TP 1 :
Les pointeurs

Objectifs

Manipuler des variables à l'aide des pointeurs.

Rappel :

Un pointeur est une variable qui désigne un emplacement mémoire (une adresse) occupée par une donnée dont le type est défini lors de la déclaration. Lorsqu'on définit un pointeur, on précise également le type de donnée que contient l'emplacement pointé.

Une variable pointeur est déclarée en utilisant le caractère « * » devant le nom de la variable. Cette notation permet de distinguer les variables réelles des pointeurs.

Précisons également qu'un pointeur « pointe » sur une donnée en mémoire et que cette donnée peut être codée sur plusieurs emplacements mémoire. Dans ce cas, le pointeur « pointe » sur le premier emplacement de la donnée.

Pour déclarer un pointeur, il suffit d'utiliser la syntaxe suivante :

int *i ; // « * » dans la déclaration indique qu'il s'agit d'un pointeur

Le pointeur est alors utilisable pour affecter le contenu de l'adresse correspondante :

***i = 134; // « * » dans l"affectation défini le contenu de l"adresse pointée**

Il est possible de manipuler les pointeurs de telle manière qu'ils « pointent» sur un emplacement déjà occupé par une variable :

char c ="a" ;

char *p1 = &c ; // « & » signifie adresse de la variable c. p1 « pointe » sur c

Exercice 1 :

Soit le programme suivant :

```
#include <stdio.h>
void afficher( char *p) {
  while ( *p != '\0')
{
// A COMPLETER
}
}
int main()
{
```

```
char *chaine = "bonjour    les   amis";
afficher(chaine);
 return 0;
}
```

Complétez la boucle « while » de la fonction « afficher » pour qu'elle affiche tous les caractères de la chaîne SANS les espaces (Rappel : le caractère '\0' est le caractère de fin de chaine en C).

Exercice 2 :

Soit la fonction « main » suivante appelant une fonction « calcul » qui réalise deux opérations : la somme et la différence de 2 entiers.

```
int main() {
 int a=12, b=45;  int somme, difference;
 calcul(a,b, &somme, &difference);
 printf("a+b=%d , a-b=%d\n", somme, difference);
return 0; }
```

Ecrire la fonction « calcul » pour qu'elle puisse réaliser le travail demandé.

Exercice 3 :

Le but est de copier une chaîne de caractère dans une autre en inversant la casse (Passer de minuscule à majuscule et vice-versa). La fonction « scase », donc voici la déclaration, assura cette tâche :

```
void scase( char *p1, char *p2);
```

Voici le programme principal chargé d'appeler "scase" :

```
int main() {
char ch1[] = "abcdefghi jKLMnopqrstuvwxyz";
char ch2[26];
scase(ch1, ch2);
printf("ch1 = %s , ch2= %s \n", ch1, ch2);
return 0; }
```

Complétez la fonction « scase » en utilisant les 2 pointeurs passés en paramètres de la fonction. N'oubliez pas que, d'après la table ASCII, les majuscules et les minuscules sont séparés de 32 caractères :

char c1 = 'A' ;

char c1bis = 'a' ;

char c2 = c1 + 32 ; // c2 contient 'a'

Exercice 4 :

La fonction strlen renvoie la longueur d'une chaîne de caractères. Un débutant en langage C écrit cette fonction de la façon suivante:

int strlen(char *s) { int i; int len = 0; for (i=0; s[i]!='\0'; i++) { len++; } return len; }

En fait comme s est un pointeur, on peut s'en servir pour parcourir la chaîne plutôt que d'utiliser une variable supplémentaire i. Il suffit pour cela de l'incrémenter pour le faire pointer successivement vers les différents caractères de la chaîne.

Réécrire ainsi la fonction strlen en utilisant une seule variable locale et la tester à travers une fonction main.

Exercice 5 :

Ecrire un programme C qui lit une chaîne de caractères et affiche cette chaîne à partir de la première occurrence d'un caractère entré par l'utilisateur. En utilisant pour ceci la fonction *strchr* et un pointeur pour le parcours de la chaîne

Exercice 6 :

Ecrire un programme C qui remplit un tableau d'entiers et calcule la somme de ses éléments en utilisant un pointeur pour son parcours.

Exercice 7 :

Ecrire un programme C qui définit une structure permettant de stocker le nom, le prénom et l'âge d'une personne. Lit ensuite ces informations pour deux personnes et affiche le nom complet de la moins âgée d'entre elles en utilisant une seule fonction *printf* pour l'affichage du résultat.

Exercice 8 :

Ecrire un programme C qui vérifie si une chaîne de caractères est palindrome en utilisant deux pointeurs pour son parcours.

Exercice 9 :

Ecrire un programme C qui réserve l'espace mémoire à un tableau d'entiers de taille entrée par l'utilisateur, le lit puis l'affiche.

Exercice 10 :

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.

Exercice 11 :

Ecrire un programme C qui réserve l'espace mémoire à un tableau de caractères sous forme d'un triangle droit, le remplit par des étoiles (*) puis l'affiche.