Université de Carthage
****
Institut Supérieur des Sciences Appliquées et de Technologie de Mateur
****
Département Informatique et Télécommunications

Atelier de programmation I
Langage C

Destiné aux étudiants des classes 1 ères année Licence

Mentions : Sciences de l'Informatique (SI), Ingénierie des Systèmes Informatique (ISI)

Parcours : Génie Logiciel & Systèmes d'Information (GLSI), Systèmes Embarqués & IoT (SE & IoT), Ingénierie des Réseaux & Systèmes (IRS)

©Mr Radhi Yazidi
radhi.yazidi@issatm.ucar.tn

Année Universitaire 2021/2022

Fiche Matière

Domaine : Sciences exactes et Technologies

Mentions : Sciences de l'Informatique (SI), Ingénierie des Systèmes Informatique (ISI)

Parcours : Génie Logiciel et Systèmes d'Information, Systèmes Embarqués & IoT, Ingénierie des Réseaux et Systèmes

Spécialité : Tronc commun

Semestre : Semestre 1

Unité d'enseignement : Algorithmique et Programmation 1

Elément constitutif : Atelier de programmation 1

Prérequis : Algorithmique (niveau baccalauréat)

Volume horaire : 10.5h Cours, 31.5h TP

Programme (Commission Nationale Sectorielle En Informatique) :

1. Les types abstraits de données
2. Les spécifications algébriques
3. Algorithmique de bases (Schéma séquentiel, Schéma conditionnel, Schéma Itératif)
4. Les procédures et les fonctions
5. Notion de programme
6. Présentation du langage de programmation C (Structure d'un langage C, les types scalaires, déclaration de variables, l'instruction d'affectation, les opérations d'Entrée/Sortie, l'instruction conditionnelle, l'instruction itérative, les fonctions, le passage de paramètres : par variable et par adresse)

Learning Outcomes :

A la fin de ces ateliers, l'étudiant sera capable de :

L.O.1 : Implémenter des algorithmes en utilisant les différents outils offerts par le langage C.

L.O.2 : Concevoir et développer des programmes en langage C.

L.O.3 : Exploiter les structures de contrôles et les structures de données simples offertes par le langage C afin de proposer des solutions optimales aux problèmes rencontrés.

SOMMAIRE

Atelier N° 1 : Introduction au langage C

Objectifs

- Comprendre les étapes de création d'un programme C.
- Se familiariser avec l'environnement Code::Blocks.
- Maîtriser les possibilités d'E/S en langage C.

Les étapes de création d'un programme C

Edition du code source

Elle consiste à saisir, à partir d'un clavier, le programme en langage C puis l'enregistrer dans un fichier avec l'extension **'. c'**.

Compilation

Elle consiste à traduire le programme source en langage machine, en faisant appel à un programme nommé compilateur. Le résultat de cette opération est un fichier objet portant l'extension **'. obj'**.

Edition des liens

Le fichier objet créé par le compilateur n'est pas directement exécutable. Le rôle de l'éditeur des liens est d'aller chercher dans les bibliothèques standards le module objet de chaque fonction utilisée (comme printf, scanf, sqrt ...).

Le résultat de cette étape est un fichier exécutable portant l'extension **'. exe'**.

Exécution

Ce programme pourra ultérieurement être exécutable sans qu'il soit nécessaire de faire appel à n'importe quel composant de l'environnement de programmation C.

L'environnement de développement intégré (EDI) Code::Blocks

Code::Blocks est un environnement de développement intégré (IDE en anglais) libre et multiplateforme. Il est écrit en C++ et utilise la bibliothèque wxWidgets. Code::Blocks est orienté C et C++, mais il supporte d'autres langages.

Il est disponible gratuitement sous les systèmes Windows (avec compilateur C++ MinGW intégré), Linux et Mac OS : www.codeblocks.org.

