TP 1

EXERCICE 1

Ecrire un programme qui permet d'afficher le mot «Bonjour » sur la console.

Réponse:

```
#include <stdio.h>
void main()
{
	printf("Bonjour\n");
	system("PAUSE"); // Pour éviter la fermeture automatique de la console avec devC++
			 // Facultatif avec Visual C++
}
```

EXERCICE 2

Ecrire un programme qui permet de lire à partir du clavier deux entiers A et B et d'afficher par la suite sur la console leur somme C.

Remarque: Avant la saisie des valeurs de A et B, écrire sur la console un message demandant à l'opérateur d'entrer Ces valeurs.

Réponse:

```
#include <stdio.h>
void main()
{
	int A,B,C;
	printf("Entrer A : "); scanf("%d",&A);
	printf("Entrer B : "); scanf("%d",&B);
	C=A+B;
	printf("La valeur de C est %d\n", C);
}
```

Remarque: Si vous utiliser devC++, ajouter avant la fin du programme l'instruction **`system("PAUSE");`** Ceci est valable pour le reste des exercices.

EXERCICE 3

Reprendre les exemples préliminaires traités en cours (chapitre 1) et les tester.

//Exemple 1 (utilisation et affichage de variables réelles)

```
#include <stdio.h>
void main()
{
    float L,l, Surface;
    l=5; L=25; Surface=l*L;
    printf("La surface est = %.2f\n", Surface);
}
```

//Exemple 2 (utilisation d'une constante)

```
#include <stdio.h>
#define TVA 18.6
void main()
{
    float HT,TTC;
    puts("Entrer le prix H.T. :");
    scanf("%f",&HT);
    TTC=HT*(1+(TVA/100));
    printf("prix T.T.C. %f\n",TTC);
}
```

TP 2 et 3

EXERCICE 4

Ecrire un programme qui permet de calculer la somme S de deux entiers A et B. Le même programme est capable de calculer (d=A-B), (p=A*B), (D=A/B) et le reste R de la division euclidienne de A sur B.

EXERCICE 5

Ecrire un programme permettant de permuter les valeurs de deux variables réelles A et B (avec et sans introduction de variables intermédiaires). Réaliser un programme équivalent pour deux caractères.

EXERCICE 6

Reprendre les exemples préliminaires traités en cours (chapitre 2) et les tester.

```
//Exemple 3
//Taille occupée par un type de données sur la machine avec la commande sizeof()

#include <stdio.h>
void main()
{
       printf("Taille de int = %d octets\n", sizeof(int));
       printf("Taille de short = %d octets\n", sizeof(short));
       printf("Taille de long = %d octets\n", sizeof(long));
       printf("Taille de unsigned short = %d octets\n", sizeof(short));
       printf("Taille de float = %d octets\n", sizeof(float));
       printf("Taille de double = %d octets\n", sizeof(double));
       printf("Taille de char = %d octets\n", sizeof(char));
       printf("Taille de long double = %d octets\n", sizeof(long double));
}
```

```
//Exemple 4
// Valeurs acceptées pour un type de données (interpréter le code)

#include <stdio.h>
void main()
{
       unsigned short x1=40000; // sachant que unsigned short : de 0 à 65535
       short x2=40000; // sachant que short : de -32768 à 32767
       printf("x1 = %d et x2 = %d",x1,x2);
       printf("\nx2=%d",40000-65536);
       printf("\nx1 = %f \n",x1);
}
```

```
//Exemple 5
// Valeurs acceptées pour un type de données (interpréter le code)

#include <stdio.h>
void main()
{
      float y; y = 9.122;
      printf("\ny = %d",y);
      printf("\ny = %f",y);
      printf("\ny = %.1f\n",y);
}
```

```
//Exemple 6
// Valeurs acceptées pour un type de données (interpréter le code)

#include <stdio.h>
void main()
{
      short x=32767; // sachant que short : de -32768 à 32767
      printf("\nx = %d\n",x2);
      x= x+1; printf("valeur suivante : %d\n",x);
      x= x+1; printf("valeur suivante : %d\n",x);
}
```

TP 4

EXERCICE 7

On considère x et y deux variables réelles tel que y est initialisée à la valeur 9,25.
D'un autre coté, a et b sont deux variables entières tel que a est initialisée à la valeur 7. Prévoir le résultat de chacune des syntaxes suivantes :

```
a) x = (float) a;
b) b = (int) y;
c) b= a/2;
d) x= a/2;
e) x= a/(float)2;
```

Ecrire un programme qui permet de vérifier les résultats. Ce programme explique le mode de conversion entre variables de types entiers et réelles.

