Université Ibn Zohr
Faculté des Sciences
Agadir

Support de cours avec exercices corrigés
Programmation en C
2005-2006

Préparé

Par

Abdelaziz LABRAG

Adresse E-Mail : aziz_labrag@yahoo.fr

Tables des Matières

I. Introduction

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

Un des principaux intérêts du C est que c'est un langage très portable. Un programme écrit en C en respectant la norme ANSI est portable sans modifications sur n'importe quel système d'exploitation disposant d'un compilateur C : Windows, UNIX.

Un programme en langage C se caractérise par :

- Un fichier source (un simple fichier texte) dont l'extension est par convention (.c).
- Une fonction principale appelée main ( ) renfermant les instructions qui doivent être exécutées.
- Une manière d'écriture unique c'est-à-dire qu'un nom contenant des majuscules est différent du même nom écrit en minuscules.
- Un ajout de commentaires qui ont pour but d'expliquer le fonctionnement du programme sans que le compilateur ne les prenne en compte. Un commentaire sera donc noté de la façon suivante: /*Voici un commentaire!*/

II. Structure générale d'un programme en langage C

Un fichier source (programme) écrit en langage C est un simple fichier texte, sans mise en forme particulière ou caractère spéciaux (il contient uniquement les caractères ASCII de base) dont l'extension est par convention (.c)[1]. Il est composé d'un ensemble de fonctions, dont une est privilégiée , la fonction main (le programme principale), qui renferme les instructions qui doivent être exécutées. Celles-ci sont comprises entre des accolades qui suivent le nom de la fonction. Voici le premier exemple (inévitable !) de programme en langage C.

```
# include <stdio.h>
main ()
{ printf ("Bonjour tout le monde, c'est mon premier programme en langage C") ;
}
```

En faisant appel à la fonction prédéfinie printf (fonction de la bibliothèque standard stdio.h), ce programme nous permet d'afficher à l'écran la chaîne de

[1] L'extension est en minuscules. Le .C (majuscule) est interprété par certains compilateurs comme l'extension du C++. Comme il existe des différences entre la compilation d'un programme en C et la compilation de même programme en C++, cela peut parfois générer des problèmes.

caractères " Bonjour tout le monde, c'est mon premier programme en langage C ". Cette fonction fait partie de plusieurs fonctions situées (définies) dans un fichier nommé fichier en-tête stdio.h. Son rôle ne sera complètement compréhensible qu'ultérieurement.

Chaque fichier source est composé de :

– directives du préprocesseur (lignes commençant par #), qui seront prises en compte avant compilation,
– définitions de types,
– définitions de fonctions,
– définitions de variables,
– déclarations de variables,

a) *Points remarquables de l'exécution d'un programme en langage C*

Un programme en langage C est une suite ordonnée d'instructions. L'exécution d'un programme en C se ramène à :

i) Entrer les valeurs et les mémoriser dans ses variables.
ii) Effectuer des calculs à partir du contenu de ses variables et mémoriser les résultats dans ses variables.
iii) Afficher les contenus de ses variables.

Savoir écrire un programme en langage C, c'est savoir trouver l'ordre dans lequel il faut exécuter des instructions pour obtenir les valeurs prévues en entrée et en sortie.

b) *Les identificateurs*

Ils servent à donner un nom aux entités du programme (variables, fonctions, ...). Ils sont formés d'une suite de lettres, chiffres et du signe souligné, le premier caractère étant impérativement une lettre.

c) *Les mots clés*

Certains mots sont réservés pour le langage et ne peuvent donc pas être utilisés comme identificateurs. En voici la liste :

int	char	type	struct
auto	long	enum	register
signed	const	case	for

void	switch	break	return
goto	loat	unsigned	default
sizeof	do	while	continue
volatile	static	union	extern
double	if	short	else

III. Les types de base du langage C

C'est à partir de ces types de base que l'on peut construire tous les autres types, dits types dérivés (tableaux, structures,….)

a) *Les entiers*

Un entier est un nombre sans virgule qui peut être exprimé dans différentes bases :

- ✓ Base décimale : l'entier est représenté par dix symboles distincts se forme d'une suite de chiffre unitaires {**0**, 1, **2**, 3, 4, **5**, **6**, 7, 8,**9**}.

Pour représenter le nombre **199**, on écrit $199 = 1\ 10^2 + 9\ 10^1 + 9\ 10^0$

- ✓ Base binaire : l'entier est représenté par deux symboles distincts {**0**, **1**}.

Pour représenter le nombre $(101)_2$ en base binaire, on écrit $(101)_2 = 1\ 2^2 + 0\ 2^1 + 1\ 2^0 = 5$

- ✓ Base hexadécimale : l'entier est représenté par les éléments de l'ensemble {**0**, **1**, **2**, 3, 4, **5**, **6**, 7, **8**, **9**, A, B, C, D, E, F}. (on utilise les six premières lettres comme des chiffres)

Le mot-clé désignant les entiers est **int** qui correspond à un mot machine. Sa taille dépend donc de la taille du processeur utilisé (Elle est en général **16** ou **32** bits).

On a donc trois types d'entiers :

- **short int** (que l'on peut abréger en **short**), sont codés sur deux octets : **16** bits.
- **int**, sont codés sur quatre octets : **32** bits.
- **long int** (que l'on peut abréger en **long**), sont codés sur une longueur plus grande que les **int**.

Remarque :

Les informations traitées par un ordinateur sont représentées et manipulées sous forme de binaire. Donc le codage d'une information consiste à établir une

correspondance entre la représentation externe (habituelle) de l'information, et sa représentation interne dans la machine, qui est une suite de bits.

b) *Les caractères*

Le mot-clé désignant les caractères est **char.**

Le type char (provenant de l'anglais character) permet de stocker la valeur ASCII d'un caractère, c'est-à-dire un nombre entier! Par défaut les nombres sont signés, cela signifie qu'ils comportent un signe. Pour stocker l'information concernant le signe (en binaire), les ordinateurs utilisent le complément à deux. Une donnée de type char est donc signée, cela ne signifie pas bien sûr que la lettre possède un signe mais tout simplement que dans la mémoire la valeur codant le caractère peut-être négative... Si jamais on désire, par exemple, stocker la lettre B (sa représentation en code ASCII est 66), on pourra définir cette donnée par le nombre 66.

c) *Les réels*

Ils permettent de représenter un nombre en virgule flottante, et se divisent en trois classes:

- **Float**, sont codés sur quatre octets : 32 bits.
- **double**, sont codés sur huit octets : 64 bits.
- **long double**, sont codés sur dix octets : 80 bits

d) *Les constantes*

Chaque constante a une valeur et un type. En général on a trois genres de constantes :

- Constantes entiers
- Constantes réelles
- Constantes caractères : une constante caractère est écrite entre apostrophe.

Exemples:

'\n' provoque un saut de ligne : line feed ; '\a' provoque un beep : audible bell ; '\f' provoque un saut de page : form feed.

La déclaration des constantes permet de définir des identificateurs dont la valeur reste constante au cours de l'exécution du programme. En langage C, les constantes

sont définies grâce à la directive #define, qui permet de donner une valeur à un symbole.

Exemple :

```
#define Pi 3.14 ;
```

Cette directive remplacera tous les identifiants "Pi" existant dans votre programme par la valeur de 3.14.

Toutefois, il est possible de déclarer que la valeur d'une variable ne doit pas changer durant l'exécution du programme en utilisant le mot clef **const** signifiant constante.

Exemple :

```
Const int n=3 ;      /*  on déclare la constante n de type entier et de valeur égale à 3  */
```

IV. Les variables

Une variable[2] est caractérisée par :

- Un nom composé d'une suite de caractères commençant obligatoirement par une lettre.
- Un type qui permet de réserver l'espace mémoire nécessaire pour stocker la valeur de la variable.
- Une valeur arbitraire si la variable n'est pas explicitement affectée. Une variable une fois définie contient toujours une valeur.
- Une adresse qui désigne l'emplacement de la variable en mémoire. Cette adresse est définie une fois pour toute et ne varie pas au cours du programme. Le contenu de la variable peut être modifié autant de fois que l'on veut, ce qui ne change pas, c'est son adresse.

La syntaxe de la déclaration d'une variable est la suivante :

```
type identificateur <=expression>
```

Exemples :

```
int x=2 ;                          /* déclaration de la variable x comme entier  */
float x1, x2 ;                                  /* x1 et x2 sont des réels*/
```

[2] En langage C toute variable utilisée dans un programme doit être définie. Cette définition consiste à la nommer, à lui donner un type et éventuellement lui donner une valeur initiale si la variable n'est explicitement affectée.

V. Les fonctions de base pour les entrées /sorties

Les fonctions de base d'entrées/sorties qui vont permettre de lire des données au clavier et d'afficher des résultats à l'écran sont : la fonction **scanf** et la fonction **printf**.

a) *La fonction printf*

La fonction **printf** permet d'afficher à l'écran les arguments qu'on lui fournit.

i. Syntaxe :

```
printf("format",arg1,arg2,.....,argN);
```

L'argument format est une chaîne de caractères qui peut contenir des spécifications de format de données par le caractère spécial % suivi d'une ou plusieurs lettres significatives. Voici quelques spécifications usuelles :

Spécification de format	Signification
%d	entier
%f	réel
%c	caractère
%s	Chaîne de caractères

ii. Exemple :

```
printf(" la somme de %d et %d  vaut %d \n",2,4,2+4) ;
```

Dans cet exemple la fonction **printf** affichera à l'écran la phrase suivante :

la somme de 2 et 4 vaut 6

b) *La fonction scanf*

La fonction **scanf** permet de lire des données au clavier.

i. Syntaxe :

```
scanf("format",&arg1,&arg2,.....,&argN) ;
```

L'argument format est une chaîne de caractères qui peut contenir des spécifications de format de données par le caractère spécial % suivi d'une ou plusieurs lettres. Le symbole & devant chaque argument signifie adresse mémoire.

ii. Exemple :

scanf(" %d ",&i) ; l'adresse mémoire de la variable i, qui doit être un entier, est remplie avec ce que l'utilisateur introduira au clavier.

VI. Les expressions et les opérateurs

a) *Les expressions en langage C*

Une expression est obtenue à partir d'opérateurs, de variables, de constantes, de fonctions et de parenthèses. Une expression retourne toujours un résultat de la valeur de l'expression, qui a un type. Cette valeur retournée peut être affectée, testée ou utilisée comme opérande d'une autre expression.

Exemples :

x+3 est une expression .

2+4*x est une expression.

Dans le langage C, il y a plusieurs opérateurs. Nous décrivons ici les plus usuels, mais il faut savoir qu'il existe des priorités et des ordres d'évaluation qui régissent l'évaluation des expressions. Par exemple l'expression 2+4*x est équivalant à 2+(4*x) c'est-à-dire l'opérateur * ayant une priorité supérieure à celle de +. De même l'expression 2-3-5 équivaut à (2-3)-5, l'ordre d'évaluation de l'opérateur - étant de gauche à droite.

b) *L'opérateur d'affectation =*

L'opération la plus importante dans un langage de programmation est celle qui consiste à affecter une valeur à une variable. En langage C cette opération est notée =.

Exemples :

L'affectation x=3 ; range la valeur de 3 dans la variable x.

L'affectation x=y ; range le contenu de la variable y dans la variable x.

c) *Les opérateurs d'affectation élargie* + =, - =, * =, ….

i. Syntaxe :

Ivalue op = expression où op est l'un des opérateurs suivants :

*, /, %, +, -, <<, >>, &, ^

A op = B est équivalent à A= A op B

Remarque :

En langage C, chaque instruction se termine par ";"

ii. Exemple :

```
x+=3 ;                    /*   équivalent à   x=x+3    */
```

d) *Les opérateurs arithmétiques* +, -, *, /, %

Ils se comportent comme on s'y attend pour ce qui concerne +, - et *. Par contre, il faut faire attention avec l'opérateur /, car si les opérandes sont des flottants, le résultat est un flottant obtenu en divisant les deux nombres. Mais si les deux opérandes sont des entiers, le résultat est un entier obtenu comme résultat de la division entière. L'opérateur % n'est défini que pour les entiers et le résultat est le reste de la division entière des opérandes (modulo).

Exemples :

```
R=n%p;          /*  R est le reste de la division entière de n par p    */
Q=n/p ;         /*  Q est le quotient de la division entière de n par p  */
```

e) *Les opérateurs de comparaison* = =, !=, <=, >=, <, >.

Dans la plupart des langages de programmation le résultat d'une comparaison est un booléen (vrai ou faux). En langage C, le résultat d'une comparaison est 1 ou 0 selon que cette comparaison est vraie ou fausse respectivement.

Opérateur	Signification	Exemple	Résultat
= =	Egal à	x= =2	Retourne 1 si x est égal à 2, sinon 0
!=	Différent de	x!=2	Retourne 1 si x est différent de 2, sinon 0
<	Inférieur strictement à	x<2	Retourne 1 si x est infé.stri. à 2, sinon 0
>	Supérieur strictement à	x>2	Retourne 1 si x est sup.stri. à 2, sinon 0
<=	Inférieur ou égal à	x<=2	Retourne 1 si x est inf. ou égal à 2, sinon 0
>=	Supérieur ou égal à	x>=y	Retourne 1 si x est sup. ou égal à 2,sinon 0

Exemple :

2*a > b+3 ;

- 0 si le résultat de comparaison est fausse,
- 1 si le résultat de comparaison est vrai.

f) *Les opérateurs logiques* &&, ||, !

Opérateur	Signification	Exemple
&&	Et	x&&y

||	Ou	x||y
!	Non	!x

Comme pour les autres opérateurs, l'évaluation d'une expression booléenne obéit aux règles de priorité et d'ordre d'évaluation. Il faut préciser que l'évaluation est arrêtée dès que l'on est capable de donner la valeur de l'expression.

