Perimetre)
FIN
	#include <stdio.h>
	main ()
	{
	  float Rayon,Pi , Perimetre ;
	  /* lecture du rayon */
	  printf(" tapez la valeur du rayon\n") ;
	  scanf(" %f ",&Rayon) ;
	  /* initialisation de Pi /*
	  Pi=3.14159 ;
	  /* calcul du perimetre et écriture */
	  Perimetre=2*Pi*Rayon ;
	  printf("%0.2f\n", Perimetre) ;
	}

* *La structure de séquence*

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

* *La structure de sélection simple*

SI condition **ALORS**
expression1
SINON
expression2
FINSI

Exercice : Ecrire un algorithme qui calcule et affiche le Max de 2 réels A et B donnés par l'utilisateur:

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

* *Structure de répétition (nombre connu)*

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

Ecire un algo qui calcule X^n (n >0) et affiche le résultat. X est un réel donné par l'utilisateur, n est un entier (supposé strictement positif) donné par l'utilisateur et on stockera le résultat dans Y:

Pour calculer la moyenne de 16 notes:

* *Structures de répétitions (tant que condition choisie vraie)*

<!> Attention aux boucles infinies <!>

TANT-QUE condition **FAIRE**
    expression
FINTANTQUE

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

REPETER
Expression
TANTQUE condition

peut s'écrire

REPETER
Expression
JUSQUA condition opposée

Exercice même ecrire que précédamment:Pour calculer X^n (n >0) et stocker le résultat dans Y:

Exercice

Ecrire un algorithme permettant de faire le calcul suivant :
En entrée on a six entiers **sexe, an_nais, mois_nais, dept_nais, x et y** représentant le numéro de sécurité sociale de l'utilisateur (par exemple 1, 46, 04, 97, 129, 132) , et le programme produit les sorties suivantes :

« Bonjour monsieur » si c'est un homme et
« Bonjour madame » si c'est une femme

« Tu es bien petit » si son age est inférieur ou égal à 6 ans
« que fais tu la a ton age » si il a entre 6 et 16 ans
« bonjour pépé » si il a plus de 16 ans

« bonjour la Guadeloupe » si dept_ nais vaut 97 et que x est un nombre commençant par 1

Ecrire ensuite le programme C correspondant

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

Exercice

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

Exercice

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