Réponse:

```
#include <stdio.h>
void main()
{
	float x, y=9.25;
	int a=7, b;

	printf("La syntaxe a) donne :\n");
	x = (float) a;
	printf("a= %d et x= %.2f\n",a,x); system("PAUSE");

	printf("\nLa syntaxe b) donne :\n");
	b = (int) y;
	printf("y= %.2f et b= %d",y,b);
	printf("\nb est la partie entiere de y\n"); system("PAUSE");

	printf("\nLa syntaxe c) donne :\n");
	b= a/2;
	printf("b = %d\n",b); system("PAUSE");

	printf("\nLa syntaxe d) donne :\n");
	x= a/2;
	printf("x = %.2f\n",x); system("PAUSE");

	printf("\nLa syntaxe e) donne :\n");
	x= a/(float)2; printf("x = %.2f\n",x);
}
```

EXERCICE 8

a) Soit x un entier et car un caractère. Que donne l'instruction **`x = car;`** ?
Ecrire un programme qui permet de lire un caractère car et d'afficher son code ASCII. Remarquer que la conversion de char vers int est automatique (pas besoin de mettre **`x = (int)car;`**) et interpréter le résultat donné par l'instruction

```
printf("\nLe code ASCII du caractere %c est %d\n",car,car);
```

b) C'est quoi les codes ASCII des lettres a et A ?
c) Que donne l'instruction **`car='a'+2 ;`** ?
d) Soit y un entier. Que donne l'instruction **`car = y;`**? Tester la pour y = 100, pour y = 356 et pour y=-156. Sachant que les codes ASCII varient entre -128 et 127, remarquer que la conversion de int vers char se fait avec un modulo pré de 256 (vous pouvez utiliser l'instruction

```
printf("\nLe code ASCII du caractere %c est %d\n",y,(char)y);
```

ou encore **`car = y; printf("\nLe code ASCII du caractere %c est %d\n",car,car);`**)

TP 5

EXERCICE 9

Cet exercice donne un exemple sur les fonctions d'entrée/sortie pour un caractère.

a) En utilisant les fonctions getchar et putchar, écrire un programme qui permet de saisir et d'afficher un caractère ca.

b) Dans le même programme, utiliser les fonctions printf et scanf pour saisir et afficher un deuxième caractère appelé car.

Remarque:

En cas de lecture multiple de données de type caractère, la touche entrée (appuyée lors de la dernière saisie) est considérée aussi comme étant un caractère et ce dernier est stocké dans le **buffer** (zone de mémoire tampon intermédiaire entre clavier ou console et le programme). Ceci faussera la prochaine lecture puisque la variable correspondante prendra la première valeur disponible dans le buffer.
Pour cela, avant chaque saisie (avec scanf par exemple), il est conseillé de vider le buffer grâce à l'instruction :

```
fflush(stdin);
```

EXERCICE 10

Le programme suivant donne un exemple sur les fonctions d'entrée/sortie pour les chaînes de caractères. Tester ce programme:

```
#include <stdio.h>
void main()
{

       char chaine[10]; // une chaine prévue de contenir au plus 10 caracteres

       printf("Entrer une chaine de caractere : ");
       gets(chaine); printf("chaine = "); puts(chaine);

       printf("\nEntrer une nouvelle valeur de la chaine : ");
       scanf("%s",&chaine); printf("chaine = %s \n", chaine);
}
```

TP 6

EXERCICE 11

Ecrire un programme qui calcule et qui affiche le maximum et le minimum de quatre variables réelles A, B, C et D données par l'utilisateur.

EXERCICE 12 (facultatif)

Soit A une constante égale à 2 (on utilise ici : `#define A 2` dans la zone du préprocesseur).
Ecrire un programme qui teste si un entier saisi à partir du clavier est divisible par A ou non.
Vous pouvez réaliser les mêmes tests pour A=3, A=5 et A= 7.

EXERCICE 13

Ecrire un programme qui permet de saisir un caractère « car » et d'indiquer s'il s'agit d'une lettre de l'alphabet ou non. Dans le premier cas, le programme indiquera s'il s'agit d'une voyelle ou d'une consonne.

Indications :

code ASCII de 'A' = 65 code ASCII de 'Z' = 90
code ASCII de 'a' = 97 code ASCII de 'z' = 122
Voyelles : a, e, i, o, u, y, A, E, I, O, U, Y.
Vous pouvez utiliser l'instruction (switch …case).