Exemple :

Supposons que la variable x contient la valeur 5, l'évaluation de l'expression (x>=2)||(x>y) s'arrête avant d'avoir évalué l'expression (x>y), car (x>=2) étant vrai donc on peut conclure que (x>=2)||(x>y) est vrai.

g) *Les opérateurs bit à bit* &, |, ^

Ce type d'opérateur traite ses opérandes comme des données binaires, plutôt que des données décimales, hexadécimales ou octales, ainsi effectue une opération bit à bit c'est-à-dire avec des bits de même poids.

Op.	Signification	Exemple	Résultat	Effet
&	Et bit à bit	9&12(1001&1100)	8(1000)	Retourne 1 si les deux bits de même poids sont à 1
|	Ou bit à bit	9|12 (1001|1100)	13(1101)	Retourne 1 si l'un ou l'autre de deux bits de même poids sont à 1(ou les deux)
^	Ou bit à bit exclusif	9^12(1001^1100)	5(0101)	Retourne 1 si l'un des deux bits de même poids sont à 1(mais pas les deux)

h) *Conversion de types*

Le problème de conversion de types se pose quand une expression est composée de données de natures différentes.

D'une manière générale une conversion ne peut se faire que suivant une " hiérarchie" à savoir : int ⟶ long ⟶ float ⟶ double ⟶ long double.

Cette opération peut être réalisée de deux manières :

- ✓ **Conversion implicite** : consiste en une modification du type de donnée effectuée automatiquement par le compilateur.

Exemple 1 :

```
int x ;
x = 5.8 ;
```

x contiendra après affectation la valeur de 5.

✓ **Conversion explicite**: consiste en une modification du type de donnée forcée.

Exemple 2 :

```
int x ;
x = (int) 5.8 ;
```

x contiendra après affectation la valeur de 5.

Autres Exemples : int i ; float f ; double d ; long li ;

li =f+i ;

i est transformé en float puis additionné à f, le résultat est transformé en long et rangé dans li.

d = li+i ;

i est transformé en long puis additionné à li, le résultat est transformé en double et rangé dans d.

i=f+d ;

f est transformé en double puis additionné à d, le résultat est transformé en int et rangé dans i.

i) *Incrémentation/ décrémentation*

Souvent on est amené à incrémenter ou à décrémenter une variable. Le langage C offre deux opérateurs (unairs) pour effectuer ces opérations :

++ : Incrémentation de 1

-- : Décrémentation de 1

Exemple 1 :

```
i=i+1;          /*  incrémenter de 1 le contenu de la variable i  */
i=i-1;          /*  décrémenter de 1 le contenu de la variable i  */
```

Exemple 2 :

++i ;

Incrémenter de 1 la valeur de la variable i, et sa valeur est celle de i après incrémentation.

Exemple 3:

i++;

Incrémenter de 1 la valeur de la variable i, et sa valeur est celle de i avant incrémentation.

Exemple 4 :

--i ;

Décrémenter de 1 la valeur de la variable i, et sa valeur est celle de i après décrémentation.

Exemple 5 :

i--;

Décrémenter de 1 la valeur de la variable i, et sa valeur est celle de i avant décrémentation.

Exemple 6 :

Si la valeur de i est 5, l'expression n=++i-4 affectera à i la valeur 6 et à n la valeur 2.
En revanche, si la valeur de i est 5, l'expression n=i++-4 affectera à i la valeur 6 et à n la valeur 1.

Exemple 7 :

Si la valeur de i est 7, l'expression n=--i-4 affectera à i la valeur 6 et à n la valeur 2.
En revanche, si la valeur de i est 5, l'expression n=i-- -4 affectera à i la valeur 4 et à n la valeur 1.

VII. Les instructions de contrôle

En langage C, les instructions sont exécutées séquentiellement c'est-à-dire dans l'ordre où elles apparaissent. Il dispose des instructions nommées " **instructions de contrôle** '', permettant de réaliser des choix ou d'effectuer des boucles.

Une expression suivie d'un point virgule[3] est appelée **instruction simple**. Voici un exemple d'instruction :

n=i-4 ;

[3] Le point virgule marque la fin d'une instruction.

Un bloc d'instructions est composé de plusieurs instructions délimité par deux accolades : accolade ouvrante ''{'' et accolade fermante " }".

a) *L'instruction if*

i. Syntaxe

```
if(expression)
        instruction1 ;
else
        instruction2 ;
```

si l'expression est vraie, l'instruction1 est exécutée. Sinon c'est l'instruction2 qui est exécutée. Notez que le mot "else" et l'instruction2 qu'il introduit sont facultatifs.

ii. Exemple 1 :

```
#include <studio.h>
float Moy ;
main( )
{   float Moy ;
    printf( '' entrer votre moyenne\n'') ;
    scanf ('' %f '',&Moy) ;
    if(Moy>=16 && Moy <=20)
        printf( '' mention très bien avec félicitations du jury\n'') ;
    else
        printf( '' votre mention est insuffisante pour accéder à l'école supérieure M''\n'') ;
}
```

iii. Exemple 2 : Un programme de facturation avec remise.

Ecrire un programme en langage C qui permet de lire en donnée un simple prix hors taxes et calcule le prix TTC correspondant (avec un taux de TVA constant de 20%). Il établit ensuite une remise dont le taux dépend de la valeur ainsi obtenue, à savoir :

1% pour un montant inférieur à 5000 Dh ;
5% pour un montant supérieur ou égal à 5000 Dh.

Solution

```
#include<stdio.h>
void main ()
{       double ht,ttc,net,tauxr,remise ;
     printf ("donner le prix hors taxes:\n");
     scanf ("%lf",&ht);
     ttc=(ht+(ht*20)/100);
     if (ttc<5000)              tauxr=1;
     else                       tauxr=5;
     remise=(ttc*(tauxr/100)) ;
     net=ttc-remise ;
     printf ("prix ttc est %lf Dh\n",ttc);
     printf ("remise est %lf Dh\n",remise);
     printf ("prix net à payer est %lf Dh\n",net);
}
```

b) *Les instructions itératives (répétitives): while, do…while, for*

Elles permettent d'exécuter une même séquence d'instructions un certain nombre de fois.

i. La boucle while

- **syntaxe**

```
while (condition)
        Instructions ;
```

La structure de contrôle **while** permet de répéter l'instruction tant que la condition reste vraie.

- **Exemple 1 :**

```
#include<stdio.h>
void main ( )
{       int i=1;
        while (i<=3)
    printf ("bonjour %d fois\n",i);
    i++ ;                          /*   équivalent à   i=i+1 ;   */
}
```

ii. La boucle do…while

- **Syntaxe**

```
do
Instructions ;
while(condition) ;
```

La boucle **do…while**, répète les instructions qu'elle contient tant que la condition mentionnée (condition) est vraie. Une itération est toujours exécutée.

- **Exemple 2 :**

```
#include<stdio.h>
void main ( )
{       int n;
do
{       printf ("donner un nombre  positif\n");
        scanf ("%d",&n);
        printf ("vous avez fourni %d \n",n);
}
while (n<=0);
        printf ("réponse correcte");
}
```

iii. La boucle for

- **Syntaxe**

```
for (instruction 1;condition de poursuite ;instruction 2)
instruction 3; ou bloc d'instructions délimité entre deux accolades "{"et "}"
```

Équivalent à

```
Instruction 1 ;
while(condition de poursuite)
{       instruction 3 ; ou bloc d'instructions
        instruction 2 ;
}
```

Le rôle de l'instruction **for** est de répéter le bloc (délimité par les accolades "{"et "}") figurant à sa suite en respectant les consignes suivants :

- avant de commencer cette répétition, exécuter l'instruction 1(généralement c'est une initialisation) ;
- avant chaque nouvelle exécution du bloc, examiner la condition de poursuite ;
- si elle est satisfaite, exécuter le bloc indiqué, sinon passer à exécuter l'instruction suivant ce bloc ;
- à la fin de chaque exécution du bloc, exécuter l'instruction 2.

- **Exemple 3 :**

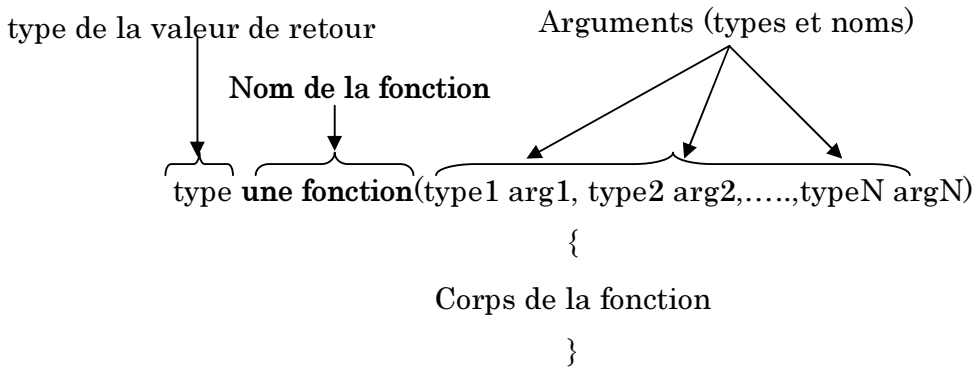
```
#include<stdio.h>
void main ( )
{       int i ;
        for (i=1 ;i<=3 ;i++)            /*      exécute 3 fois la boucle (i de 1 à 3)     */
        printf ("bonjour %d fois\n",i);
 }
```

VIII. Les fonctions

Un programme en langage C est constitué d'un ensemble de **fonctions**[4] toutes accessibles à partir d'une fonction principale appelée main(). Ces fonctions permettent d'exécuter dans plusieurs parties du programme un ensemble d'instructions par simple appel dans le corps du programme principal.

[4] Une fonction est un sous programme qui effectue une tache bien précise, à laquelle on passe des données et qui retourne une valeur ou non.

- **Syntaxe**



- **Exemple 1 :**

Calcul de la somme de deux entiers en utilisant une fonction nommée "somme"

```
#include<stdio.h>
void main()
{       int a,b;
int somme(int, int);                    /*   déclaration[5] de la fonction somme     */
printf ("tapez deux nombres entiers:\n");
scanf ("%d%d",&a,&b);
printf (" La somme de %d et %d est %d\n",a,b,somme(a,b));
}
int somme(int x, int y)                         /* appel à la fonction somme */
{       return (x+y);                           /* la valeur retournée par la fonction somme*/
}
```

[5] Avant utilisation, une fonction doit, comme les autres objets de C, être déclarée. Cette déclaration permet de préciser le type de la valeur de retour de la fonction.

- **Exemple 2 :**

Calcul du périmètre d'un cercle en utilisant une fonction nommée "perimetre"

```
#include<stdio.h>
void[6] main()
{       float R;
        float perimetre( float);
        printf("taper la valeur du rayon R du cercle en cm\n");
        scanf("%f",&R);
        printf("%f est le perimetre du cercle de rayon %f \n",perimetre(R),R);
}
float perimetre( float x)
{       return ((44*x)/7);
}
```

IX. Les tableaux

a) *Définition*

On appelle tableau une variable composées de **même type**, stockée (rangée) de façons consécutive en mémoire (les unes à la suite des autres).

Voici une façon de représenter un tableau :

donnée	donnée	…	donnée	donnée

Lorsque l'élément d'un tableau n'est pas lui-même un tableau il s'agit d'un tableau unidimensionnel. Dans le cas contraire, on parle de tableau multidimensionnel (matrice ou table).

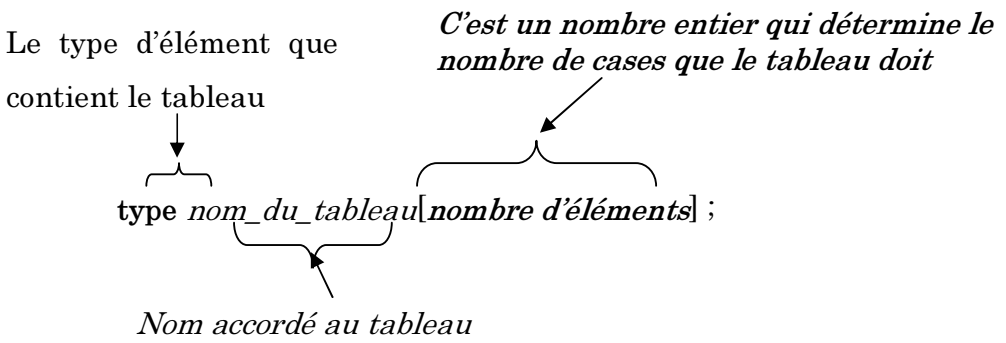
b) *Tableaux unidimensionnels*

Tableau unidimensionnel (à une dimension) est un tableau qui contient des éléments simples (d'un type donné).

i. Déclaration

En langage C, la syntaxe de la déclaration d'un tableau à une dimension est la suivante :

[6] Le type **void** précise que la fonction ne renvoie (retourne) aucune valeur.



Exemple 1:

```
int tab[5]          /*  un tableau nommé tab qui contient 5 éléments de type entier  */
char tableau[8]     /*  tableau contient 8 éléments de type caractères  */
```

ii. Comment accéder aux éléments d'un tableau unidimensionnel?

Pour accéder aux élément d'un tableau, il fallait donner le nom du tableau, suivi de l'indice de l'élément entre crochets°: Nom_du_tableau [indice]

- L'indice du premier élément du tableau est **0** ;
- Un indice est toujours positif.