Applications

Exercice 1 : Edition d'un exemple

Editer, compiler et exécuter le programme C ci-dessus.

```
#include <stdio.h>
void main ()
      {float a = 5.7,b = 9.34;
       printf("\n\n\n****Ceci est mon premier programme C****\n");
       printf("\tLa somme de %f et %f = %f\n",a ,b , a+b);
       printf("\t%f * %f = %f\n",a ,b, a*b);
       printf("\ta=%f , b=%f , a - b = %f",a ,b ,a-b);
      }
```

Exercice 2 : Utilisation de la bibliothèque « math.h »

Editer, compiler et exécuter le programme C ci-dessus.

```
#include <stdio.h>
#include <math.h>
void main ()
      { int a, b ;
        printf ("Donner deux entiers : ") ;
        scanf ("%d%d", &a, &b) ;
        printf ("\nLe carré de %d est %f\n", a, pow(a,2)) ;
        printf ("\n %d + %d = %d\n", a, b, a+b) ;
        printf ("La racine carré de %d est %f\n", a, sqrt(a)) ;
      }
```

Exercice 3

Quels sont les résultats fournit par le programme suivant ?

```
#include <stdio.h>
void main ()
      {int i, n;
       i=0 ; n=i++ ;
       printf ("A : i=%d n= %d \n", i, n) ;
       i=0 ; n=++i ;
       printf ("B : i=%d n= %d \n", i, n) ;
      }
```

Exercice 4

Ecrire un programme C qui permet de saisir le rayon d'un cercle puis calculer et afficher sa surface. La valeur Pi doit être déclarée comme constante.

Exercice 5

Ecrire un programme C qui permet de saisir deux entiers, effectuer la permutation (sans utiliser une variable intermédiaire) puis les afficher.

Exercice 6

Ecrire un programme C qui permet de calculer et afficher la durée T (**en heures et minutes**) que met un avion pour parcourir une distance D avec une vitesse V.

Indications

Utiliser les opérateurs MOD (%) et DIV (/).

Utiliser le contrôle de gabarit.

Exemple

Si D=125 km et V=100 km/h, alors l'affichage sera sous cette forme 01 : 15

Exercice 7

Ecrire un programme C qui lit une fraction au format a/b où a et b sont deux entiers, et donne son équivalent décimal avec une précision de quatre chiffres après la virgule.

Exemple

Si l'utilisateur entre 3/2, le programme doit afficher : 3/2 = 1.5000

Exercice 8

Ecrire un programme C qui lit deux entiers est affiche le plus grand d'entre eux.

Soient a et b deux nombres quelconques, leur max est donné par :
max (a, b) = (a + b + |a - b|) /2 ; Où |a-b| est la valeur absolue de la différence (a-b).

Exercice 9

Ecrire un programme C qui lit en entrée un caractère alphabétique entre a et y, qui peut être soit une majuscule ou une minuscule, puis affiche la lettre qui vient juste après lui dans l'ordre alphabétique.

Exercice 10

Ecrire un programme C qui lit en entrée trois entiers et affiche leur moyenne avec une précision de deux chiffres après la virgule.

Compléments – A

La compilation informatique

La compilation informatique désigne le procédé de traduction d'un programme, écrit et lisible par un humain, en un programme exécutable par un ordinateur. De façon plus globale, il s'agit de la transformation d'un programme écrit en code source, en un programme transcrit en code cible, ou binaire. Habituellement, le code source est rédigé dans un langage de programmation (langage source), il est de haut niveau de conception et facilement accessible à un utilisateur. Le code cible, quant à lui, est transcrit en langage de plus bas niveau (langage cible), afin de générer un programme exécutable par une machine.

Utilisation des bibliothèques de fonctions

La pratique en C exige l'utilisation de bibliothèques de fonctions. Ces bibliothèques sont disponibles dans leur forme précompilée (extension : **.LIB**). Pour pouvoir les utiliser, il faut inclure des fichiers entête (header files - extension **.H**) dans nos programmes. Ces fichiers contiennent des prototypes des fonctions définies dans les bibliothèques et créent un lien entre les fonctions précompilées et nos programmes.