EXERCICE 14

Ecrire un programme qui permet de lire votre nom et prénom (séparés par un espace), et qui assure que toutes les lettres soient en majuscules (vous remplacez les lettres minuscules par des lettres majuscules).
En cas de présence de caractères qui n'appartiennent pas à l'alphabet (é, è, ü… 1, 2, …, $, #,…), le programme retourne un message d'erreur.

Indications :

- Vous déclarez une chaîne de caractères de taille 20 : (`char chaine[20];`)
- Remarquez la différence entre (`scanf("%s",&chaine);`) et (`gets(chaine);`) au niveau de la lecture d'une chaîne qui contient un espace.
- Vous pouvez ajouter l'instruction (`fflush(stdin);`) qui permet de vider le buffer en cas de lectures multiples.
- Pour passer de « a » à « A », il faut retrancher 32 du code ASCII (65= 97-32).
- chaine[i-1] est Le ième caractère de la chaine.
- Une chaîne de caractère se termine par le caractère '\0'.

EXERCICE 15 (facultatif)

Ecrire un programme qui permet de lire un texte à partir du clavier (un mot ou une phrase), puis de remplacer les voyelles par des tirets (-) et les espaces par des étoiles (*).

TP7

EXERCICE 16

Ecrire un programme qui permet d'afficher la table étendue des caractères ASCII (256 caractères, codes ASCII allant de -128 à 127) selon le format suivant :

CARACTERE Ç CODE ASCII -128
CARACTERE ü CODE ASCII -127
.
.
.
CARACTERE ⌂ CODE ASCII 127

Remarque :
La table standard des codes ASCII concerne les codes de 0 à 127 seulement. Elle nécessite 7 bits (contrairement à la table étendue qui nécessite 8 bits) et contient peu de caractères spéciaux (ni de caractères accentués, ni de caractères spécifiques à une langue autre que l'anglais). Les caractères de 0 à 31 sont appelés des caractères de contrôle car ils permettent de faire des actions telles que le retour à la ligne, la tabulation, etc.

EXERCICE 17 (facultatif)

Ecrire un programme qui permet de saisir un entier positif et d'indiquer sa longueur, qui est le nombre de chiffres (de décimales) qu'il contient.
Exemple : la longueur de 322 est 3, la longueur de 12 est 2 et la longueur de 50916 est 5.

EXERCICE 18

Soit T un tableau de 10 entiers. On rappelle que la déclaration du tableau T est réalisée par la syntaxe « `int T[10];` », et que le ième élément du tableau est « `T[i-1]`». Le tableau T contient donc les éléments suivants : `T[0],T[1],T[2],…,T[9].`

Ecrire un programme qui permet de saisir un entier positif et de stocker les chiffres qui le compose dans le tableau T.

Exemple : pour un entier saisi égal à 23191, on aura :
T[0]=1, T[1]=9, T[2]=1, T[3]=3, T[4]=2, T[5]=0, T[6]=0, T[7]=0, T[8]=0, T[9]=0.

EXERCICE 19

Ecrire un programme qui permet de lire une valeur entière comprise entre 5 et 25. Ce programme utilisera une structure «while» qui impose de refaire la saisie tant que la valeur saisie est en dehors de l'intervalle [5, 25].

TP8

EXERCICE 20

Soit T un tableau de N entiers. N est défini en tant que constante égale à 10.
Ecrire un programme qui lit les N éléments du tableau en vérifiant que les valeurs saisies sont comprises entre 10 et 20. Ce programme affichera les N valeurs du tableau, la somme de ces valeurs, la valeur maximale et la valeur minimale dans ce tableau.

EXERCICE 21

Ecrire un programme qui résout une équation du second degré. Il doit déterminer si l'équation admet une solution réelle et déterminer sa valeur si elle existe.