Exemple 2:

```
int tab[3] ;                                /*  indice est 3  */
```

C'est un tableau de trois éléments de type entier. Donc les éléments de ce tableau sont désignés par : tab[0] ; tab[1] ; tab[2].

iii. Comment manipuler les éléments du tableau unidimensionnel?

Comme nous l'avons déjà vu, un tableau est repéré par le nom du tableau et son indice. Alors on peut manipuler tous les éléments du tableau comme étant une simple variable.

Exemple 3:

Soit le tableau de 8 éléments nommé tab :

```
int tab[8] ;                                /*  déclaration du tableau tab  */
```

Donc, pour affecter la valeur de 2 au huitième élément du tableau on écrit :

```
tab[7]=2 ;                                  /*  tab[7] est le huitième élément du tab  */
```

Pour affecter au cinquième élément du tab, le résultat d'addition des éléments 1 et 2, on écrira :

```
tab[4]=tab[0]+tab[1] ;
```

iv. Initialisation d'un tableau unidimensionnel

Il existe plusieurs façons d'initialiser un tableau :

Affecter des valeurs aux éléments du tableau un par un ;

Exemple 4 : un tableau tab de trois éléments

```
tab[0]=tab[1]=tab[2]=0;
```

- Initialiser tous les éléments au moment de la déclaration du tableau

```
int tab[3]={1,0,5};
```

- Initialiser tous les éléments grâce à la boucle for :

```
int tab[3];
int indice ;
for(indice=0 ;indice<=2 ;i++)
{       tab[indice]=0 ;
}
```

- Initialiser tous les éléments du tableau à zéro :

```
int tab[3]={0};
```

c) *Tableaux multidimensionnels*

Les tableaux multidimensionnels sont des tableaux dont les éléments sont eux même des tableaux.

Exemple 1:

```
int tab[2][3];
```

Déclaration d'un tableau de dimension 2 (2 lignes, 3 colonnes)

i. Déclaration

La déclaration d'un tableau multidimensionnel de dimension N se fait de la manière suivante :

```
type nom_du_tableau [a1] [a2]…[aN];
```

Chaque élément entre crochets désigne le nombre d'éléments dans chaque dimension.

Exemple 2:

```
int tab [3][4];
```

Déclaration d'un tableau d'entiers positifs à deux dimensions (3 lignes, 4 colonnes)

ii. Initialisation d'un tableau multidimensionnel

- Initialisation individuelle de chaque élément :

```
nom_du_tableau [0] [0]=1;
nom_du_tableau [0] [1]=2;
nom_du_tableau [0] [2]=5;
```

- Initialisation à la définition :

```
type nom_du_tableau [a1] [a2]…[aN]={0,2,1,5};
```

- Initialisation grâce à la boucle for :

Exemple 3:

Soit le tableau tab [3] [4].

On peut initialiser tous les éléments de ce tableau à la valeur **0** en utilisant la boucle for :

```
int i,j ;
for(i=0 ;i<=2 ;i++)
{       for(j=0;j<=3;j++)
        {       tab[i][j]=0 ;   }
}
```

Exemple d'application :

i. Ecrire un programme en langage C qui introduit dix nombres entiers dans un tableau.

ii. Chercher le plus grand élément et le plus petit élément de ce tableau.

Solution

```
#include<stdio.h>
void main ()
{       int i,min,max; int tab[10] ;
printf ("saisie dix entiers pour construire votre tableau:\n");
for(i=0 ;i<10 ;i++)
scanf ("%d",&tab[i]);          max=min=tab[0];
for(i=0 ;i<10 ;i++) { if(tab[i]>max) max=tab[i];
if(tab[i]<min)          min=tab[i]; }
printf ("le plus grand élément du tableau est %d:\n",max);
printf ("le plus petit élément du tableau est %d:\n",min);}
```

Exercices: Partie I

Exercice 0:

Quels seront les résultats fournis par l'exécution du programme suivant :

```
# include <stdio.h>
main ( )
{ int N = 1, P =2, Q=1,R ;
R=N= = (P=Q) ;
printf ("A : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,R ) ;
Q = N++  +  P++ ;
R = P > Q ;
printf ("B : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,R ) ;
Q = N++  +  ++P ;
R= P= =Q ;
printf ("C : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,R ) ;
Q = N++  +  P-- ;
R = P  | |  Q ;
printf ("D : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,R ) ;
P=2 ; N=3 ;
Q = N++   -   --P ;
R= P & Q ;
printf ("E : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,P&&R ) ;
Q = (--N) - (--P);
R= P && Q ;
printf ("F : N = %d  P = %d  Q = %d  | |  R = %d \n",N ,P,Q,Q&&R ) ;
Q = --N  +  P++ ;
R= N >P ? --Q+N++ : --P - Q++ ;
printf ("G : N = %d  P = %d  Q = %d && R = %d \n",N ,P,Q,P| |R ) ;
R= N < P ? –P + N++ : P++ - ++Q ;
printf ("H : N = %d  P = %d  Q = %d  R = %d \n",N ,P,Q,R ) ;
}
```

Exercice 1:

Donner un programme en langage C qui lit deux entiers **p** et **q** entrés au clavier et fait leur permutation et affiche les nombres permutés.

Exercice 2

Proposer un programme permettant de calculer et d'afficher la valeur absolue **|x-y|** de deux réels.

Exercice 3:

Donner un programme en langage C qui permet d'afficher le signe d'un nombre entier N saisi au clavier.

Exercice 4:

Ecrire un programme en langage C qui lit deux nombres entiers **n** et **p** entrés au clavier (avec $n \neq 0$ et $p \neq 0$) et écrit le reste de la division entière s'ils sont tous les deux positifs et le quotient s'ils sont tous les deux négatifs.

Exercice 5:

Ecrire un programme en langage C qui lit deux nombres réels **X** et **Y** entrés au clavier et permet d'afficher le signe de l'expression suivante : $X^2 - (Y^3+1)$.

Exercice 6:

Ecrire un programme en langage C qui lit trois nombres **x**, **y**, et **z** quelconques (Avec $x \neq y \neq z$), et trie ces nombres dans l'ordre croissant.

Exercice 7:

La mention obtenue aux examens de fin d'année dépend de la moyenne obtenue :

16<=moyenne<=20	⟹	Mention TRES BIEN
14<=moyenne< 16	⟹	Mention BIEN
12<=moyenne< 14	⟹	Mention ASSEZ BIEN
10<=moyenne< 12	⟹	Mention PASSABLE
0<=moyenne< 10	⟹	Mention INSUFFISANT

Ecrire un programme en langage C qui entre trois notes comprises entre 0 et 20 et sort la mention obtenue.

Exercice 8 :

Ecrire un programme qui lit deux valeurs entières (**m et n**) entrés au clavier et qui affiche le signe du produit sans faire la multiplication.

Exercice 9 :

Ecrire un programme qui lit deux valeurs entières (N **et** P) entrés au clavier et qui affiche le signe de la somme de N et P sans faire l'addition. Utiliser la fonction **fabs** de la bibliothèque < math.h>

Exercice 10 :

Ecrire un programme qui devra :

- Demander à l'utilisateur d'introduire au clavier un entier, qui ensuite testera si sa valeur est comprise entre **10** et **100**.
- Si le nombre est à l'extérieur de l'intervalle, affiche un message "hors limite".
- Si l'entier introduit appartient à l'intervalle, déterminer s'il est divisible par **2**, **3**, **5** ou **7**.
- Afficher un message différent pour chaque cas. N.B : plus d'un cas peut être vrai – ex. **10** est divisible à la fois par **2** et **5**.

S'il n'est pas divisible par aucune de ces quatre valeurs, afficher un message adéquat

Exercice 11 :

Ecrivez un programme qui devra :

- Demander à l'utilisateur d'entrer son année de naissance
- Vérifier si l'année est "raisonnable" (entre **1900** et **1990**)
  En cas d'erreur, on sort du programme avec un message d'erreur
- Si l'année est "correcte", afficher séparément les quatre chiffres qui la composent.
- Proposer à l'utilisateur un menu pour
  i. Afficher la somme des quatre chiffres
  ii. Afficher Le produit des quatre chiffres
  iii. Afficher l'âge de l'utilisateur

Aide : /* extraction des chiffres individuels */

void main ()

```
{       int x = 1988 ,units, tens, hundreds, thousands;
        units = x % 10;
        x = x /10;
        tens = x % 10;
        x = x /10;
        hundreds = x % 10;
        x = x /10;
        thousands = x % 10;
        printf (" %d\n%d\n%d\n%d\n", thousands, hundreds, tens, units);
}
```

L'opérateur % n'est défini que pour les entiers et le résultat est le reste de la division entière des opérandes (modulo).

Exercice 12 :

Ecrire un programme en langage C qui lit deux entiers entrés au clavier **n** et **p** et permet d'afficher le plus grand commun diviseur de ces deux nombres (PGCD).

Exercice 13 :

En utilisant l'instruction itérative for ; écrire un programme en langage C qui permet d'afficher les **dix** premiers nombres impairs.

Exercice 14 :

En utilisant l'instruction itérative for ; écrire un programme en langage C qui permet d'afficher les **vingt** premiers nombres pairs.

Exercice 15 :

Ecrire un programme en langage C qui lit deux entiers entrés au clavier **n** et **p** et permet d'afficher les nombres de n à p, **sauf** les multiples de **3** et les multiples de **5**.

Exercice 16 :

Ecrire un programme en langage C qui permet de calculer et d'afficher les sommes suivantes :

- S=1+3+5+7……+N. (Avec N est un nombre entier impair)
- S=2+4+6……+N. (Avec N est un nombre entier pair)
- $S=1+2^2+3^2+\ldots+N^2$.
- S=1+1/2+1/3+……+1/N. (Avec N est un entier strictement positif)

Exercice 17 :

Ecrire un programme qui devra :

- Demander à l'utilisateur de choisir une opération parmi l'addition ou le produit, puis de fournir deux nombres entiers.
- Afficher alors le résultat correspondant.

Exercice 18 :

Ecrire un programme en langage C qui lit N nombres entiers entrés au clavier et qui afficher leur somme, leur produit et leur moyenne. Choisir un type approprié pour les valeurs à afficher.

Exercice 19 :

Ecrire un programme en langage C qui lit un nombre entier N entré au clavier et qui permet d'afficher si l'entier introduit est un nombre premier ou non ?

Exercice 20 :

Ecrire, compiler, et exécuter un programme en langage C qui lit trois nombres x, y, et z quelconques (avec $x \neq y \neq z$), et trie ces nombres dans l'ordre croissant.

Exercice 21 :

Soit le programme suivant :

```
void main ()
{       int i=0, n,som ;
        som=0
while(i<3) ;
        {       printf ("donner un entier:\n")
        scanf ("%d",n);
        som+=n;
        i++;
    }printf ("la somme des trois entiers que vous avez saisis est \n",som);
```

i. Quelles erreurs seront détectées par le compilateur C dans ce fichier source.

ii. Après avoir corriger les erreurs, quel sera le résultat fourni par l'exécution de ce programme dans le cas où vous avez introduit les trois nombres suivants : 2, 4 et 6.

iii. Ecrire un programme réalisant exactement la même chose en utilisant la boucle **do ... while** et la boucle **for**.

Exercice 22 :

Ecrire un programme en langage C qui permet de Résoudre la fameuse équation ax^2+bx+c dans IR.

Exercice 23 (examen d'informatique - 1ère session 2003-2004):

Soit le programme suivant :

```
void main()
{       /* déclaration des variables
int i ;
printf ("tapez un nombre entier : ");
scanf ("%c",i);
if(i=0);
printf ("i est un entier nul\n",i);
else if (i>0)
printf ("i est un entier positif\n",&i);
else printf ("i est un entier négatif\n");
}
```

Corriger le programme ci-dessus pour qu'on obtienne les résultats suivants :

Première exécution :

tapez un nombre entier : **-9**

-9 est un entier négatif

Deuxième exécution :

tapez un nombre entier : **0**

0 est un entier nul

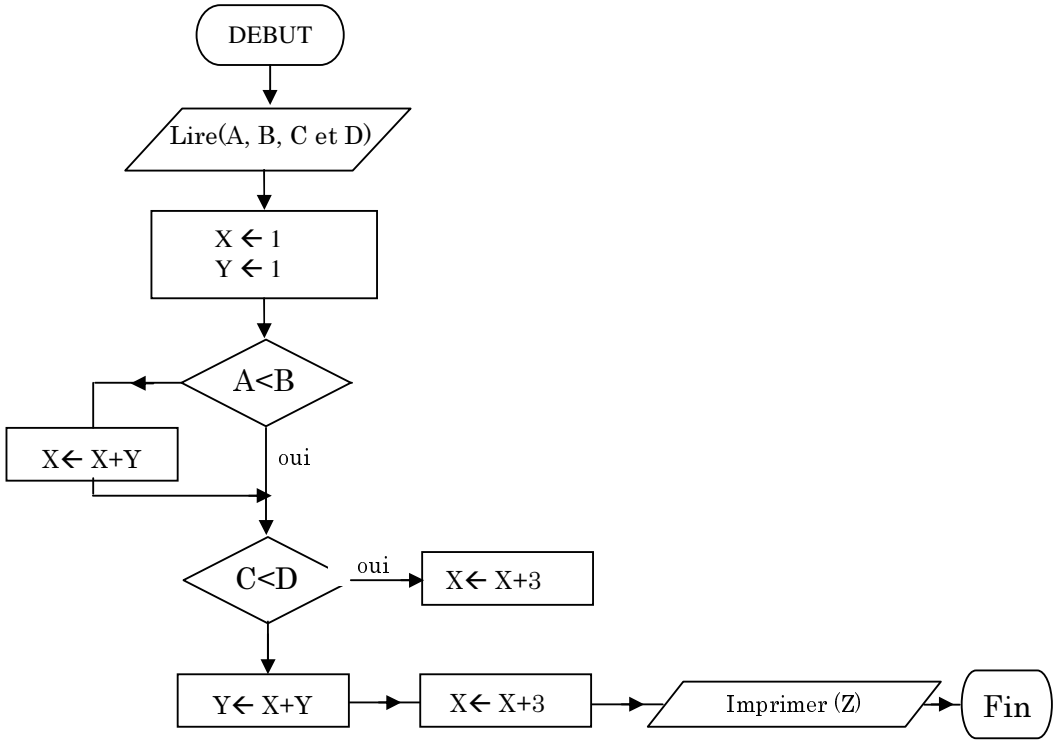
Troisième exécution :

tapez un nombre entier : 8

8 est un entier positif

Exercice 24 (examen d'informatique – session de rattrapage 2003-2004):

1. Traduire l'organigramme ci-dessous en un langage C (toutes les variables sont de type int)
2. Trouver la valeur de Z dans le cas suivant : A=2, B=3, C=3 et D=2.



