Exemple 3

```
#include <stdio.h>              /* bibliotheque d'entrees-sorties standard */
#include <conio.h>
void main()
{
 int a, b, calcul ;  /* déclaration de 3 variables */
 char u, v;
 printf("BONJOUR");       /* utilisation d'une fonction-bibliotheque */
 a = 10 ;  /* affectation* /
 b = 50 ;  /* affectation */
 u = 65 ;
 v = 'A' ;
 calcul = (a + b)*2 ;  /* affectation et opérateurs   */
 printf(« Voici le resultat : %d\n », calcul) ;
 printf(« 1er affichage de u : %d\n »,u) ;
 printf(« 2ème affichage de v : %c\n »,u) ;
 printf(« 1er affichage de u: %d\n »,v) ;
 printf(« 2éme  affichage de v: %c\n »,v) ;
 puts("Pour continuer frapper une touche...");
 getch();  /* Attente d'une saisie clavier */
}
```

Résultat après compilation et exécution:

```
BONJOUR
 Voici le resultat : 120
  1er affichage de u : 65
  2ème affichage de v : A
  1er affichage de u: 65
  2ème  affichage de v: A
 Pour continuer frapper une touche...
```


b) Les réels

Un réel est composé - d'un signe - d'une mantisse - d'un exposant
Un nombre de bits est réservé en mémoire pour chaque élément.

Le langage C distingue 2 types de réels:

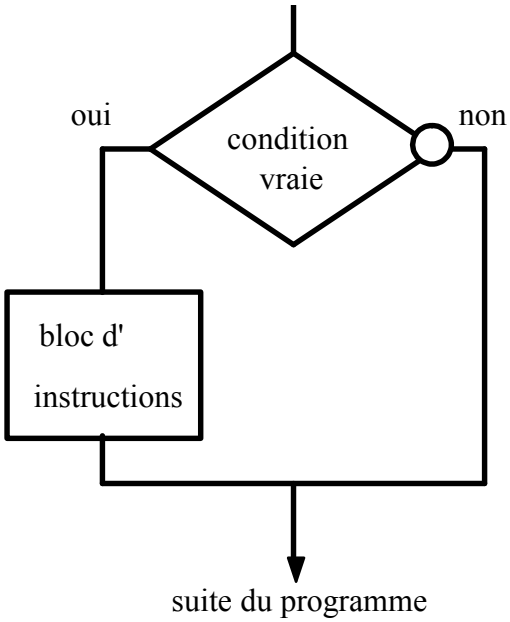
TYPE	DESCRIPTION	TAILLE MEMOIRE
float	réel standard	4 octets
double	réel double précision	8 octets

4. LES INITIALISATIONS

Le langage C permet l'initialisation des variables dans la zône des déclarations:

char c; est équivalent à char c = 'A';
c = 'A';
int i; est équivalent à int i = 50;
i = 50;
Cette règle s'applique à tous les nombres, char, int, float ...

Le bloc "sinon" est optionnel: **si (expression vraie)**
alors {BLOC D'INSTRUCTIONS}



Syntaxe en C: **if (expression)**

```
{
.....;          /* bloc d'instructions */
.....;
.....;
}
```

Remarque: les {} ne sont pas nécessaires lorsque les blocs ne comportent qu'une seule instruction.

2. LES OPERATEURS LOGIQUES

test d'égalité: **if (a==b)** **" si a égal b "**

test de non égalité: **if (a!=b)** **" si a différent de b "**

tests de relation d'ordre: **if (a<b)** **if (a<=b)** **if (a>b)** **if (a>=b)**

test de ET LOGIQUE: **if ((expression1) && (expression2))**
" si l'expression1 ET l'expression2 sont vraies "