Identification des fichiers

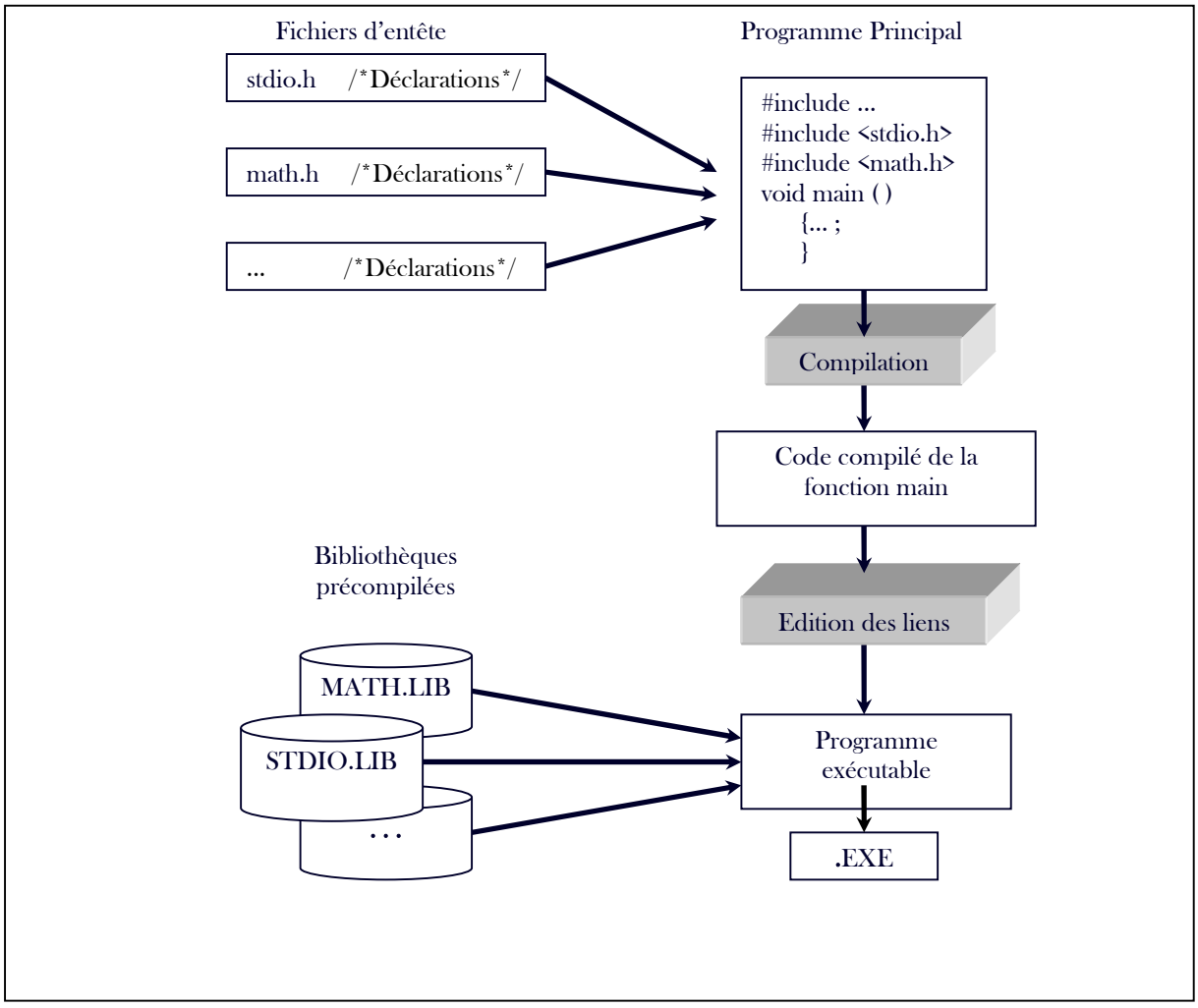
Lors de la programmation en langage C, nous travaillons avec différents types de fichiers qui sont identifiés par leurs extensions : *.C fichiers sources, *.OBJ fichiers compilés, *.EXE fichiers exécutables, *.LIB bibliothèques de fonctions précompilées, *.H fichiers d'entêtes.