Exercice 25 (examen d'informatique – session de rattrapage 2003-2004):

Ecrire un programme en langage c qui affiche le contenu du tableau

Tab[3][2]={{2,4},{20,40},{200,400}}

Sous la forme suivante :

2 4

20 **40**

200 **400**

Correction des exercices :

Exercice 0:

A : N = 1 P = 1 Q = 1 R = 1

B : N = 2 P = 2 Q = 2 R = **0**

C : N = 3 P = 3 Q = 5 R = **0**

D : N = 4 P = 2 Q = 6 R = 1

E : N = 4 P = 1 Q = 2 R = **0**

F : N = 3 P = **0** Q = 3 || R = **0**

G : N = 3 P = 1 Q = 1 && R = 1

H : N = 3 P = 2 Q = 2 R = -1

Exercice 1:

```
#include<stdio.h>
void main ()
{       int n,p,q;
    printf ("saisie deux entiers n et p:\n");
    scanf ("%d%d",&n,&p);
    q=n ;
    n=p ;
    p=q ;
printf ("La permutation de n=%d et p=%d est la suivante : n=%d et
p=%d:\n",n,p,p,n);
}
```

Exercice 2:

1 ère méthode :

```
#include<stdio.h>
void main ()
{       float x,y ;
    printf ("saisie deux réels x et y:\n");
    scanf ("%f%f",&x,&y);
    if (x= =y)
    printf ("La valeur absolue de |x-y| est nulle \n");
```

```
    else if (x>y)
        printf ("La valeur absolue de |x-y| est %.2f \n",x-y);
        else if (x<y)
        printf ("La valeur absolue de |x-y| est %.2f \n",y-x);
}
```

2 ème méthode :

```
#include<stdio.h>
void main ()
{       float x,y ;
    float VA; /* VA c'est la valeur absolue de |x-y|*/
    printf ("saisie deux réels x et y:\n");
    scanf ("%f%f",&x,&y);
    if (x= =y)
    printf ("La valeur absolue de |x-y| est nulle \n");
    else if (x>y)
        {VA=x-y;
        printf ("La valeur absolue de |x-y| est %.2f \n",VA);
        }
        else
        {VA=y-x;
        printf ("La valeur absolue de |x-y| est %.2f \n",VA);
        }
}
```

Exercice 3 :

```
#include<stdio.h>
void main ()
{       int n;
        printf ("Tapez un nombre entier n :\n");
        scanf ("%d",&n);
        if (n= =0)
        printf ("%d est un entier nul\n",n);
        else if (n>0)
```

```
  printf ("%d est un entier positif \n",n);
  else printf ("%d ne pourrais être qu'un entier négatif \n",n);
}
```

Exercice 4 :

```
#include<stdio.h>
void main ()
{ int n,p,R,Q;
  printf ("Tapez deux nombres entiers n et p avec n>p:\n");
  scanf ("%d%d",&n,&p);
  R=n%p;
  Q=n/p ;
  if (n>0 && p>0)
  printf ("Le reste de la division entière de %d par %d est %d\n",n,p,R);
  if (n<0 && p<0)
  printf ("Le quotient de %d par %d est %d\n",n,p,Q);
}
```

Exercice 5 :

```
#include <stdio.h>
void main ()
{ float X,Y,Z;
    printf ("Donner deux réels X et Y \n");
    scanf ("%f%f", &X,&Y);
    Z=X*X- (Y*Y*Y+1);
    if (Z>0)
    printf ("Le signe de l'expression X*X- (Y*Y*Y+1) est positif \n");
    else if (Z<0)
    printf ("Le signe de l'expression X*X- (Y*Y*Y+1) est négatif \n");
    else
    printf ("Le signe de l'expression X*X- (Y*Y*Y+1) ne pourrais être que nul \n") ;
}
```

Exercice 6 :

```
#include<stdio.h>
void main ()
{ float x,y,z;
printf ("Taper trois réels x # y # z :\n");
scanf ("%f%f%f",&x,&y,&z);
if (x>y)
{       if (y>z)
    printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",z,y,x);
    else
        if (x>z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",y,z,x);
        else printf ("L'ordre croissant des trois réels est %.2f < %.2f< %.2f\n",y,x,z);
}
else
  {
    if (y<z)
    printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",x,y,z);
    else
        if (x<z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",x,z,y);
        else printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",z,x,y);
  }
}
```

Exercice 7 :

```
#include<stdio.h>
void main ()
{ float moy,a,b,c;
  printf ("saisie trois notes comprises entre 0 et 20 :\n");
  scanf ("%f%f%f",&a,&b,&c);
  moy= (a+b+c)/3;
  if ( moy<=0 || moy>=20)
```

```
  printf ("votre moyenne est incompréhensible\n");
  else{if (moy>=16 && moy<=20)
    printf ("bravo, vous avez obtenu la mention très bien avec une
moyenne=%.2f\n",moy);
    else
        if (moy>=14 && moy<16)
        printf ("vous avez obtenu la mention bien avec une moyenne=%.2f\n",moy);
        else
            if (moy>=12 && moy<14)
            printf ("vous avez obtenu la mention assez bien avec une
            moyenne=%.2f\n",moy);
            else
              if (moy>=10 && moy<12)
              printf ("vous avez obtenu la mention passable avec une
              moyenne=%.2f\n",moy);
              else
                if (moy>=0 && moy<10)
                printf ("mention insuffisante avec une moyenne=%.2f, \n",moy);
    }
}
```

Exercice 8 :

```
#include<stdio.h>
void main ()
{       int m, n;
  printf ("\Entrez deux entiers m et n :\n");
  scanf ("%d %d", &m, &n);
  if (m>0 && n>0 || m<0 && n<0) /* les parenthèse délimitant les conditions est
  facultative du fait que la priorité le justifie*/
  printf ("\n Le produit de m par n est positif\n");
  else if (m>0 && n<0 || m<0 && n>0)
  printf ("\n Le produit de m par n est négatif\n");
  else printf ("\n Le produit de m par n ne pourrais être que nul\n");
}
```

Exercice 9:

```
#include <stdio.h>
#include <math.h>
void main ()
{       int N, P;
    printf ("Introduisez deux nombres entiers N et P :\n");
    scanf ("%d %d", &N, &P);
    if ( (N>0 && P>0) || (N<=0 && P>0 && fabs (N)<fabs (P)) || (N>0 && P<=0
    && fabs (N)>fabs (P)))
    printf ("Le signe de la somme de %d et %d est positif\n",N,P);
          else if ( (N<0 && P<0) || (N<0 && P>=0 && fabs (N)>fabs (P)) || (N>=0
               && P<0 && fabs (N)<fabs (P)))
               printf ("Le signe de la somme de %d et %d est négatif\n",N,P);
               else printf ("La somme de N et P ne pourrais être que nulle \n");
}
```

Exercice 10 :

```
#include<stdio.h>
void main ()
{       int n;
    printf ("Saisez le nombre entier n:\n");
    scanf ("%d",&n);
    if (n>100 && n<10)
    printf ("Hors limite\n");
    else
    {   if (n%2==0)
        printf ("%d est divisible par 2\n",n);
        if (n%3==0)
        printf ("%d est divisible par 3\n",n);
        if (n%5==0)
        printf ("%d est divisible par 5\n",n);
        if (n%7==0)
        printf ("%d est divisible par 7\n",n);
    }
```

```
    if (n%2!=0 && n%3!=0 && n%5!=0 && n%7!=0)
    printf ("Le nombre %d n'est pas divisible par aucune de ces quatre valeurs
    %d,%d,%d et %d\n",n,2,3,5,7);
}
```

Exercice 11 :

```
#include<stdio.h>
void main ()
{       int U,T,Hu,Th,c,x,y,g,h;
        printf ("Bonjour Mr ou Melle saisie svp votre année de naissance:\n\n");
        scanf ("%d",&x);
        h=x;
        if (x<=1900 || x>=1990)
        printf (" votre année doit être comprise entre 1900 et 1990\n\n");
        else
        {
        if (x>=1900 && x<=1990)
        printf ("\n l'année est raisonnable alors:\n");
        U=x%10;
        x=x/10;
        T=x%10;
        x=x/10;
        Hu=x%10;
        x=x/10;
        Th=x%10;
printf ("les quatre chiffres qui la composent est:\n Units=%d\n Tens=%d\n
Hundreds=%d\n Thousands=%d\n",U,T,Hu,Th);
printf ("****** Maintenant je vous propose un menu ******\n\n");
printf ("Voulez-vous afficher la somme des quatre chiffres? tapez le nombre 1\n");
printf ("Voulez-vous afficher le produit des quatre chiffres? tapez le nombre 2\n");
printf ("Voulez-vous afficher votre age? tapez le nombre 3\n");
printf ("entrer votre choix\n");
        scanf ("%d",&c);
```

```
        if (c= =1)
                printf ("la somme des quatre chiffres est %d\n",U+T+Hu+Th);
                else
                        if (c= =2)
                        printf ("le produit des quatre chiffres est %d\n",U*T*Hu*Th);
                        else
                                if (c= =3)
                                {
                                printf ("saisie l'année actuel\n");
                                 scanf ("%d",&y);
                                 g=y-h;
                                 printf ("votre âge est %d\n",g);
                                }
}
}
```

Exercice 12 :

a) Le PGCD de deux entiers en utilisant l'instruction itérative do…while

```
#include<stdio.h>
void main ()
{       int a,b,c,R,z,y;
        printf ("saisie deux entiers strictement positifs a et b:\n");
        scanf ("%d%d",&a,&b);
      if (a<b)
      { c=a;
       a=b;
       b=c;
      }
      z=a;
      y=b;
      do
      { R=a%b;
```

```
        a=b;
        b=R;
}
    while (R!=0);
    printf ("Le plus grand commun diviseur de %d et %d est %d\n",z,y,a);
}
```

b) Le PGCD de deux entiers en utilisant l'instruction itérative while

```
#include<stdio.h>
void main ()
{       int a,b,c,R,z,y;
    printf ("saisie deux entiers strictement positifs a et b:\n");
    scanf ("%d%d",&a,&b);
    if (a<b)
      {
      c=a ;
      a=b ;
       b=c ;
      }
      z=a;
      y=b;
      while (R!=0)
      { R=a%b;
      a=b;
      b=R;
      }
      printf ("Le plus grand diviseur commun de %d et %d est %d\n",z,y,a);
}
```

c) Le PGCD de deux entiers en utilisant l'instruction itérative for

```
#include<stdio.h>
void main ()
{       int a,b,c,R,z,y;
    printf ("saisie deux entiers strictement positifs a et b:\n");
```

```
    scanf ("%d%d",&a,&b);
      if (a<b)
      { c=a ;
      a=b ;
       b=c ;
      }
      z=a;
       y=b;
      for (R=1;R!=0;b=R)
      {
      R=a%b;
      a=b;
      }
      printf ("Le plus grand commun diviseur de %d et %d est %d\n",z,y,a);
      }
```

Exercice 13:

```
#include<stdio.h>
void main ()
{       int i, x;
    printf ("Les dix premiers nombres impaires sont:\n");
    for (i=0;i<10;i++)
    { x=2*i+1;
    printf ("%d\n",x);
    }
}
```

Exercice 14:

```
#include<stdio.h>
void main ()
{       int i,x;
    printf ("Les vingt premiers nombres paires sont:\n");
    for (i=1;i<=20;i++) /* 0 est un entier ni pair ni impair*/
    {   x=2*i;
```

```
        printf ("%d\n",x);
    }
}
```

Exercice 15:

```
#include<stdio.h>
void main ()
{       int n, p, i;
    printf ("Tapez deux nombres entiers n et p avec n>p:\n");
    scanf ("%d%d",&n,&p);
    printf ("les nombres de %d à %d, sauf les multiples de 3 et les multiples de 5
    sont:\n",n,p);
    for (i=n; i>=n && i<=p; i++)
        {       if (i%3!=0 && i%5!=0)
                printf ("%d\n",i);
        }
}
```