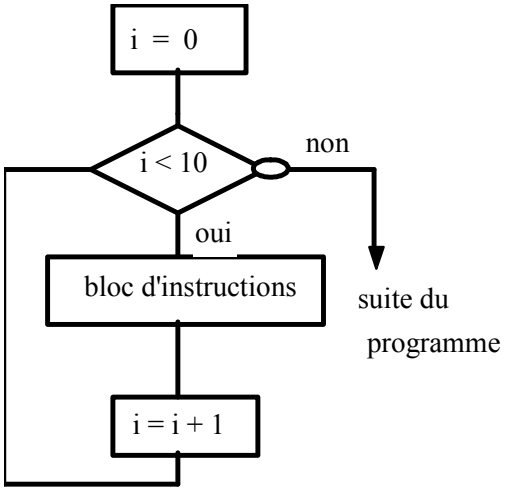
test de OU LOGIQUE **if ((expression1) || (expression2))**
" si l'expression1 OU l'expression2 est vraie "

test de NON LOGIQUE **if (!(expression1))**
" si l'expression1 est fausse "

Exemples:

```
for(i = 0 ; i<10 ; i++)
{
.....;                  /* bloc d'instructions */
.....;
.....;
}
```

correspond à l'organigramme suivant:



La boucle

```
for(;;)
{
.....;                  /* bloc d'instructions */
.....;
.....;
}
```

est une boucle infinie (répétition infinie du bloc d'instructions).
Utilisation de variables différentes:

```
resultat = 0;
for(i = 0 ; resultat<30 ; i++)
        {
        .....;                  /* bloc d'instructions */
        .....;
        .....;
        resultat = resultat + 2*i;
        }
```

Exercice: Ecrire un programme C qui permet de saisir un entier puis calcule sa factoriel.

6. L'INSTRUCTION REPETER ... TANT QUE ...

Il s'agit de l'instruction: **répéter{BLOC D'INSTRUCTIONS}**
tant que (expression vraie)

2. INITIALISATION DES TABLEAUX

On peut initialiser les tableaux au moment de leur déclaration:

Exemples:

int liste[10] = {1,2,4,8,16,32,64,128,256,528};

float nombre[4] = {2.67,5.98,-8,0.09};

int x[2][3] = {{1,5,7},{8,4,3}}; /* 2 lignes et 3 colonnes */

3. LES CHAINES DE CARACTERES

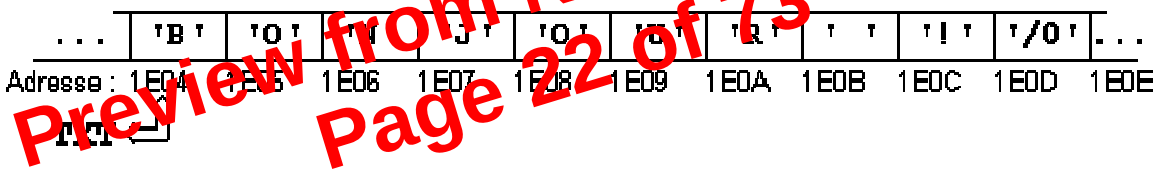
En langage C, les chaînes de caractères sont des **tableaux de caractères**. Leur manipulation est donc analogue à celle d'un tableau à une dimension:

Déclaration: **char nom[dim];**

Exemple: **char texte[10];**

Le compilateur réserve (dim-1) places en mémoire pour la chaîne de caractères. En effet, il ajoute toujours le caractère **NUL ('\0')** à la fin de la chaîne en mémoire.

char TXT[10] = "BONJOUR !";



Affichage à l'écran:

On peut utiliser la fonction **printf** et le format %s:
char texte[10] = "BONJOUR";
printf("VOICI LE TEXTE: %s\n",texte);

On utilisera si possible la fonction **puts** non formatée:
puts(texte); est équivalent à **printf("%s\n",texte);**

Saisie: On peut utiliser la fonction **scanf** et le format %s. Une chaîne étant un pointeur, on n'écrit pas le symbole &. On utilisera de préférence la fonction **gets** non formatée.