Exemple

Nous avons écrit un programme qui fait appel à des fonctions mathématiques prédéfinies. Pour pouvoir utiliser ces fonctions, le programme a besoin de la bibliothèque : **MATH.LIB**

Nous devons donc inclure les fichiers entête correspondants dans le code source de notre programme à l'aide de l'instruction : **#include <math.h>**

Après la compilation, les fonctions précompilées de la bibliothèque seront ajoutées à notre programme pour former une version exécutable (voir schéma).


Fichiers d'entête
stdio.h /*Déclarations*/
math.h /*Déclarations*/
... /*Déclarations*/
Programme Principal
#include ...
#include <stdio.h>
#include <math.h>
void main ( )
{... ;
}
Compilation
Code compilé de la fonction main
Bibliothèques précompilées
MATH.LIB
STDIO.LIB
...
Edition des liens
Programme exécutable
.EXE

Atelier N° 2 : Les instructions de contrôle

Objectifs

- Résoudre des problèmes en utilisant les instructions de contrôle.
- Faire la différence entre les différentes instructions de contrôle répétitives.

Les instructions de contrôle conditionnelles

Exercice 1

Ecrire un programme C qui permet de résoudre l'équation suivante :

$$a\ x + b = 0$$

a et b étant deux réels.

Exercice 2

Ecrire un programme C qui permet de réaliser les opérations de base d'une calculatrice (+, - , *, /) à partir de trois données saisies par l'utilisateur (deux opérandes et un opérateur).

Exercice 3

Proposer un programme C qui permet de saisir un caractère puis faire la discussion suivante :

- Si ce caractère est alphabétique minuscule alors il sera affiché en majuscule.
- S'il est alphabétique majuscule alors il sera affiché en minuscule.
- Sinon un message d'erreur sera affiché.

Exercice 4

Ecrire un programme C qui permet de saisir un nombre N et tester s'il est cubique.
Un nombre est dit cubique s'il est égal à la somme des cubes de ses chiffres.
$153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

Les instructions de contrôle répétitives

Exercice 5

Ecrire un programme C qui calcule et affiche la somme des N premiers termes de la série harmonique :
$1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N$.
La valeur de N doit être comprise entre 2 et 20.

Exercice 6

Proposer un programme C qui calcule le quotient et le reste de la division d'un entier N par un entier D en utilisant seulement les deux opérateurs (++ , –).

Exercice 7

Proposer un programme C qui permet de calculer la factorielle (N! = 1*2*3*... *(N-1)*N) d'un entier naturel N en respectant que 0! = 1. La valeur de N doit être comprise entre 0 et 7.

Modifier le programme de telle façon qu'il accepte les valeurs de N comprises entre 8 et 15.

Exercice 8

Calculer la moyenne de notes fournies au clavier avec un dialogue de ce type :

```
Note 1 : 12
Note 2 : 15.25
Note 3 : 13.5
Note 4 : 8.75
Note 5 : -1
Moyenne de ces 4 notes : 12.37
```

Le nombre de notes n'est pas connu à priori et l'utilisateur peut en fournir autant qu'il le désire. Pour signaler qu'il a terminé, on convient qu'il fournira une note fictive négative. Celle-ci ne devra pas être prise en compte dans le calcul de la moyenne.

Exercice 9

Ecrire un programme C qui permet de saisir autant de nombres que l'utilisateur le veuille, et de déterminer le nombre de réels strictement positifs et celui des négatifs. On s'arrête lorsque la valeur est 0.

Exercice 10

Ecrire un programme C qui calcule la $n^{ième}$ valeur de la Suite de Fibonacci (une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent) définie par :

$U_0 = 1$, $U_1 = 1$

$U_n = U_{n-1} + U_{n-2}$ avec n >=2

Exercice 11

Ecrire un programme C qui affiche un menu comportant 4 choix (numérotés de 1 à 4). En voulant quitter, l'utilisateur doit taper « 0 ».

Atelier N° 3 : Les tableaux

Objectifs

- Être capable de manipuler un tableau (initialisation, saisie, affichage ...).
- Maîtriser la manipulation avancée des tableaux (parcours, recherche, tri, ...).