Exercice 16:

a) Calcul de la somme S=1+3+5…+N en utilisant l'instruction itérative for

```
#include <stdio.h>
main ()
{       long int S; int N, i;
    printf ("Taper la valeur de l'entier impair N \n");
    scanf ("%d", &N);
    for (S=0, i=0; 2*i+1<=N; i++)
    S+=2*i+1;
    printf ("La somme S= 1+3+...+%d des premiers nombres entiers impaires est
%ld\n",N,S);
}
```

b) Calcul de la somme S=2+4+6…+N en utilisant l'instruction itérative for

```
#include <stdio.h>
main ()
{       long int S; int N, i;
    printf ("Taper la valeur de l'entier pair N \n");
```

```
    scanf ("%d", &N);
    for (S=0,i=1;2*i<=N; i++)
    S+=2*i;
    printf ("La somme S= 2+4+...+%d des premiers nombres entiers paires est
%ld\n",N,S);
}
```

c) Calcul de la somme S=1+2²+3²…+N² en utilisant l'instruction itérative for

```
#include <stdio.h>
main ()
{       long int S; int N, i;
    printf ("Taper la valeur de l'entier N \n");
    scanf ("%d", &N);
    for (S=0, i=1; i<=N; i++)
    S+=i*i;
    printf ("La somme S= 1+2*2+3*2+...%d*2 = %ld\n",N,S);
}
```

d) Calcul de la somme S=1+1/2+1/3…+1/N en utilisant l'instruction itérative for

```
#include <stdio.h>
main ()
{       float S, i; int N;
    printf ("Taper la valeur de l'entier N \n");
    scanf ("%d", &N);
    for (S=0,i=1;i<=N; i++)
    S=S+1/i;
    printf ("La somme S= 1+1/2+...+1/%d est %.2f \n",N,S);
}
```

Exercice 17:

```
#include <stdio.h>
main ()
{       char op; int n, p;
    printf ("Choisir l'opération souhaité (+ ou*) ? \n");
    scanf ("%c", &op);
    printf ("Saisir deux nombres entiers \n");
```

```
    scanf ("%d %d", &n, &p);
    if (op= = '+')
    printf ("La somme de %d et %d est %d\n",n,p,n+p) ;
    else printf ("Le produit de %d par %d est %d\n",n,p,n*p) ;
}
```

Exercice 18:

a) En utilisant l'instruction itérative while

```
#include <stdio.h>
void main ()
{       int i =1, j, N;
        long int S=0;
        double Prod = 1;
    printf ("Saisir le nombre de données N:\n");
    scanf ("%d", &N);
    while (i<=N)
    {
        printf ("Enter le %d entier est:\n", i);
        scanf ("%d", &j);
        S+=j;
        Prod*=j;
        i++;
    }
    printf ("La somme de ces %d entiers vaut:%ld\n", N, S);
    printf ("Leur moyenne est %f\n", (float) S/N);
    printf ("Et leur produit est %lf \n",Prod);
}
```

b) En utilisant l'instruction itérative do…while

```
#include <stdio.h>
void main ()
{       int i =1, j, N;
        long int S=0;
        double Prod = 1;
```

```
    printf ("Saisir le nombre de données N:\n");
    scanf ("%d", &N);
    do
    {   printf ("Enter le %d entier est:\n", i);
        scanf ("%d", &j);
        S+=j;
        Prod*=j;
        i++;
    }
    while (i<=N);
    printf ("La somme de ces %d entiers vaut:%ld\n", N, S);
    printf ("Leur moyenne est %f\n", (float) S/N);
    printf ("Et leur produit est %lf \n",Prod);
}
```

c) En utilisant l'instruction itérative for

```
#include <stdio.h>
void main ()
{       int i, j, N;
        long int S;
        double Prod ;
    printf ("Saisir le nombre de données N:\n");
    scanf ("%d", &N);
    for (S=0, Prod = 1, i =1; i<=N; i++)
    {   printf ("Enter le %d entier est:\n", i);
        scanf ("%d", &j);
        S+=j;
        Prod*=j;
    }
    printf ("La somme de ces %d entiers vaut:%ld\n", N, S);
    printf ("Leur moyenne est %f\n", (float) S/N);
    printf ("Et leur produit est %lf \n",Prod);
}
```

Exercice 19:

```
#include <stdio.h>
main ()
{ int N,P=2;
printf ("saisie le nombre entier N:\n");
scanf ("%d",&N);
while (P<N && N%P!=0)
P=P+1;
if (N= =P)
printf ("%d est un nombre premier:\n",N);
else
printf ("%d est un nombre non premier:\n",N);
}
```

Exercice 20:

```
#include<stdio.h>
void main ()
{       float x,y,z;
printf ("Taper trois réels x # y # z :\n");
scanf ("%f%f%f",&x,&y,&z);
        if (x>y)
        {       if (y>z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",z,y,x);
        else
        if (x>z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",y,z,x);
        else
        printf ("L'ordre croissant des trois réels est %.2f < %.2f< %.2f\n",y,x,z);
        }
        else
        { if (y<z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",x,y,z);
        else
```

```
        if (x<z)
        printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",x,z,y);
        else printf ("L'ordre croissant des trois réels est %.2f < %.2f < %.2f\n",z,x,y);
        }
}
```

Exercice 21:

```
#include<stdio.h>
void main ()
{       int i=0, n,som=0;
while(i<3)
        {       printf ("donner un entier:\n");
        scanf ("%d",&n);
        som+=n;
        i++;
    }
    printf ("la somme des trois entiers que vous avez saisis est %d\n",som);
```

i. Le résultat fourni par l'exécution de ce programme est :

```
        donner un entier :2
        donner un entier :4
        donner un entier :6
        la somme des trois entiers que vous avez saisis est 12
```

ii. **En utilisant la boucle do°…°while**

```
#include<stdio.h>
void main ()
{       int i=0, n,som=0 ;
do {    printf ("donner un entier:\n");
        scanf ("%d",&n);
        som+=n;
        i++;
}
while(i<3) ;
printf ("la somme des trois entiers que vous avez saisis est %d\n",som);
```

iii. En utilisant la boucle for

```
#include<stdio.h>
void main ()
{       int i, n,som=0 ;
for(i=0 ; i<3 ; i++)
                {       printf ("donner un entier:\n");
                scanf ("%d",&n);
                som+=n;
         }
printf ("la somme des trois entiers que vous avez saisis est %d\n",som);
```

Exercice 22 :

```
/*ce programme calcul la solution de la fameuse équation ax2+bx+c=0*/
#include<stdio.h>
#include<math.h>
void main()
{       double a,b,c,delta,x,y;
        printf("taper trois réels:\n");
        scanf("%lf%lf%lf",&a,&b,&c);
        if(a==0)
        {       if(b==0)
                {       if(c==0)
                                printf("toute valeur est admise\n");
                        else    printf("aucune valeur n'est admise\n");
                }
                else    printf("%3.2lf est la solution double de l'equation\n",-c/b);
        }
        else
        {       delta=b*b-4*a*c;
                x=(-b+sqrt(delta))/(2*a);
                y=(-b-sqrt(delta))/(2*a);
                if(delta>=0)
                {       if(delta==0)
```

```
                printf(" %3.2lf est la solution double de l'equation\n",-b/(2*a));
                else
                printf("%3.2lf et %3.2lf deux solutions de l'equation \n",x,y);
            }
            else
            printf("pas solution dans R\n");
        }
}
```

Exercice 23 :

```
#include<stdio.h>
void main()
{
/* déclaration des variables*/
int i ;
printf ("tapez un nombre entier : ");
scanf ("%d",&i);
if(i==0)
printf ("%d est un entier nul\n",i);
else if (i>0)
printf (" %d est un entier positif\n",i);
else printf ("%d est un entier négatif\n",i);
}
```

Exercice 24 :

```
#include<stdio.h>
void main ()
{       int A,B,C,D,X,Y,Z;
printf ("Taper A,B,C et D:\n");
scanf ("%d%d %d%d",&A,&B,&C,&D);
        if ( !(A<B))    X=X+Y ;
        if (C<D)        X=X+3 ;
        else            Y=X+Y ;
        Z=X+Y*Y;
printf ("Z=%d\n",Z);
}
```

Exercice 25 :

```
#include<stdio.h>
void main()
{       int Tab[3][2]={2,4,20,40,200,400} ;
        int i,j ;
for(i=0; i<3; i++)
        {       for(j=0; j<2; j++)
                printf ("%5d", Tab[i][j]);
                printf ("\n");
        }
}
```

Exercices: Partie II

Exercice 1 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Calculer et afficher ensuite la somme des éléments du tableau.

Exercice 2 :

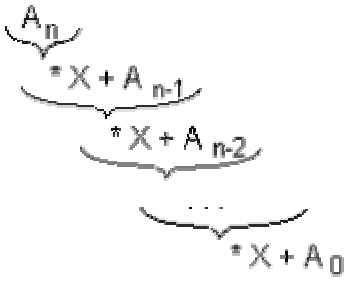
Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

Exercice 3 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.
Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

Exercice 4 :

Calculer pour une valeur X donnée du type float la valeur numérique d'un polynôme de degré n:
P(X) = AnXn + An-1Xn-1 + ... + A1X + A0
Les valeurs des coefficients An, ... , A0 seront entrées au clavier et mémorisées dans un tableau A de type float et de dimension n+1.
a) Utilisez la fonction pow() pour le calcul.
b) Utilisez le schéma de Horner qui évite les opérations d'exponentiation:



Exercice 5 :

Ecrire un programme qui détermine la plus grande et la plus petite valeur dans un tableau d'entiers A. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maxima ou minima, le programme retiendra la position du premier maximum ou minimum rencontré.

Exercice 6 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Calculer et afficher ensuite la somme des éléments du tableau.

Exercice 7 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Effacer ensuite toutes les occurrences de la valeur **0** dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

Exercice 8 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.
Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

Exercice 9 :

Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Copiez ensuite toutes les composantes strictement positives dans un deuxième tableau TPOS et toutes les valeurs strictement négatives dans un troisième tableau TNEG. Afficher les tableaux TPOS et TNEG

Exercice 10 :

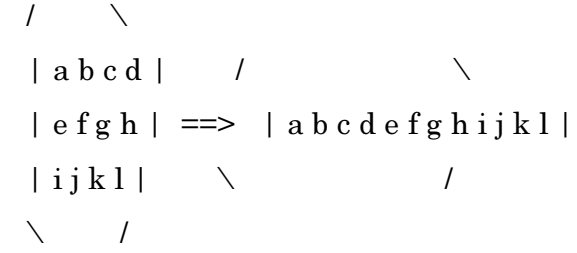
Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int (dimensions maximales: 50 lignes et 50 colonnes). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de tous ses éléments.

Exercice 11 :

Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int (dimensions maximales: 50 lignes et 50 colonnes). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de chaque ligne et de chaque colonne en n'utilisant qu'une variable d'aide pour la somme.

Exercice 12 :

Ecrire un programme qui transfère un tableau M à deux dimensions L et C (dimensions maximales: 10 lignes et 10 colonnes) dans un tableau V à une dimension L*C.
Exemple:



Exercice 13 :

Ecrire un programme qui calcule le produit scalaire de deux vecteurs d'entiers U et V (de même dimension). Exemple:

```
/       \ /       \
| 3  2  -4 | * | 2  -3  5 | = 3*2+2*(-3)+(-4)*5 = -20
\       / \       /
```

Exercice 14 :

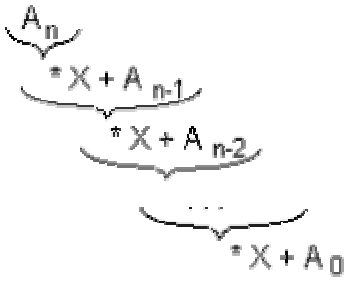
Calculer pour une valeur X donnée du type float la valeur numérique d'un polynôme de degré n:

P(X) = AnXn + An-1Xn-1 + ... + A1X + A0

Les valeurs des coefficients An, ... , A0 seront entrées au clavier et mémorisées dans un tableau A de type float et de dimension n+1.

a) Utilisez la fonction pow() pour le calcul.

b) Utilisez le schéma de Horner qui évite les opérations d'exponentiation:



Exercice 15 :

Ecrire un programme qui détermine la plus grande et la plus petite valeur dans un tableau d'entiers A. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maxima ou minima, le programme retiendra la position du premier maximum ou minimum rencontré.

Exercice 16 :

Un tableau A de dimension N+1 contient N valeurs entières triées par ordre croissant; la (N+1)ième valeur est indéfinie. Insérer une valeur VAL donnée au clavier dans le tableau A de manière à obtenir un tableau de N+1 valeurs triées.

Exercice 17 :

Problème: Rechercher dans un tableau d'entiers A une valeur VAL entrée au clavier. Afficher la position de VAL si elle se trouve dans le tableau, sinon afficher un message correspondant. La valeur POS qui est utilisée pour mémoriser la position de la valeur dans le tableau, aura la valeur -1 aussi longtemps que VAL n'a pas été trouvée.

Implémenter deux versions:

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

b) La recherche dichotomique ('recherche binaire', 'binary search')

Condition: Le tableau A doit être trié

Comparer le nombre recherché à la valeur au milieu du tableau,

- s'il y a égalité ou si le tableau est épuisé, arrêter le traitement avec un message correspondant.
- si la valeur recherchée précède la valeur actuelle du tableau, continuer la recherche dans le demi-tableau à gauche de la position actuelle.
- si la valeur recherchée suit la valeur actuelle du tableau, continuer la recherche dans le demi-tableau à droite de la position actuelle.

Ecrire le programme pour le cas où le tableau A est trié par ordre croissant.

Question: Quel est l'avantage de la recherche dichotomique? Expliquer brièvement.

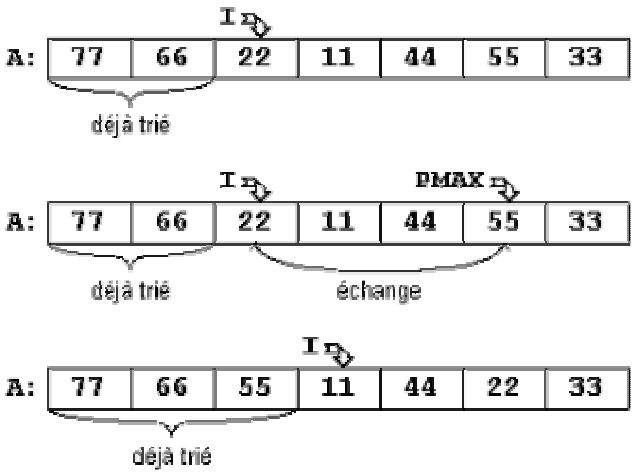
Exercice 18 :

Problème: On dispose de deux tableaux A et B (de dimensions respectives N et M), triés par ordre croissant. Fusionner les éléments de A et B dans un troisième tableau FUS trié par ordre croissant.