char texte[10];
printf("ENTRER UN TEXTE: ");
scanf("%s",texte); est équivalent à **gets(texte);**

```
void main()
{
  int n1, n2, res1, res2;                /* variables locales */
  printf("ENTRER UN NOMBRE: ");
  scanf("%d",&n1);
  res1 = carre(n1);
  printf("ENTRER UN AUTRE NOMBRE: ");
  scanf("%d",&n2);
  res2 = carre(n2);
  printf("VOICI LEURS CARRES: %d   %d\n\n",res1, res2);
  printf("POUR SORTIR FRAPPER UNE TOUCHE: ");
  getch();
}
```

On peut ainsi appeler la fonction carre autant de fois que l'on veut avec des variables différentes.
x est un paramètre, ou argument: ce n'est pas une variable du programme.

S'il y a plusieurs arguments à passer, il faut respecter la syntaxe suivante:

Exemples: void fonction1(int x, int y) void fonction2(int a, float b, char c)

6. RESUME SUR VARIABLES ET FONCTIONS

On a donc vu qu'une variable **globale** est déclarée au début du programme et qu'elle est connue de tout le programme. **Les variables globales sont initialisées à 0 au début de l'exécution du programme, sauf si on les initialise à une autre valeur.**

On a vu aussi qu'une variable **locale** (déclarée au début d'une fonction ou de main()) n'est connue que de cette fonction ou de main(). Une variable locale est encore appelée **automatique.**

Les variables locales ne sont pas initialisées (sauf si on le fait dans le programme) et elles perdent leur valeur à chaque appel à la fonction.

On peut allonger la durée de vie d'une variable locale en la déclarant **static.** Lors d'un nouvel appel à la fonction, la variable garde la valeur obtenue à la fin de l'exécution précédente. Une variable **static** est initialisée à 0 lors du premier appel à la fonction.

Exemple: **int i;** devient **static int i;**

7. LE PASSAGE DE PARAMETRES ENTRE FONCTIONS OU ENTRE FONCTIONS ET PROGRAMME PRINCIPAL

En langage C, le passage de paramètre se fait uniquement par adresse. Autrement dit, **une fonction ne peut pas modifier la valeur des variables locales à main() ou à une autre fonction. Elle ne peut modifier que le contenu de l'adresse de cette variable.**

REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUES ET TECHNOLOGIQUES

UNIVERSITE DE JENDOUBA
FACULTE DES SCIENCES JURIDIQUES, ECONOMIQUES ET DE GESTION DE JENDOUBA

Année Universitaire : 2009/2010
Semestre : 1
Module : Atelier de programmation I
Classe : 1ère année LFIAG
Enseignant : Riadh BOUSLIMI

TP n°1

Exercice 1

Ecrire un programme qui saisit 2 entiers et affiche successivement la somme, la différence, le produit et le quotient de ces 2 entiers.

Exercice 2

Ecrire un programme qui demande à l'utilisateur de donner le rayon d'un cercle et lui retourne sa surface et son périmètre.

Exercice 3

Ecrire un programme qui saisit deux entiers a et b et permute la valeur de ces deux entiers.

Exercice 4

Ecrire un programme qui saisit un réel x et un entier n et affiche x à la puissance n.

Exercice 5

Ecrire un programme qui saisit un caractère c puis affiche son code ASCII en décimal, en octal(base huit) et en hexadécimal(base seize).

base: base dans laquelle est exprimé le nombre,
cette fonction renvoie l'adresse de la chaîne.

exemple: itoa(12,texte,10); texte vaut "12"

Exercice n°1

Écrire 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 n°2

Écrire 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 n°3

Écrire 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 n°4

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 trouvé.

Implémenter deux versions:

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

b) La recherche dichotomique

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.

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

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

Exercice n°5

Ecrire un programme qui demande l'introduction du nom et du prénom de l'utilisateur et qui affiche alors la longueur totale du nom sans compter les espaces. Employer la fonction **strlen**.