Les tableaux unidimensionnels

Exercice 1

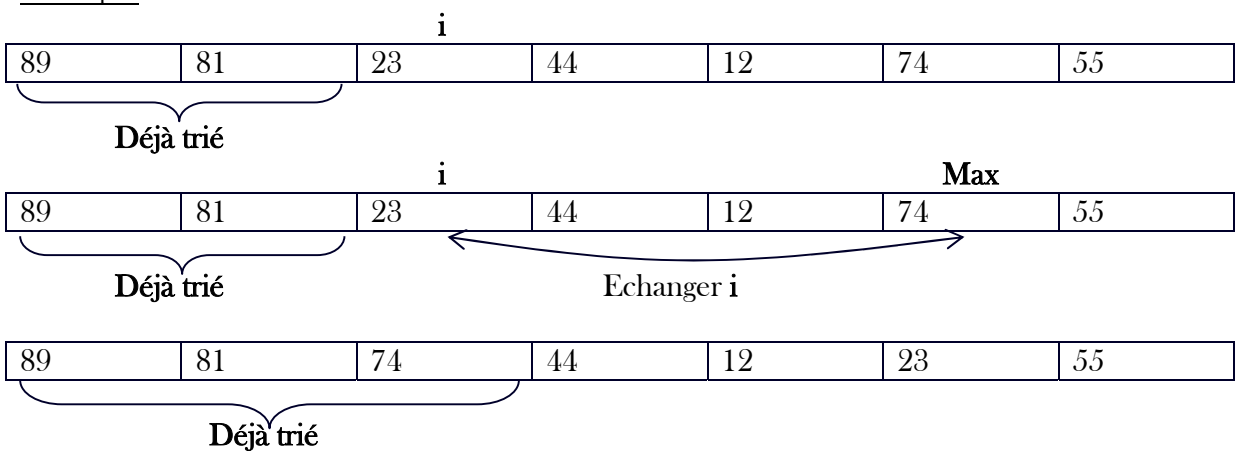
Proposer un programme C qui permet de saisir un tableau de caractères (dimension maximale : 20). Vérifier ensuite si l'ensemble des caractères saisis forme une chaîne palindrome.

Exercice 2

Proposer un programme C qui permet de saisir un tableau d'entiers (dimension maximale : 20). Trier ensuite le tableau par ordre décroissant (utiliser la méthode de tri par sélection).

Principe : parcourir le tableau de gauche à droite à l'aide de l'indice **i**. Pour chaque élément **T[i]** du tableau, déterminer la position **Max** du (premier) maximum à droite de **T[i]** et échanger **T[i]** et **T [Max]**.

Exemple



Les tableaux bidimensionnels

Exercice 3

Proposer un programme C qui permet de saisir un tableau d'entiers à deux dimensions N et M (dimension maximale : 10). Afficher ensuite le tableau sous la forme de N lignes et M colonnes.

Exercice 4

Proposer un programme C qui permet de saisir une matrice et un vecteur, effectuer la multiplication et stocker le résultat dans un autre vecteur puis l'afficher.

Exercice 5

Proposer un programme C qui permet de saisir une matrice carrée de taille N*N, puis effectuer sa transposition par permutation des éléments.

Atelier N° 4 : Les chaînes de caractères

Objectifs

- Comprendre le fonctionnement de quelques fonctions prédéfinies sur les chaînes de caractères.
- Maîtriser la manipulation avancée des chaînes de caractères (recherche de nombre de mots, insertion d'une chaîne dans une autre ...).

Exercice 1

Proposer un programme C qui permet de saisir une chaîne de caractères **CH1**, déterminer et afficher sa longueur, la copier dans une autre chaîne **CH2** et enfin afficher cette dernière.

N.B : N'utiliser aucune fonction prédéfinie (**strlen**, **strcpy**).

Exercice 2

Ecrire un programme C qui supprime toutes les lettres « e » (minuscules) présentes dans une phrase (max. 100 caractères) fourni au clavier. Le texte ainsi modifié sera créé, en mémoire, à la place de l'ancien.