Méthode: Utiliser trois indices IA, IB et IFUS. Comparer A[IA] et B[IB]; remplacer FUS[IFUS] par le plus petit des deux éléments; avancer dans le tableau FUS et dans le tableau qui a contribué son élément. Lorsque l'un des deux tableaux A ou B est épuisé, il suffit de recopier les éléments restants de l'autre tableau dans le tableau FUS.

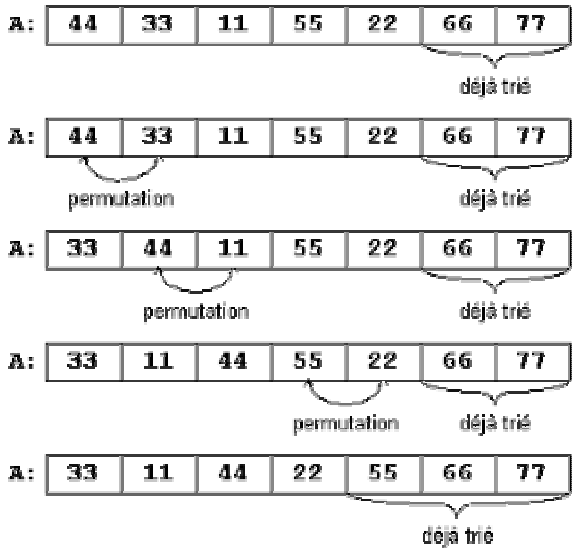
Exercice 19 :

Problème: Classer les éléments d'un tableau A par ordre décroissant.
Méthode: Parcourir le tableau de gauche à droite à l'aide de l'indice I. Pour chaque élément A[I] du tableau, déterminer la position PMAX du (premier) maximum à droite de A[I] et échanger A[I] et A[PMAX].
Exemple:



Exercice 20 :

Problème: Classer les éléments d'un tableau A par ordre croissant.
Méthode: En recommençant chaque fois au début du tableau, on effectue à plusieurs reprises le traitement suivant: On propage, par permutations successives, le plus grand élément du tableau vers la fin du tableau (comme une bulle qui remonte à la surface d'un liquide).
Exemple:



Implémenter l'algorithme en considérant que:

* La partie du tableau (à droite) où il n'y a pas eu de permutations est triée.

* Si aucune permutation n'a eu lieu, le tableau est trié.

Exercice 21 :

Ecrire un programme qui lit les points de N élèves d'une classe dans un devoir et les mémorise dans un tableau POINTS de dimension N.

* Rechercher et afficher:

- la note maximale,

- la note minimale,

- la moyenne des notes.

* A partir des POINTS des élèves, établir un tableau NOTES de dimension 7 qui est composé de la façon suivante:

NOTES[6] contient le nombre de notes 60

NOTES[5] contient le nombre de notes de 50 à 59

NOTES[4] contient le nombre de notes de 40 à 49

NOTES[0] contient le nombre de notes de 0 à 9

Etablir un graphique de barreaux représentant le tableau NOTES. Utilisez les symboles ####### pour la représentation des barreaux et affichez le domaine des notes en dessous du graphique.

Idée: Déterminer la valeur maximale MAXN dans le tableau NOTES et afficher autant de lignes sur l'écran. (Dans l'exemple ci-dessous, MAXN = 6).

Exemple:

```
La note maximale est 58
La note minimale est 13
La moyenne des notes est 37.250000
  6 >                      #######
  5 >                ####### #######
  4 >          ####### ####### #######
  3 >          ####### ####### ####### #######
  2 >    ####### ####### ####### ####### #######
  1 >    ####### ####### ####### ####### #######
    +-------+-------+-------+-------+-------+-------+-------+
    I 0 - 9 I 10-19 I 20-29 I 30-39 I 40-49 I 50-59 I  60   I
```

Exercice 22:

Ecrire un programme qui met à zéro les éléments de la diagonale principale d'une matrice carrée A donnée.

Exercice 23 :

Ecrire un programme qui construit et affiche une matrice carrée unitaire U de dimension N. Une matrice unitaire est une matrice, telle que:

$$uij = \begin{cases} 1 & \text{si } i=j \\ 0 & \text{si } ij \end{cases}$$

Exercice 24 :

Ecrire un programme qui effectue la transposition tA d'une matrice A de dimensions N et M en une matrice de dimensions M et N.

a) La matrice transposée sera mémorisée dans une deuxième matrice B qui sera ensuite affichée.

b) La matrice A sera transposée par permutation des éléments.

Rappel:

$$tA = t \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} = \begin{pmatrix} a & e & i \\ b & f & j \\ c & g & k \\ d & h & l \end{pmatrix}$$

Exercice 25 :

Ecrire un programme qui réalise la multiplication d'une matrice A par un réel X.

Rappel:

```
    /       \    /             \
     | a b c d |     | X*a X*b X*c X*d |
  X * | e f g h |  =  | X*e X*f X*g X*h |
     | i j k l |     | X*i X*j X*k X*l |
     \       /    \             /
```

a) Le résultat de la multiplication sera mémorisé dans une deuxième matrice A qui sera ensuite affichée.

b) Les éléments de la matrice A seront multipliés par X.

Exercice 26 :

Ecrire un programme qui réalise l'addition de deux matrices A et B de mêmes dimensions N et M.

Rappel:

```
/       \ /         \    /                \
| a b c d |   | a' b' c' d' |     | a+a' b+b' c+c' d+d' |
| e f g h | + | e' f' g' h' |  =  | e+e' f+f' g+g' h+h' |
| i j k l |   | i' j' k' l' |     | i+i' j+j' k+k' l+l' |
\       / \         /    \                /
```

a) Le résultat de l'addition sera mémorisé dans une troisième matrice C qui sera ensuite affichée.

b) La matrice B est ajoutée à A.

Exercice 27 :

En multipliant une matrice A de dimensions N et M avec une matrice B de dimensions M et P on obtient une matrice C de dimensions N et P:

A(N,M) * B(M,P) = C(N,P)

La multiplication de deux matrices se fait en multipliant les composantes des deux matrices lignes par colonnes:

$$c_{ij} = \sum_{k=1}^{k=M} (a_{ik} * b_{kj})$$

Rappel:

```
/     \    /    \     /                              \
| a b c |     | p q |      | a*p + b*r + c*t     a*q + b*s + c*u |
| e f g |  *  | r s |  =   | e*p + f*r + g*t     e*q + f*s + g*u |
| h i j |     | t u |      | h*p + i*r + j*t     h*q + i*s + j*u |
| k l m |     \    /      | k*p + l*r + m*t     k*q + l*s + m*u |
\     /              \                              /
```

Ecrire un programme qui effectue la multiplication de deux matrices A et B. Le résultat de la multiplication sera mémorisé dans une troisième matrice C qui sera ensuite affichée.

Exercice 28 :

Ecrire un programme qui construit le triangle de PASCAL de degré N et le mémorise dans une matrice carrée P de dimension N+1.

Exemple: Triangle de Pascal de degré 6:

n=0 1

n=1 1 1

n=2 1 2 1

n=3 1 3 3 1

n=4 1 4 6 4 1

n=5 1 5 10 10 5 1

n=6 1 6 15 20 15 6 1

Méthode:

Calculer et afficher seulement les valeurs jusqu'à la diagonale principale (incluse).

Limiter le degré à entrer par l'utilisateur à 13.

Construire le triangle ligne par ligne:

- Initialiser le premier élément et l'élément de la diagonale à 1.
- Calculer les valeurs entre les éléments initialisés de gauche à droite en utilisant la relation:

$P_{i,j} = P_{i-1,j} + P_{i-1,j-1}$

Exercice 29 :

Rechercher dans une matrice donnée A les éléments qui sont à la fois un maximum sur leur ligne et un minimum sur leur colonne. Ces éléments sont appelés des points-cols. Afficher les positions et les valeurs de tous les points-cols trouvés.

Exemples: Les éléments soulignés sont des points-cols:

```
/       \ /       \ /       \ /     \
| 1 8 3 4 0 |   | 4 5 8 9 |   | 3 5 6 7 7 |   | 1 2 3 |
|         |   | 3 8 9 3 |   | 4 2 2 8 9 |   | 4 5 6 |
| 6 7 2 7 0 |   | 3 4 9 3 |   | 6 3 2 9 7 |   | 7 8 9 |
\       / \       / \       / \     /
```

Méthode: Etablir deux matrices d'aide MAX et MIN de même dimensions que A, telles que:

$$MAX[i,j] = \begin{cases} 1 \text{ si } A[i,j] \text{ est un maximum} \\ \text{sur la ligne } i \\ 0 \text{ sinon} \end{cases}$$

$$MIN[i,j] = \begin{cases} 1 \text{ si } A[i,j] \text{ est un minimum} \\ \text{sur la colonne } j \\ 0 \text{ sinon} \end{cases}$$

Solutions :

Exercice 1 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 long SOM;  /* somme des éléments - type long à cause */
          /* de la grandeur prévisible du résultat. */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
  /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
  /* Calcul de la somme */
 for (SOM=0, I=0; I<N; I++)
    SOM += T[I];
  /* Edition du résultat */
 printf("Somme des éléments : %ld\n", SOM);
 return 0;
}
```

Exercice 2:

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné    */
 int N;     /* dimension        */
 int I,J;   /* indices courants */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
```

```
    printf("%d ", T[I]);
 printf("\n");
 /* Effacer les zéros et comprimer :        */
 /* Copier tous les éléments de I vers J et */
 /* augmenter J pour les éléments non nuls. */
 for (I=0, J=0 ; I<N ; I++)
    {
     T[J] = T[I];
     if (T[I]) J++;
    }
 /* Nouvelle dimension du tableau ! */
 N = J;
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 return 0;
}
```

Exercice 3 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné */
 int N;     /* dimension     */
 int I,J;   /* indices courants */
 int AIDE;  /* pour l'échange   */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 /* Inverser le tableau */
 for (I=0, J=N-1 ; I<J ; I++,J--)
     /* Echange de T[I] et T[J] */
       {
        AIDE = T[I];
        T[I] = T[J];
        T[J] = AIDE;
       }
```

```
 /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 return 0;
}
```

Exercice 4 :

```
#include <stdio.h>
#include <math.h>
main()
{ float A[20];/* tableau des coefficients de P */
 int I;     /* indice courant    */
 int N;     /* degré du polynôme */
 float X;   /* argument */
 float P;   /* résultat */
 /* Saisie du degré N et de l'argument X */
 printf("Entrer le degré  N du polynôme (max.20) : ");
 scanf("%d", &N);
 printf("Entrer la valeur X de l'argument : ");
 scanf("%f", &X);
 /* Saisie des coefficients */
 for (I=0 ; I<=N ; I++)
   {    printf("Entrer le coefficient A%d : ", I);
        scanf("%f", &A[I]);
   }
 /* a) Calcul à l'aide de pow
 for (P=0.0, I=0 ; I<=N ; I++)
      P +=  A[I]*pow(X,I);      */
  /* b) Calcul de Horner */
 /* commencer le calcul avec le dernier coefficient...*/
 for (P=0.0, I=0 ; I<=N ; I++)
      P = P*X + A[N-I];
  /* Edition du résultat */
 printf("Valeur du polynôme pour X = %.2f : %.2f\n", X, P);
 return 0;
}
```

Exercice 5 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 int MIN;   /* position du minimum */
 int MAX;   /* position du maximum */
 /* Saisie des données */
```

```
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &A[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Recherche du maximum et du minimum */
 MIN=0;
 MAX=0;
 for (I=0; I<N; I++)
    {     if(A[I]>A[MAX]) MAX=I;
          if(A[I]<A[MIN]) MIN=I;
    }
  /* Edition du résultat */
 printf("Position du minimum : %d\n", MIN);
 printf("Position du maximum : %d\n", MAX);
 printf("Valeur   du minimum : %d\n", A[MIN]);
 printf("Valeur   du maximum : %d\n", A[MAX]);
 return 0;
}
```

Exercice 6 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 long SOM;  /* somme des éléments - type long à cause */
          /* de la grandeur prévisible du résultat. */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
  /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
```

```
 /* Calcul de la somme */
 for (SOM=0, I=0; I<N; I++)
    SOM += T[I];
 /* Edition du résultat */
 printf("Somme de éléments : %ld\n", SOM);
 return 0;
}
```

Exercice 7 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné    */
 int N;     /* dimension        */
 int I,J;   /* indices courants */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 /* Effacer les zéros et comprimer :        */
 /* Copier tous les éléments de I vers J et */
 /* augmenter J pour les éléments non nuls. */
 for (I=0, J=0 ; I<N ; I++)
    {
     T[J] = T[I];
     if (T[I]) J++;
    }
 /* Nouvelle dimension du tableau ! */
 N = J;
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 return 0;
}
```

Exercice 8 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50]; /* tableau donné */
 int N;     /* dimension     */
 int I,J;   /* indices courants */
 int AIDE;  /* pour l'échange   */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 /* Inverser le tableau */
 for (I=0, J=N-1 ; I<J ; I++,J--)
     /* Echange de T[I] et T[J] */
       {
        AIDE = T[I];
        T[I] = T[J];
        T[J] = AIDE;
       }
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 return 0;
}
```

Exercice 9 :