Exemple:

```
Introduisez votre nom et votre prénom: HSINI Sana

Bonjour HSINI Sana !
Votre nom est composé de 9 lettres.
```

Exercice n°6

Écrire un programme qui lit 5 mots, séparés par des espaces et qui les affiche ensuite dans une ligne, mais dans l'ordre inverse.

Les mots sont mémorisés dans 5 variables M1, ... ,M5.

Exemple

```
voici une petite phrase !
! phrase petite une voici
```

Exercice n°7

Écrire un programme qui lit une ligne de texte (ne dépassant pas 200 caractères) la mémorise dans une variable TXT et affiche ensuite:

a) la longueur L de la chaîne.

b) le nombre de `'e'` contenus dans le texte.

c) toute la phrase à rebours, sans changer le contenu de la variable TXT.

d) toute la phrase à rebours, après avoir inversé l'ordre des caractères dans TXT:

Exemple :

```
voici une petite phrase !
! esarhp etitep enu iciov
```

Exercice n°8

Ecrire un programme qui lit un texte TXT (de moins de 200 caractères) et qui enlève toutes les apparitions du caractère 'e' en tassant les éléments restants. Les modifications se feront dans la même variable TXT.

Exemple:

```
Cette ligne contient quelques lettres e.
Ctt lign contint qulqus lttrs .
```


Exercice n°9

Ecrire un programme qui lit un verbe régulier en "er" au clavier et qui en affiche la conjugaison au présent de l'indicatif de ce verbe. Contrôlez s'il s'agit bien d'un verbe en "er" avant de conjuguer. Utiliser les fonctions **gets**, **puts**, **strcat** et **strlen**.

Exemple:

```
Verbe : fêter
je fête  tu fêtes  il fête  nous fêtons  vous fêtez  ils fêtent
```

Exercice n°10

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2, les compare lexicographiquement et affiche le résultat:

Exemple:

```
Introduisez la première chaîne: ABC
Introduisez la deuxième chaîne: abc
"ABC" précède "abc"
```

```
      AFFI[0]='\0';
      strcat(AFFI, "ils ");
      strcat(AFFI, VERB);
      strcat(AFFI, "ent");
      puts(AFFI);
     }
 printf("Tapez une touche pour continuer...");
 getch();
}
```

Exercice n°10

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
 /* Déclarations */
 char CH1[200], CH2[200]; /* chaînes entrées */
 int RES;  /* résultat de la fonction strcmp */

 printf("Introduisez la première chaîne de caractères : "); gets(CH1);
 printf("Introduisez la deuxième chaîne de caractères : "); gets(CH2);

 /* Comparaison et affichage du résultat */
 RES = strcmp(CH1,CH2);
 if (RES<0)
     printf("\"%s\" précède \"%s\"\n",CH1 ,CH2);
 else if (RES>0)
     printf("\"%s\" précède \"%s\"\n",CH2 ,CH1);
 else
     printf("\"%s\" est égal à \"%s\"\n",CH1, CH2);
  return 0;
}
```

Exercice n°11

```
#include<stdio.h>
#include<string.h>
#include<conio.h>

void main(){
/* Déclarations */
 char mot[30];
 int n,i;
 printf("Entrer un mot:");
 scanf("%s",mot);
 n=strlen(mot)-1;

 //1ere solution
 i=0;
 while((mot[i]==mot[n-i]) && (i<=(n/2))){
   i++;
 }

 /*
    //2ème solution
     i=-1; /*initialisation*/
     do{
         i++; /*incrementation*/
     }while((mot[i]!=mot[n-i]) || (i==(n/2)));

  */
 if(mot[i]==mot[n-i])
   printf("Le mot %s est palindrome \n",mot);
 else
   printf("Le mot %s n'est pas palindrome \n",mot);

 printf("Tapez une touche pour continuer...");
 getch();
 }
```

Correction du TP n°5

Exercice n°1