Indications :
- Lire les paramètres a, b et c de l'équation :
(E) : $a\,X^2 + b\,X + c = 0$
- Calculer le discriminant $\Delta = b^2 - 4\,a\,c$
- Inclure la bibliothèque <math.h> qui contient la fonction racine carrée sqrt(). Exemple :
(`printf("%f\n",sqrt(4));`) permet d'afficher le résultat (`2.000000`)
- Il existe trois résultats possibles en fonction de la valeur de Δ :
  - aucune racine : pour $\Delta < 0$
  - une racine double $\frac{-b}{2a}$: pour $\Delta = 0$
  - les deux racines $\frac{-b}{2a} \pm \frac{\sqrt{\Delta}}{2a}$: pour $\Delta > 0$

EXERCICE 22

Ecrire un programme qui permet de calculer et d'afficher la valeur de la série S_N :

$$S_N = \frac{1^2}{2!} + \frac{2^2}{3!} + \frac{3^2}{4!} + \ldots + \frac{N^2}{(N+1)!}, \quad \text{avec } k! = 1 \times 2 \times \ldots \times k$$

EXERCICE 23 (facultatif)

Ecrire un programme qui calcule la moyenne et la variance de N réels saisis au clavier (sans utilisation de tableau).

TP9

EXERCICE 24

Ecrire des fonctions qui calculent respectivement la somme, le produit, le minimum, le maximum, la moyenne, de deux entiers A et B. Utiliser ces fonctions dans un programme.

EXERCICE 25

Ecrire une fonction qui teste si un entier est premier. Afficher dans un programme tous les entiers premiers inférieurs à 50.

Indications :
Un nombre premier A est un entier naturel, qui se divise seulement par 1 et lui-même.
Si aucun entier entre 2 et A/2 ne divise A alors A est premier.
Si 2 ne divise pas A et aucun entier entre 3 et A/3 ne divise A alors A est premier…
Si 2, 3,…, k-1 ne divisent pas A et aucun entier entre k et A/k ne divise A alors A est premier.

EXERCICE 26 (facultatif)

Ecrire une procédure qui vérifie si un entier A est symétrique, c'est-à-dire, la séquence de ses chiffres est symétrique. Exemple : si A=12321 ou A=12344321 alors la procédure permet d'afficher le texte « Entier symétrique ». Pour A=1237231, la procédure affiche le texte « Entier asymétrique». Utiliser la procédure dans un programme.

EXERCICE 27 (facultatif)

Ecrire une fonction qui permet de convertir un nombre de la base binaire à la base décimale. Ecrire une autre fonction qui effectue l'opération inverse.

EXERCICE 28 (facultatif)

Ecrire des fonctions permettant de calculer la moyenne, la variance, le minimum et le maximum d'un tableau. Pour les valeurs minimale et maximale, on donne aussi les positions.

TP10

EXERCICE 29

Ecrire une fonction qui permet de calculer l'occurrence d'un entier A dans un tableau d'entiers T de taille N.

EXERCICE 30

Ecrire une procédure qui permet de réaliser une décomposition élémentaire d'un entier A en ses diviseurs premiers.
Exemple :
24 est décomposé comme suit : 24 = 2 × 2 × 2 × 3, la procédure affiche : 1, 2, 2, 2, 3
42 est décomposé comme suit : 42 = 2 × 3 × 7, la procédure affiche : 1, 2, 3, 7
17 est décomposé comme suit : 17 = 1 × 17, la procédure affiche : 1, 17

EXERCICE 31

Ecrire une procédure qui permet de permuter deux entiers A et B. Utiliser cette procédure dans un programme.

EXERCICE 32 (facultatif)

Ecrire une procédure qui transforme un entier A d'une manière symétrique.
C'est-à-dire, si A=3524 alors A devient = 4253. Utiliser cette procédure dans un programme.

EXERCICE 33 (facultatif)

Ecrire une procédure qui permet d'inverser une chaîne de caractère ch.

EXERCICE 34 (facultatif : jeu vache-taureau)

Coder le fameux jeu vache-taureau dont l'objectif est de trouver le nombre (à quatre chiffres) générés aléatoirement par l'ordinateur. Ce nombre doit contenir des chiffres différents.
L'utilisateur a 5 ou 6 essais pour trouver le nombre. A chaque essai, le programme renvoie autant de « taureaux » pour des chiffres qui existent et à la bonne position, et autant de « vaches » pour des chiffres qui existent à des mauvaises positions.

Exemple : Nombre recherché égal à 4157 : pour l'essai 1234 le programme retourne 2 vaches et pour l'essai 4567, le programme retourne 1 vache et 2 taureaux.
Indications :

```
int random(int amax){
        int a;
        a=((int) (rand()%1000)*(amax-1)/1000);
        return(a);
}
```

La fonction précédente permet de demander à l'ordinateur de générer aléatoirement un entier entre 0 et amax. L'utilisation de la fonction rand() nécessite les bibliothèques <stdlib.h> et <time.h>.