```
#include <stdio.h>
main()
{
 /* Déclarations */
 /* Les tableaux et leurs dimensions */
 int T[50], TPOS[50], TNEG[50];
 int N,     NPOS,     NNEG;
 int I; /* indice courant */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
```

```
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &T[I]);
   }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
    printf("%d ", T[I]);
 printf("\n");
 /* Initialisation des dimensions de TPOS et TNEG */
 NPOS=0;
 NNEG=0;
 /* Transfer des données */
 for (I=0; I<N; I++)
     { if (T[I]>0) {
                 TPOS[NPOS]=T[I];
                 NPOS++;
                }
       if (T[I]<0) {
                 TNEG[NNEG]=T[I];
                 NNEG++;
                }
     }
  /* Edition du résultat */
 printf("Tableau TPOS :\n");
 for (I=0; I<NPOS; I++)
    printf("%d ", TPOS[I]);
 printf("\n");
 printf("Tableau TNEG :\n");
 for (I=0; I<NNEG; I++)
    printf("%d ", TNEG[I]);
 printf("\n");
 return 0;
}
```

Exercice 10 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50][50]; /* tableau donné */
 int L, C;  /* dimensions       */
 int I, J;  /* indices courants  */
 long SOM;  /* somme des éléments - type long à cause */
          /* de la grandeur prévisible du résultat. */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &L );
```

```
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &C );
 for (I=0; I<L; I++)
   for (J=0; J<C; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &T[I][J]);
      }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
   {
    for (J=0; J<C; J++)
       printf("%7d", T[I][J]);
    printf("\n");
   }
 /* Calcul de la somme  */
 for (SOM=0, I=0; I<L; I++)
    for (J=0; J<C; J++)
       SOM += T[I][J];
 /* Edition du résultat */
 printf("Somme des éléments : %ld\n", SOM);
 return 0;
}
```

Exercice 11 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int T[50][50]; /* tableau donné    */
 int L, C;      /* dimensions       */
 int I, J;      /* indices courants */
 long SOM;  /* somme des éléments - type long à cause   */
          /* de la grandeur prévisible des résultats. */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &L );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &C );
 for (I=0; I<L; I++)
   for (J=0; J<C; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &T[I][J]);
      }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
   {
```

```
    for (J=0; J<C; J++)
        printf("%7d", T[I][J]);
    printf("\n");
   }
 /* Calcul et affichage de la somme des lignes */
 for (I=0; I<L; I++)
    {
     for (SOM=0, J=0; J<C; J++)
          SOM += T[I][J];
     printf("Somme - ligne %d : %ld\n",I,SOM);
    }
 /* Calcul et affichage de la somme des colonnes */
 for (J=0; J<C; J++)
    {
     for (SOM=0, I=0; I<L; I++)
          SOM += T[I][J];
     printf("Somme - colonne %d : %ld\n",J,SOM);
    }
  return 0;
}
```

Exercice 12:

```
#include <stdio.h>
main()
{ /* Déclarations */
 int M[10][10]; /* tableau à 2 dimensions */
 int V[100];    /* tableau à 1 dimension  */
 int L, C;      /* dimensions       */
 int I, J;      /* indices courants */
 /* Saisie des données */
 printf("Nombre de lignes   (max.10) : ");
 scanf("%d", &L );
 printf("Nombre de colonnes (max.10) : ");
 scanf("%d", &C );
 for (I=0; I<L; I++)
   for (J=0; J<C; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &M[I][J]);
      }
 /* Affichage du tableau 2-dim */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
   {
    for (J=0; J<C; J++)
        printf("%7d", M[I][J]);
    printf("\n");
   }
```

```
 /* Transfer des éléments ligne par ligne */
 for (I=0; I<L; I++)
      for (J=0; J<C; J++)
           V[I*C+J] = M[I][J];
  /* Affichage du tableau 1-dim */
 printf("Tableau résultat : ");
 for (I=0; I<L*C; I++)
     printf("%d ", V[I]);
 printf("\n");
 return 0;
}
```

Exercice 13 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int U[50], V[50]; /* tableaux donnés */
 int N;     /* dimension        */
 int I;     /* indice courant   */
 long PS;   /* produit scalaire */
 /* Saisie des données */
 printf("Dimension des tableaux (max.50) : ");
 scanf("%d", &N );
 printf("** Premier tableau **\n");
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &U[I]);
    }
 printf("** Deuxième tableau **\n");
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &V[I]);
    }
 /* Calcul du produit scalaire */
 for (PS=0, I=0; I<N; I++)
      PS += (long)U[I]*V[I];
  /* Edition du résultat */
 printf("Produit scalaire : %ld\n", PS);
 return 0;
}
```

Exercice 14 :

```
#include <stdio.h>
#include <math.h>
main()
{ float A[20];/* tableau des coefficients de P */
 int I;      /* indice courant    */
 int N;      /* degré du polynôme */
 float X;    /* argument */
 float P;    /* résultat */
 /* Saisie du degré N et de l'argument X */
 printf("Entrer le degré  N du polynôme (max.20) : ");
 scanf("%d", &N);
 printf("Entrer la valeur X de l'argument : ");
 scanf("%f", &X);
 /* Saisie des coefficients */
 for (I=0 ; I<=N ; I++)
   {    printf("Entrer le coefficient A%d : ", I);
        scanf("%f", &A[I]);
   }
 /* a) Calcul à l'aide de pow
 for (P=0.0, I=0 ; I<=N ; I++)
      P +=  A[I]*pow(X,I);      */
  /* b) Calcul de Horner */
 /* commencer le calcul avec le dernier coefficient...*/
 for (P=0.0, I=0 ; I<=N ; I++)
      P = P*X + A[N-I];
  /* Edition du résultat */
 printf("Valeur du polynôme pour X = %.2f : %.2f\n", X, P);
 return 0;
}
```

Exercice 15 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 int MIN;   /* position du minimum */
 int MAX;   /* position du maximum */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &A[I]);
   }
```

```
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Recherche du maximum et du minimum */
 MIN=0;
 MAX=0;
 for (I=0; I<N; I++)
    {     if(A[I]>A[MAX]) MAX=I;
          if(A[I]<A[MIN]) MIN=I;
    }
  /* Edition du résultat */
 printf("Position du minimum : %d\n", MIN);
 printf("Position du maximum : %d\n", MAX);
 printf("Valeur   du minimum : %d\n", A[MIN]);
 printf("Valeur   du maximum : %d\n", A[MAX]);
 return 0;
}
```

Exercice 16

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné    */
 int VAL;   /* valeur à insérer */
 int N;     /* dimension        */
 int I;     /* indice courant   */
 /* Saisie des données */
 printf("Dimension N du tableau initial (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {    printf("Elément %d : ", I);
        scanf("%d", &A[I]);
   }
 printf("Elément à insérer : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
   printf("%d ", A[I]);
 printf("\n");
 /* Déplacer les éléments plus grands que */
 /* VAL d'une position vers l'arrière.    */
 for (I=N ; (I>0)&&(A[I-1]>VAL) ; I--)
      A[I]=A[I-1];
 /* VAL est copié à la position du dernier */
 /* élément déplacé. */
```

```
 A[I]=VAL; /* Nouvelle dimension du tableau ! */
 N++;
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 return 0;
}
```

Exercice 17 :

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int VAL;   /* valeur à rechercher   */
 int POS;   /* position de la valeur */
 int N;     /* dimension      */
 int I;     /* indice courant */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
    {
     printf("Elément %d : ", I);
     scanf("%d", &A[I]);
    }
 printf("Elément à rechercher : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
     printf("%d ", A[I]);
 printf("\n");
 /* Recherche de la position de la valeur */
 POS = -1;
 for (I=0 ; (I<N)&&(POS==-1) ; I++)
       if (A[I]==VAL)
          POS=I;
  /* Edition du résultat */
 if (POS==-1)
     printf("La valeur recherchée ne se trouve pas "
          "dans le tableau.\n");
  else
     printf("La valeur %d se trouve à la position %d. \n",
```

```
 VAL, POS);
 return 0;
}
```

b) La recherche dichotomique ('recherche binaire', 'binary search')

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné */
 int VAL;   /* valeur à rechercher   */
 int POS;   /* position de la valeur */
 int N;     /* dimension      */
 int I;     /* indice courant */
 int INF, MIL, SUP; /* limites du champ de recherche */
  /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {     printf("Elément %d : ", I);
         scanf("%d", &A[I]);
   }
 printf("Elément à rechercher : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Initialisation des limites du domaine de recherche */
 INF=0; SUP=N-1;
 /* Recherche de la position de la valeur */
 POS=-1;
 while ((INF<=SUP) && (POS==-1))
       {
        MIL=(SUP+INF)/2;
        if (VAL < A[MIL])
             SUP=MIL-1;
        else if (VAL > A[MIL])     INF=MIL+1;
        else                       POS=MIL;
       }
   /* Edition du résultat */
 if (POS==-1)
     printf("La valeur recherchée ne se trouve pas "
          "dans le tableau.\n");
 else
     printf("La valeur %d se trouve à la position %d. \n",
 VAL, POS);
 return 0;
}
```

Exercice 18 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 /* Les tableaux et leurs dimensions */
 int A[50], B[50], FUS[100];
 int N, M;
 int IA, IB, IFUS; /* indices courants */
 /* Saisie des données */
 printf("Dimension du tableau A (max.50) : ");
 scanf("%d", &N );
 printf("Entrer les éléments de A dans l'ordre croissant :\n");
 for (IA=0; IA<N; IA++)
   {
    printf("Elément A[%d] : ", IA);
    scanf("%d", &A[IA]);
   }
 printf("Dimension du tableau B (max.50) : ");
 scanf("%d", &M );
 printf("Entrer les éléments de B dans l'ordre croissant :\n");
 for (IB=0; IB<M; IB++)
   {
    printf("Elément B[%d] : ", IB);
    scanf("%d", &B[IB]);
   }
 /* Affichage des tableaux A et B */
 printf("Tableau A :\n");
 for (IA=0; IA<N; IA++)
    printf("%d ", A[IA]);
 printf("\n");
 printf("Tableau B :\n");
 for (IB=0; IB<M; IB++)
    printf("%d ", B[IB]);
 printf("\n");
 /* Fusion des éléments de A et B dans FUS */
 /* de façon à ce que FUS soit aussi trié. */
 IA=0; IB=0; IFUS=0;
 while ((IA<N) && (IB<M))
       if(A[IA]<B[IB])
          {
           FUS[IFUS]=A[IA];
           IFUS++;
           IA++;
          }
       else
          {
           FUS[IFUS]=B[IB];
           IFUS++;
```

```
          IB++;
         }
/* Si IA ou IB sont arrivés à la fin de leur tableau, */
/* alors copier le reste de l'autre tableau.          */
while (IA<N)
      {
       FUS[IFUS]=A[IA];
       IFUS++;
       IA++;
      }
while (IB<M)
      {
       FUS[IFUS]=B[IB];
       IFUS++;
       IB++;
      }
 /* Edition du résultat */
printf("Tableau FUS :\n");
for (IFUS=0; IFUS<N+M; IFUS++)
   printf("%d ", FUS[IFUS]);
printf("\n");
return 0;
}
```

Exercice 19:

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné */
 int N;     /* dimension     */
 int I;     /* rang à partir duquel A n'est pas trié */
 int J;     /* indice courant      */
 int AIDE;  /* pour la permutation */
 int PMAX;  /* indique la position de l'élément */
          /* maximal à droite de A[I]         */
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (J=0; J<N; J++)
   {
    printf("Elément %d : ", J);
    scanf("%d", &A[J]);
   }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (J=0; J<N; J++)
   printf("%d ", A[J]);
 printf("\n");
```

```
/* Tri du tableau par sélection directe du maximum. */
 for (I=0; I<N-1; I++)
    {
     /* Recherche du maximum à droite de A[I] */
     PMAX=I;
     for (J=I+1; J<N; J++)
        if (A[J]>A[PMAX]) PMAX=J;
     /* Echange de A[I] avec le maximum */
     AIDE=A[I];
     A[I]=A[PMAX];
     A[PMAX]=AIDE;
    }
  /* Edition du résultat */
 printf("Tableau trié :\n");
 for (J=0; J<N; J++)
    printf("%d ", A[J]);
 printf("\n");
 return 0;
}
```

Exercice 20 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50]; /* tableau donné */
 int N;     /* dimension     */
 int I;     /* rang à partir duquel A est trié */
 int J;     /* indice courant      */
 int AIDE;  /* pour la permutation */
 int FIN;   /* position où la dernière permutation a eu lieu. */
       /* permet de ne pas trier un sous-ensemble déjà trié. */
  /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (J=0; J<N; J++)
   {    printf("Elément %d : ", J);
        scanf("%d", &A[J]);
   }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (J=0; J<N; J++)
   printf("%d ", A[J]);
 printf("\n");
 /* Tri du tableau par propagation de l'élément maximal. */
 for (I=N-1 ; I>0 ; I=FIN)
    {     FIN=0;
     for (J=0; J<I; J++)
          if (A[J]>A[J+1])
```

```
          {FIN=J;
           AIDE=A[J];
           A[J]=A[J+1];
           A[J+1]=AIDE;
          }
    }
  /* Edition du résultat */
 printf("Tableau trié :\n");
 for (J=0; J<N; J++)
    printf("%d ", A[J]);
 printf("\n");
 return 0;}
```

Exercice 21 :