```
#include<stdio.h>
#include<conio.h>

float MOYENNE(float X, float Y)
{
 return (X+Y)/2;
}

void main()
{
/* Variables locales */
 float A,B;
 /* Traitements */
 printf("Introduire deux nombres : ");
 scanf("%f %f", &A, &B);
 printf("La moyenne arithmétique de %f et %f est %f\n",
                                          A, B, MOYENNE(A,B));
 printf("Tapez une touche pour continuer…");
 getch();
}
```

Exercice n°2

```
#include <stdio.h>
#include<conio.h>

double MIN(double X, double Y)
{
 if (X<Y)
     return X;
 else
     return Y;
}

double MAX(double X, double Y)
{
 if (X>Y)
     return X;
 else
     return Y;
}


void main()
{
/* Variables locales */
 double A,B,C,D;
 /* Traitements */
 printf("Introduire 4 réels : ");
 scanf("%lf %lf %lf %lf", &A, &B, &C, &D);
 printf("Le minimum des 4 réels est %f \n",
                                  MIN( MIN(A,B), MIN(C,D)) );
 printf("Le maximum des 4 réels est %f \n",
                                  MAX( MAX(A,B), MAX(C,D)) );
 printf("Tapez une touche pour continuer…");
 getch();

}
```

Exercice n°3 (9pts)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main(){
	char ch1[100];
	char ch2[100];
	int i,j;
	int LongeurCh1,LongeurCh2,nbOcc;
	printf("Entrer ch1:");gets(ch1);
	printf("Entrer ch2:");gets(ch2);
	LongeurCh1=strlen(ch1);
	LongeurCh2=strlen(ch2);
	nbOcc=0;
	i=0;
	while(i<LongeurCh1){
		j=0;
		while((ch1[i]!=ch2[j]) && (i<LongeurCh1))
			i++;

		while((ch1[i]==ch2[j]) && (i<LongeurCh1) && (j<LongeurCh2)){
			i++;
			j++;
		}

	  if (j==LongeurCh2)
			nbOcc++;

	}
	printf("\n Le nombre d'occurrences de %s dans %s est :
%d\n",ch2,ch1,nbOcc);
	printf("Tapez une touche pour continuer…");
	getch();
}
```

Université de Jendouba
Faculté des Sciences Juridiques, Economiques et de Gestion de Jendouba

Classe	**1ère année Licence Fondamentale en Informatique Appliquée à la Gestion**
Matière	**Atelier de programmation I**
Enseignant	**Riadh BOUSLIMI**
Session	**Contrôle (Juin 2010)**
Durée	**2 heures**
Documents	**Non autorisés**

EXAMEN

Exercice n°1 (4 points)

Écrire une fonction **NbCarMajus** qui permet de compter le nombre des lettres en majuscules.

Ecrire le programme principal qui permet de saisir une chaine CH, et d'afficher le résultat à l'écran.

Exemple:

CH= "Faculte des **S**ciences **J**uridiques **E**conomiques et de **G**estion de **J**endouba**"**

N=5

Exercice n°2 (6 points)

Écrire un programme qui permet de remplir un tableau contenant des notes (avec 0≤ note≤20), dont la dimension du tableau est 5≤n≤30. On suppose que les notes ont le même coefficient.

De calculer la moyenne et d'afficher le résultat à l'écran.

Exemple:

14	12.5	6.25	13.75	11.5
0	1	2	3	4

La moyenne est : 11.6

Exercice n°3 (10 points)

Écrire un programme qui permet de saisir deux chaines CH1 et CH2, d'extraire de ces dernières que les chiffres, d'effectuer la somme des deux nombres et enfin afficher le résultat à l'écran.

Exemple:

CH1= "N5jrt23fe6K" ➔ **nb1 = 5236**

CH2= "p3Y2" ➔ **nb2 = 32**

La somme des deux nombres est : 5268

NB : vous devez employer la fonction prédéfinie **atoi** de la bibliothèque (stdlib.h) qui permet de convertir une chaine en entier.

- int **atoi**(char *chaîne) convertit la chaîne en entier

exemple:
```
ch="12" ;
nb = atoi(ch) ; /* nb=12 */
```

Bonne Chance