Atelier 1 : Les pointeurs en C

Compétences à atteindre

C1. Savoir déclarer un pointeur et faire la différence entre une variable classique et un pointeur

C2. Savoir manipuler les pointeurs (arithmétique des pointeurs)

C3. Allouer dynamiquement de la mémoire avec la fonction malloc

C4. Libérer de la mémoire allouée à un pointeur avec la fonction free

C5. Gérer un tableau et les chaines de caractères en utilisant les pointeurs

C6. Gérer une matrice en utilisant les pointeurs

C7. Utiliser les pointeurs dans les fonctions (définition et appel)

Exercice n°1

1. Saisir le programme suivant :

```
int main()
{
int i=100;
printf ("La variable i a comme adresse %d et contient la valeur
suivante %d\n",&i,i);
int *pi=NULL;
pi=&i;
printf("le pointeur pi contient la valeur %d et pointe vers la
valeur %d\n",pi,*pi);
*pi=200;
printf("le pointeur pi contient la valeur %d et pointe vers la
valeur %d\n",pi,*pi);
printf ("La variable i a comme adresse %d et contient la valeur
suivante %d\n",&i,i);


}
```

2. Exécuter le programme et interpréter le rôle d'un pointeur

3. On veut changer le programme pour n'utiliser que la variable pi, compléter le programme suivant


```
#include <stdio.h>

void main () {

int *pi=NULL;

……

*pi=200;

printf ("pi pointe vers la case contenant %d\n",……..);

}
```

Exercice n°2

1. Saisir et exécuter le programme suivant :

```
#include <stdio.h>

int main() {
            /* declaration d'un tableau d'entiers à 3 elements*/
            int a[3]={0,2,5} ;
            int dist ;
            /* declaration d'un pointeur entier */
            int *ptr_int;

            /*initialisation de ptr_int avec l'adresse du
premier element de tableau */
            ptr_int=&a[0];

            printf("la valeur de a[0] = %d\n", a[0]) ;
            printf("l'adresse de a[0] = %u (%x)\n", &a[0],&a[0])
;
            printf("la valeur de ptr_int= %u (%x)\n", ptr_int,
ptr_int) ;
            printf("la valeur pointe par ptr_int =
%d\n",*ptr_int);

            *ptr_int=*ptr_int+2; //incrementation du contenu
pointe par ptr_int
            printf("\n apres incrementation de *ptr\n\n");

            printf("la valeur de a[0] = %d\n", *ptr_int) ;

            ptr_int++; //incrémentation de l'adresse mémoire
contenu dans ptr_int
            printf("l'adresse de a[1] = %d (%x) \n",
&a[1],&a[1]) ;
            printf("la valeur de ptr_int apres incrementation
(ptr_int++) %u (%x)\n",ptr_int, ptr_int);

            dist=ptr_int-&a[0]; // vaut le nombre d'entiers
représentables entre l'adresse ptr_int et l'adresse du premier
élément de a
            printf("le nombre d'entiers representables entre
ptr_int (%x) et a[0] (%x) = %d\n", ptr_int,&a[0],dist);

            dist=(char*)ptr_int-(char*)&a[0];// vaut le nombre
de chars representables entre l.adresse ptr_int et l.adresse du
premier élément de a
          printf("le nombre de caracteres representables entre
ptr_int (%x) et a[0] (%x) = %d\n", ptr_int,&a[0],dist);
}
```

2. Soit les instructions suivantes :

```
int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
int *P;
P = A;
```

Donner les résultats de ces expressions, comparer et interpréter

Expr1	Expr2	Interprétation
*P=	A[0]=	
*P+2=	*(P+2)=	
A =	A+3 =	
&A[7]-P =	A+7-P=	

Exercice n°3

1. Ecrire la fonction permuter(int a, int b) qui permet de permuter le contenu de deux variables de type entier.
2. Ecrire la fonction principale main qui permet de saisir deux entiers x et y et de permuter leur contenu. Afficher le contenu de x et de y après permutation.

Exercice n°4

Ecrire un programme qui lit deux tableaux A et B et leurs dimensions N et M au clavier et qui ajoute les éléments de B à la fin de A en utilisant le formalisme pointeur.

Exercice n°5

Écrire un programme qui lit un entier X et un tableau A de type int au clavier et élimine toutes les occurrences de X dans A en tassant les éléments restants. Le programme utilisera des pointeurs pour parcourir le tableau.

Exercice n°6

1. Écrire une fonction char * recherche_char(char* s, char c) qui renvoie un pointeur vers la première occurrence du caractère c (le caractère passé en argument, pas le caractère 'c') dans la chaine s. Si ce caractère n'apparait pas dans la chaine, la fonction devra renvoyer NULL.
2. À l'aide de recherche_char, écrire une fonction int compte_char(char *s, char c) qui compte le nombre d'occurrences de c dans s.
3. Ecrire un programme pour tester les deux fonctions

Exercice n°7

1. Ecrire la fonction AJOUTE_CH à deux paramètres CH1 et CH2 qui copie la chaîne de caractères CH2 à la fin de la chaîne CH1 sans utiliser de variable d'aide et sans utiliser strcpy.
2. Tester la fonction

Exercice n°8

Ecrire un programme qui lit une matrice A de dimensions N et M au clavier et affiche la transposée de en utilisant le formalisme pointeur.