Exercice 3

Proposer un programme C qui permet de saisir deux chaînes de caractères et un entier N puis insérer la deuxième chaîne dans la première à partir du $N^{ième}$ caractère (le résultat sera stocké dans la première chaîne). Enfin, afficher le résultat.

Exemple

Chaîne 1 : « Institut des sciences appliquées »

Chaîne 2 : « supérieur »

N = 8

La chaîne 1 devient : « Institut supérieur des sciences appliquées »

N.B : N'utiliser aucune fonction prédéfinie (**strncat**).

Exercice 4

Proposer un programme C qui permet de saisir une chaîne de caractères **minuscule**, puis la transformer en majuscule. Enfin, afficher la chaîne de nouveau.

Indications

- La saisie doit être contrôlée caractère par caractère (Penser aux deux fonctions **getche ( )** et **islower ( )**).
- Les caractères différents des lettres minuscules ne seront pas affichés (Penser à la séquence d'échappement **\b**).
- La saisie se termine par une entrée (code ASCII **13**).

Compléments – B

Quelques fonctions sur les chaînes de caractères (string.h)

```
size_t strlen (const char *s) //Retourner la longueur de la chaîne s

char *stpcpy (char *s1, const char *s2) // Copier s2 dans s1

char *strcat (char *s1, const char *s2) /* Concaténer s2 à la fin de
s1*/

int strcmp (const char *s1, const char*s2) /* Comparer s1 et s2
suivant l'ordre lexicographique et retourner un résultat :
                < 0       Si      s1 < s2
                = = 0    Si      s1 = = s2
                > 0       Si      s1 > s2 */

char *strncat (char *s1, const char *s2, size_t n) /* Concatène les
n premiers caractères de s2 après s1 */

int strncmp (const char *s1, const char *s2, size_t n) /* Comparer
les n premiers caractères de s1 et s2 */
```

Quelques fonctions sur les caractères (ctype.h)

Les fonctions suivantes prennent en paramètre un caractère et retournent une valeur différente de zéro (0) en cas de succès :

```
int isalpha (int c) // vérifie si c appartient à 'a…z' ou 'A…Z'
int islower (int c) // vérifie si c appartient à 'a…z'
int isdigit  (int c) // vérifie si c est un chiffre (0…9)
int ispunct  (int c) /* vérifie si c est un caractère de ponctuation
(: ; ! ?  , .) */
int isspace (int c) /* vérifie si c est un caractère d'espacement
(' ' '\n' '\t' '\v' '\r' …)*/
```

Saisie et Affichage d'une chaîne de caractères (stdio.h)

```
char *gets (char *s) /* s désigne le nom de la variable qui va
stocker la chaîne.
     gets ajoute à la fin de s le zéro de fin de chaîne (\0).
     En cas d'erreur, gets retourne le pointeur NULL. */

int puts (const char *s) /* s est la chaîne à afficher.
     L'affichage sera suivi d'une insertion d'une nouvelle ligne
(\n).
     En cas d'erreur, puts retourne la valeur 0.*/
```

Atelier N° 5 : Les pointeurs & les tableaux

Objectifs

- Comprendre la notion des pointeurs en langage C.
- Maîtriser l'utilisation des pointeurs avec les tableaux.
- Manipuler l'allocation dynamique de la mémoire.

Exercice 1 (Théorique)

Donner le contenu des variables après l'exécution de chaque exemple.

```
int a = 10 ;
int * adr ;
adr = &a ;
printf ("%d", *adr) ;
```

```
int * pt, i, j ;
i = 5 ;
pt = &i ;
j = *pt + 4 ;
printf ("%d", j) ;
```

```
int var = 5 ;
int *p ;
p = &var ;
printf ("%d \n", *p);
*p = 4 ;
printf ("%d", var) ;
```

Exercice 2

Donner le contenu des variables après l'exécution de chaque instruction.

```
void main()
      {int A=1,B=2,C=3,*P1,*P2;
       P1=&A;
       P2=&C;
       *P1=(*P2)++;
       P1=P2;
       P2=&B;
       *P1 -=*P2;
       ++*P2;
       *P1*=*P2;
      }
```

	A	B	C	P1	P2
Initialisation	1	2	3	/	/
P1=&A	1	2	3	&A	/
P2=&C					
***P1=(*P2)++**					
P1=P2					
P2=&B					
***P1-=*P2**					
++*P2					
***P1*=*P2**					

Editer le programme (en ajoutant les instructions d'affichage) et vérifier le résultat trouvé.

