Université Aboubekr BELKAID
كلية العلوم – تيجاني هدام
Faculté des Sciences
قسم الإعلام الآلي
Département d'informatique

Module : Compilation **Année universitaire 2020-2021**
Classe : L3- Informatique
Réalisé par : Mr MERZOUG Mohamed
Mr ETCHIALI Abdelhak

TP N° 1

Rappel sur la programmation en C

1.1. Introduction

Le but de ce TP est de vous rappeler quelques notions sur la programmation en langage C. Pour Ecrire un Programme C vous pouvez utiliser l'éditeur l'IDE NetBeans.

Mon Premier Programme C

Lancer DevC ++ et créer un nouveau fichier (File|New|Source File…) , enregistrer ce fichier sous le nom Tp 1.c (File|Save)

Ecrire le programme suivant :

```
#include <stdio.h>
    int main(void){
        int nombre;
        printf ("Entrer Un Nombre Entier SVP :");
        scanf( "%d",&nombre);
        printf("Vous Avez Tapez le nombre entier : % d",nombre);
        getch ();
        return  0;
    }
```

Exercice 01 :

Ecrire un Programme C permettant de simuler une calculatrice. Le programme doit demander à l'utilisateur deux nombres entiers ainsi que la nature d e l'opération (A: Addition, S: Soustraction, M: Multiplication, D : Division), et lui retourner le résultat de cette opération.

1.2 Les pointeurs

Un pointeur est une variable, il est destiné à contenir l'adresse d'une variable. Pour différencier un pointeur d'une variable ordinaire, on fait précéder son nom du signe '*' lors de sa déclaration.

La déclaration **Type *P** déclare un pointeur P qui peut recevoir des a dresses de variables du type **Type**.

L'opérateur **&** désigne **l'adresse** d'une variable : **&Prix** fournit **l'adresse** d e la variable **Prix**.

L'opérateur ***** désigne le **contenu** d'une adresse : ***adr** désigne le contenu de l' adresse référencée par le pointeur **adr**.

Exercice 02 : Ecrire une fonction echange permettant d'échanger les valeurs de deux nombres entiers a et b, appeler cette fonction dans le programme principal pour échanger es valeurs de deux entiers saisis au clavier.

1.3 Pointeurs et table aux :

Le nom d'un tableau est considéré comme un pointeur sur son premier élément.

Les écritures suivantes sont équivalentes :

&tab[0] et tab ; tab[0] et *tab ; tab[i] et *(tab+1)

Exercice 03 : Ecrire un programme C permettant de rechercher la plus grande valeur dans un tableau de 10 nombres réels ;

1.4 Les Chaines de Caractères :

En C, on représente les chaînes de caractères par un tableau de caractères, dont le dernier est un caractère de code nul (\0). Ainsi une chaîne composée de n éléments sera en fait un tableau de n+1 éléments de type **char**. Une constante caractère est identifiée par les guillemets **"** (double quotte).

Exemple :

```
char message[5]= "mail";
puts (message) ;
```

message est un tableau de **5 caractères** (\0 compris).

On peut également initialiser un pointeur avec une chaîne de caractères : char *ptrch="mail";

A- Fonctions d'entrées/sorties pour les chaînes (stdio.h) :

- **gets(char*)** lecture d'une chaîne sur stdin.
- **puts(char*)** affiche, sur stdout, la chaîne de caractères puis positionne le curseur en début de ligne suivante.
- Dans **<conio.h>**, on trouve les fonctions de base de gestion de la console :
- **putch(char)** : Affiche sur l'écran (stdout) le caractère fourni en argument, cette fonction rend le caractère affiché ou EOF en cas d'erreur.
- **getch(void)** : Attend le prochain appui sur le clavier, et rend le caractère qui a été saisi. L'appui sur une touche se fait sans écho.
- **getche(void)** : Idem getch mais avec écho.
- **getchar(void)** fonctionne comme getche, mais utilise le même tampon que scanf.

B- Fonctions utiles à la manipulation de chaînes (string.h) :

- **int strlen(chaîne)** donne la longueur de la chaîne (\0 non compris)
- **char *strcpy(char *dest,char *src)** recopie la source dans la destination, rend un pointeur sur la destination.

- **char *strncpy(char *destination,char *source,int max)** idem que strcpy mais s'arrête au \0 ou au max caractères lus ;
- **char *strcat(char *dest,char *src)** concatène la source à la suite de la destination, rend un pointeur sur la destination
- **char *strncat(char *destination,char *source,int longmax)** idem que strcat mais s'arrête au \0 ou au max caractères lus ;
- **int strcmp(char *str1,char*str2)** rend 0 si str1==str2, <0 si str1<str2, >0 si str1>str2. Idem strncmp

C- Fonctions de conversions entre scalaires et chaînes (stdlib.h) :

- **int atoi(char *s)** traduit la chaîne en entier (s'arrête au premier caractère impossible, 0 si erreur dès le premier caractère) (voir aussi **atol** et **atof**)

D- Fonctions limitées au caractères (ctype.h) :

- **int isdigit(int c)** rend un entier non nul si c'est un chiffre ('0' à '9'), 0 sinon
- de même : **isalpha (int c)** (c de A à Z et a à z, mais pas les accents),
- **isalnum (int c) (isalpha|isdigit), islower (minuscule), isupper, isspace (blanc, tab, return...), isxdigit (0 à 9,A à F,a à f).**..
- **int toupper(int c)** rend A à Z si c est a à z, rend c sinon. (voir aussi **tolower(int c)**) ;

E- Fonctions de la gestion dynamique de mémoire (alloc.h)

- **void *malloc(int taille) :** réserve une zone mémoire contiguë de taille octets, et retourne un pointeur sur le début du bloc réservé. Retourne le pointeur NULL en cas d'erreur (en général car pas assez de mémoire).
- **void *calloc(int nb, int taille)** : équivalent à malloc(nb* taille).
- **void free(void *pointeur)** libère la place réservée auparavant par malloc ou calloc. Pointeur est l'adresse retournée lors de l'allocation.

Exercice 04 :

Ecrire une fonction **afficheChaine(char* chaine)** qui prend en paramètre une chaîne de caractères et l'affiche en insérant un espace entre chacun de ses caractères.

Tester la fonction précédente dans le programme principal.

Exercice 05 :

Ecrire une fonction **voyelle (chaine)** qui permet de retourner le nombre de voyelles dans la chaine **chaine**.

Tester la fonction précédente dans le programme principal.