```
#include <stdio.h>
main()
{
 int POINTS[50];  /* tableau des points */
 int NOTES[7];    /* tableau des notes  */
 int N;           /* nombre d'élèves    */
 int I, IN;       /* compteurs d'aide   */
 int SOM;         /* somme des points   */
 int MAX, MIN;    /* maximum, minimum de points    */
 int MAXN;        /* nombre de lignes du graphique */
 /* Saisie des données */
 printf("Entrez le nombre d'élèves (max.50) : ");
 scanf("%d", &N);
 printf("Entrez les points des élèves:\n");
 for (I=0; I<N; I++)
   {printf("Elève %d:", I+1);
    scanf("%d", &POINTS[I]);
   }
 printf("\n");
 /* Calcul et affichage du maximum et du minimum des points */
 for (MAX=0, MIN=60, I=0; I<N; I++)
   {if (POINTS[I] > MAX) MAX=POINTS[I];
    if (POINTS[I] < MIN) MIN=POINTS[I];
   }
 printf("La note maximale est %d \n", MAX);
 printf("La note minimale est %d \n", MIN);
 /* Calcul et affichage de la moyenne des points */
 for (SOM=0,I=0 ; I<N ; I++)
     SOM += POINTS[I];
 printf("La moyenne des notes est %f \n", (float)SOM/N);
  /* Etablissement du tableau NOTES */
 for (IN=0 ; IN<7 ; IN++)
     NOTES[IN] = 0;
 for (I=0; I<N; I++)
```

```
    NOTES[POINTS[I]/10]++;
 /* Recherche du maximum MAXN dans NOTES */
 for (MAXN=0,IN=0 ; IN<7 ; IN++)
    if (NOTES[IN] > MAXN)
       MAXN = NOTES[IN];
 /* Affichage du graphique de barreaux */
 /* Représentation de MAXN lignes */
 for (I=MAXN; I>0; I--)
   {
    printf("\n %2d  >", I);
    for (IN=0; IN<7; IN++)
        {
         if (NOTES[IN]>=I)
            printf(" #######");
         else
            printf("        ");
        }
   }
 /* Affichage du domaine des notes */
 printf("\n      +");
 for (IN=0; IN<7; IN++)
    printf("--------+");
 printf("\n      I 0 - 9 I 10-19 I 20-29 "
       "I 30-39 I 40-49 I 50-59 I  60   I\n");
  return 0;
}
```

Exercice 22

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50]; /* matrice carrée   */
 int N;        /* dimension de la matrice carrée */
 int I, J;     /* indices courants */

 /* Saisie des données */
 printf("Dimension de la matrice carrée (max.50) : ");
 scanf("%d", &N);
 for (I=0; I<N; I++)
   for (J=0; J<N; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
```

```
   {
    for (J=0; J<N; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 /* Mise à zéro de la diagonale principale */
 for (I=0; I<N; I++)
     A[I][I]=0;
 /* Edition du résultat */
 printf("Matrice résultat :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<N; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 return 0;}
```

Exercice 23 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int U[50][50]; /* matrice unitaire */
 int N;        /* dimension de la matrice unitaire */
 int I, J;     /* indices courants */
 /* Saisie des données */
 printf("Dimension de la matrice carrée (max.50) : ");
 scanf("%d", &N);
 /* Construction de la matrice carrée unitaire */
 for (I=0; I<N; I++)
    for (J=0; J<N; J++)
       if (I==J)
          U[I][J]=1;
       else
          U[I][J]=0;
  /* Edition du résultat */
 printf("Matrice unitaire de dimension %d :\n", N);
 for (I=0; I<N; I++)
   {
    for (J=0; J<N; J++)
        printf("%7d", U[I][J]);
    printf("\n");
   }
 return 0;
}
```

Remarque:
L'opération:

```
        if (I==J)
                U[I][J]=1;
        else
                U[I][J]=0;
```

peut être simplifiée par:

```
        U[I][J] = (I==J);
```

Exercice 24 :

a) La matrice transposée sera mémorisée dans une deuxième matrice B qui sera ensuite affichée.

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50][50];  /* matrice initiale */
 int B[50][50];  /* matrice résultat */
 int N, M;       /* dimensions des matrices */
 int I, J;       /* indices courants */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 /* Affectation de la matrice transposée à B */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
        B[J][I]=A[I][J];
  /* Edition du résultat */
 /* Attention: maintenant le rôle de N et M est inversé. */
  printf("Matrice résultat :\n");
 for (I=0; I<M; I++)
```

```
   {
    for (J=0; J<N; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
 return 0;
}
```

b) La matrice A sera transposée par permutation des éléments.

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50][50];  /* matrice donnée          */
 int N, M;       /* dimensions de la matrice */
 int I, J;       /* indices courants         */
 int AIDE;       /* pour la permutation      */
 int DMAX;       /* la plus grande des deux dimensions */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 /* Transposition de la matrice A par permutation des   */
 /* éléments [I][J] à gauche de la diagonale principale */
 /* avec les éléments [J][I] à droite de la diagonale.  */
 DMAX = (N>M) ? N : M;
 for (I=0; I<DMAX; I++)
    for (J=0; J<I; J++)
       {
        AIDE = A[I][J];
        A[I][J] = A[J][I];
        A[J][I] = AIDE;
       }
```

```
 /* Edition du résultat */
/* Attention: maintenant le rôle de N et M est inversé. */
 printf("Matrice résultat :\n");
for (I=0; I<M; I++)
   {
    for (J=0; J<N; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
return 0;
```

Exercice 25 :

a) Le résultat de la multiplication sera mémorisé dans une deuxième matrice A qui sera ensuite affichée.

```
#include <stdio.h>
main()
{ /* Déclarations */
 float A[50][50]; /* matrice donnée   */
 float B[50][50]; /* matrice résultat */
 int N, M;        /* dimensions des matrices */
 int I, J;        /* indices courants */
 float X;         /* multiplicateur   */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%f", &A[I][J]);
      }
 printf("Multiplicateur X : ");
 scanf("%f", &X );
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%10.2f", A[I][J]);
    printf("\n");
   }
 /* Affectation du résultat de la multiplication à B */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
```

```
        B[I][J] = X*A[I][J];
 /* Edition du résultat */
 printf("Matrice résultat :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%10.2f", B[I][J]);
    printf("\n");
   } return 0;
}
```

b) Les éléments de la matrice A seront multipliés par X.

```
#include <stdio.h>
main()
{ /* Déclarations */
 float A[50][50]; /* matrice donnée    */
 int N, M;       /* dimensions de la matrice */
 int I, J;       /* indices courants */
 float X;        /* multiplicateur   */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%f", &A[I][J]);
      }
 printf("Multiplicateur X : ");
 scanf("%f", &X);
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%10.2f", A[I][J]);
    printf("\n");
   }
 /* Multiplication des éléments de A par X */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
        A[I][J] *= X;
 /* Edition du résultat */
 printf("Matrice résultat :\n");
 for (I=0; I<N; I++)
```

```
   {
    for (J=0; J<M; J++)
        printf("%10.2f", A[I][J]);
    printf("\n");
   }
 return 0;
}
```

Exercice 26 :

a) Le résultat de l'addition sera mémorisé dans une troisième matrice C qui sera ensuite affichée.

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50][50]; /* matrice donnée   */
 int B[50][50]; /* matrice donnée   */
 int C[50][50]; /* matrice résultat */
 int N, M;      /* dimensions des matrices */
 int I, J;      /* indices courants */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 printf("*** Matrice A ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 printf("*** Matrice B ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &B[I][J]);
      }
 /* Affichage des matrices */
 printf("Matrice donnée A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
```

```
 printf("Matrice donnée B :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
 /* Affectation du résultat de l'addition à C */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
         C[I][J] = A[I][J]+B[I][J];
  /* Edition du résultat */
 printf("Matrice résultat C :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", C[I][J]);
    printf("\n");
   }
 return 0;
}
```

b) La matrice B est ajoutée à A.

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50][50]; /* matrice donnée et résultat */
 int B[50][50]; /* matrice donnée   */
 int N, M;      /* dimensions des matrices */
 int I, J;      /* indices courants */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 printf("*** Matrice A ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {       printf("Elément[%d][%d] : ",I,J);
              scanf("%d", &A[I][J]);
      }
 printf("*** Matrice B ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {       printf("Elément[%d][%d] : ",I,J);
              scanf("%d", &B[I][J]);
      }
 /* Affichage des matrices */
 printf("Matrice donnée A :\n");
```

```
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 printf("Matrice donnée B :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
 /* Addition de B à A */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
        A[I][J] += B[I][J];
  /* Edition du résultat */
 printf("Matrice résultat A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 return 0;
}
```

Exercice 27 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int A[50][50]; /* matrice donnée   */
 int B[50][50]; /* matrice donnée   */
 int C[50][50]; /* matrice résultat */
 int N, M, P;   /* dimensions des matrices */
 int I, J, K;   /* indices courants */
 /* Saisie des données */
 printf("*** Matrice A ***\n");
 printf("Nombre de lignes de   A (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes de A (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {       printf("Elément[%d][%d] : ",I,J);
              scanf("%d", &A[I][J]);
      }
```

```
 printf("*** Matrice B ***\n");
 printf("Nombre de lignes de   B : %d\n", M);
 printf("Nombre de colonnes de B (max.50) : ");
 scanf("%d", &P );
 for (I=0; I<M; I++)
   for (J=0; J<P; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &B[I][J]);
      }
 /* Affichage des matrices */
 printf("Matrice donnée A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 printf("Matrice donnée B :\n");
 for (I=0; I<M; I++)
   {
    for (J=0; J<P; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
 /* Affectation du résultat de la multiplication à C */
 for (I=0; I<N; I++)
    for (J=0; J<P; J++)
       { C[I][J]=0;
         for (K=0; K<M; K++)
            C[I][J] += A[I][K]*B[K][J];
       }
  /* Edition du résultat */
 printf("Matrice résultat C :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<P; J++)
        printf("%7d", C[I][J]);
    printf("\n");
   }
 return 0;
}
```

Exercice 28 :

```
#include <stdio.h>
main()
{ /* Déclarations */
 int P[14][14]; /* matrice résultat  */
 int N;         /* degré du triangle */
 int I, J;      /* indices courants  */
 /* Saisie des données */
 do {
     printf("Entrez le degré N du triangle (max.13) : ");
    scanf("%d", &N);
 } while (N>13||N<0);
 /* Construction des lignes 0 à N du triangle: */
 /* Calcul des composantes du triangle jusqu'à */
 /* la diagonale principale. */
 for (I=0; I<=N; I++)
    {
     P[I][I]=1;
     P[I][0]=1;
     for (J=1; J<I; J++)
          P[I][J] = P[I-1][J] + P[I-1][J-1];
    }
  /* Edition du résultat */
 printf("Triangle de Pascal de degré %d :\n", N);
 for (I=0; I<=N; I++)
   {
    printf(" N=%2d", I);
    for (J=0; J<=I; J++)
        if (P[I][J])
             printf("%5d", P[I][J]);
    printf("\n");
   }
 return 0;
}
```

Exercice 29 :

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50];  /* matrice donnée */
 int MAX[50][50];/* matrice indiquant les maxima des lignes   */
 int MIN[50][50];/* matrice indiquant les minima des colonnes */
 int N, M;    /* dimensions des matrices */
 int I, J;    /* indices courants */
 int AIDE;    /* pour la détection des extréma */
 int C;       /* compteur des points-cols */
 /* Saisie des données */
```

```
printf("Nombre de lignes   (max.50) : ");
scanf("%d", &N );
printf("Nombre de colonnes (max.50) : ");
scanf("%d", &M );
for (I=0; I<N; I++)
  for (J=0; J<M; J++)
     {
      printf("Elément[%d][%d] : ",I,J);
      scanf("%d", &A[I][J]);
     }
/* Affichage de la matrice */
printf("Matrice donnée :\n");
for (I=0; I<N; I++)
  {
   for (J=0; J<M; J++)
       printf("%7d", A[I][J]);
   printf("\n");
  }
/* Construction de la matrice d'aide MAX */
/* qui indique les positions de tous les */
/* maxima sur une ligne. */
for (I=0; I<N; I++)
   { /* Recherche du maximum sur la ligne I */
    AIDE=A[I][0];
    for (J=1; J<M; J++)
        if (A[I][J]>AIDE) AIDE=A[I][J];
    /* Marquage de tous les maxima sur la ligne */
    for (J=0; J<M; J++)
        if (A[I][J]==AIDE)  /* ou bien :                */
            MAX[I][J]=1;   /* MAX[I][J] = (A[I][J]==AIDE) */
        else
            MAX[I][J]=0;
   }
/* Construction de la matrice d'aide MIN */
/* qui indique les positions de tous les */
/* minima sur une colonne. */
for (J=0; J<M; J++)
   { /* Recherche du minimum sur la colonne J */
    AIDE=A[0][J];
    for (I=1; I<N; I++)
        if (A[I][J]<AIDE) AIDE=A[I][J];
    /* Marquage de tous les minima sur la colonne J */
    for (I=0; I<N; I++)
        if (A[I][J]==AIDE)  /* ou bien :                */
            MIN[I][J]=1;   /* MIN[I][J] = (A[I][J]==AIDE) */
        else
            MIN[I][J]=0;
   }  /* Edition du résultat */
```

```
/* Les composantes qui sont marquées comme extréma */
/* dans MAX et dans MIN sont des points-cols. */
printf("Points - cols :\n");
for (C=0, I=0; I<N; I++)
    for (J=0; J<M; J++)
        if (MAX[I][J]&&MIN[I][J])
          { C++;
            printf("L'élément %d\test un maximum "
                 "sur la ligne %d\n"
                 "            \t et un minimum "
                 "sur la colonne %d\n", A[I][J], I, J);
          }
if (C==0)
   printf("Le tableau ne contient pas de points-cols.\n");
return 0;}
```