Exercice 3

En utilisant la notion des pointeurs, proposer un programme C qui permet de saisir un tableau **T** de **N** entiers (**N** étant saisi par l'utilisateur), puis recopier les éléments pairs dans un autre tableau **R**. Enfin, afficher le tableau résultat.

N.B : Utiliser l'allocation dynamique de la mémoire.

Exercice 4

Ecrire un programme qui lit deux tableaux d'entiers A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A. Utiliser deux pointeurs PA et PB pour le transfert, enfin afficher le tableau résultant A.

Exercice 5

Ecrire un programme C qui réserve l'espace mémoire à un tableau d'entiers à deux dimensions dont la taille est entrée par l'utilisateur, puis le lit et l'affiche.

Résumé sur la gestion dynamique de la mémoire

Les principales fonctions C de gestion dynamique de la mémoire sont **malloc**, **calloc**, **realloc** et **free**. Elles sont définies dans **stdlib.h** et **alloc.h**.

La fonction malloc : void * malloc (size_t size) ;

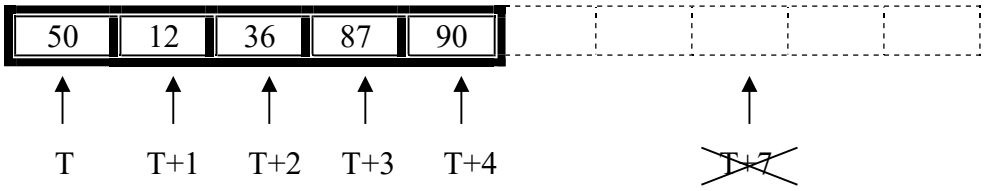
La fonction free : void free (void * zone) ;

Exemple

```
#include <stdio.h>
#include <alloc.h>
void main ( )
     {int * adr, i ;
       adr = (int *) malloc(10 * sizeof (int)) ;
       for (i = 0 ; i<10 ; i++)
           * (adr + i) = 1 ;
       for (i = 0 ; i < 10 ; i++)
           printf ("%d\t", * (adr+i)) ;
       free (adr) ;
     }
```

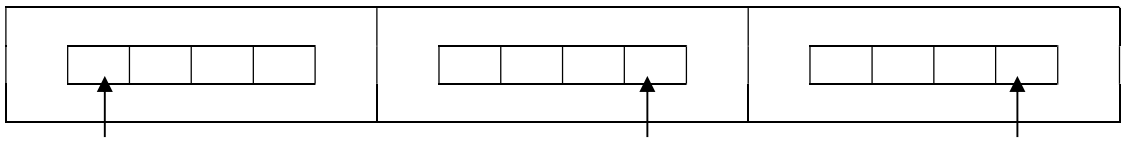
Représentation des tableaux dans la mémoire

```
int T [5] = {50, 12, 36, 87, 90} ;
```



N.B : A la compilation, *(T+7) ne sera pas signalé comme erreur mais on aura des valeurs erronées pendant l'exécution.

```
int ** M, L, C ;
```

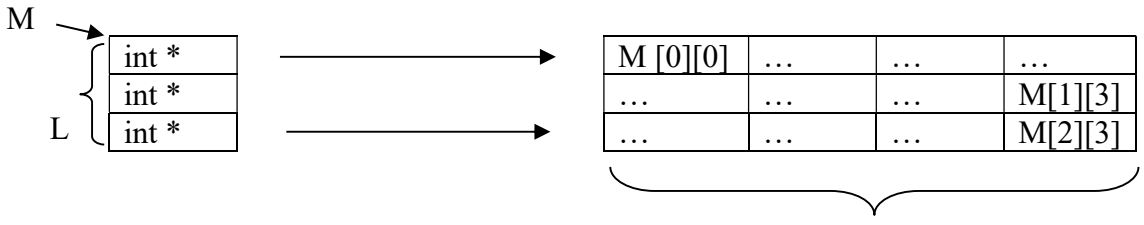


M [0][0] *(*(M + 0) + 0)

M [1][3] *(*(M + 1) + 3)

M [2][3] *(*(M + 2) + 3)

En effet, on a un tableau de pointeurs où chaque case pointe sur un tableau d'entiers.



Atelier N° 6 : Les fonctions

Objectifs

- Maîtriser la syntaxe liée aux fonctions (définition, valeur de retour, paramètres, etc.).
- Comprendre les modes de passage des paramètres aux fonctions.

Exercice 1

1. Proposer la définition d'une fonction « **Min** » qui permet de retourner le minimum de deux nombres réels passés en paramètres.
2. Proposer la définition d'une fonction « **Moyenne** » qui permet d'afficher la moyenne de deux nombres réels passés en paramètres.
3. Ecrire un programme C qui permet de saisir deux nombres réels puis afficher le minimum entre eux et leur moyenne.

Exercice 2

1. Proposer la définition d'une fonction « **Saisie** » qui permet de saisir un tableau **T** de **N** entiers.
2. Proposer la définition d'une fonction « **Affiche** » qui permet d'afficher un tableau **T** de **N** entiers.
3. Proposer la définition d'une fonction « **Inverse** » qui permet de ranger un tableau **T** de **N** entiers dans l'ordre inverse.
4. Ecrire un programme C qui permet de saisir un tableau **TAB** de **X** entiers, le ranger dans l'ordre inverse puis l'afficher.

Exercice 3

Ecrire une fonction qui reçoit en arguments 2 nombres flottants et un caractère et qui retourne un résultat correspondant à l'une des 4 opérations appliquées à ses deux premiers arguments, en fonction du dernier, à savoir : addition pour le caractère '+', soustraction pour '-', multiplication pour '*' et division pour '/'.
N.B : N'oublier pas de traiter les cas particuliers (division par 0, un caractère différend de +, *…)

Ecrire un petit programme (main) utilisant cette fonction pour effectuer les 4 opérations sur deux nombres fournis en donnée.

Atelier N° 7 : Les structures

Objectifs

- Savoir définir de nouveaux types sous le langage C.
- Être capable de manipuler les structures.
- Maîtriser la manipulation de tableaux de structures.

Exercice 1 : Manipulation simple d'une structure

1. Proposer la définition d'une structure « Date » contenant trois champs : jours, mois et année.
2. Donner la définition d'une fonction qui permet de saisir une date valide (la saisie doit être contrôlée).
3. Proposer une fonction qui nous permet de comparer deux dates et retourner la plus récente d'entre elles.
4. Donner la définition d'une fonction permettant d'afficher une date.

Ecrire un programme C qui saisit deux dates puis affiche la plus récente d'entre elles.

Exercice 2 : Manipulation d'un tableau de structures

1. Donner la définition d'une structure « Point » permettant de coder les coordonnées d'un point dans un espace à deux dimensions.
2. Donner les définitions des fonctions *get_Point* et *print_Point* qui nous permettent successivement de saisir et d'afficher un point.
3. Étant donné la structure « Point » définie précédemment, indiquer quelle est la structure de données adéquate permettant de coder une courbe ?
4. Ecrire un programme C qui saisit les informations concernant une courbe puis les affiche.

Exercice 3 : Imbrication des structures

1. Proposer des définitions pour les structures suivantes :
    - Date : jour, mois, année.
    - Personne : nom, prénom et date de naissance.
2. Donner des définitions pour les fonctions suivantes :

- *Saisie_Date* : permet de saisir une date valide.
- *Affiche_Date* : permet d'afficher une date.
- *Saisie_Personne* : permet de saisir les informations d'une personne.
- *Affiche_Personne* : permet d'afficher les informations d'une personne.

3. Ecrire un programme C qui permet de saisir une personne puis l'afficher.