

TP 2 : Tableaux, pointeurs, références

1 (Pointeurs) *Quels seront les effets des exécutions des programmes suivants ?*

```
1 #include <iostream>
2 #include <cmath>
   using namespace std;
4
   int main(void){
6   double *a;//Definit a comme un pointeur vers un double
   double c; //Definit l'emplacement memoire pour un double, appele c
8   c=5.0; //Attribue au double c la valeur 5.0
   a=&c; //a devient l'adresse de c
10  cout<<*a;
   return 0;}
```

```
1 #include <iostream>
2 #include <cmath>
   using namespace std;
4
   int main(void){
6       double *a;
       double *b;
8       double c=1.0,d=2.0,f,g;
       double *e;
10      double *h;
       a=&c;
12      b=&d;
       f=*a+*b;
14      e=&f;
       g=(*a+*b+*e)/2.0;
16      h=&g;
       cout<<*h*2.0;
18      return 0;}
```

2 (Pointeurs versus références 1) a) *Dans le programme suivant, remplacer les cases **REEMPLIR** par le contenu adéquat.* b) *Réécrire le programme de façon à remplacer la première fonction par une fonction de prototype :*

`void minmax(int i, int j, int& min, int& max)`

```
1 #include<iostream>
2 using namespace std;
4 void minmax(int i, int j, int* min, int* max){
   if(i<j) { *min=i; *max=j; }
6   else { *min=j; *max=i; }}
8 int main(){
   int a, b, w, x;
10  cout << "Tapez la valeur de a : "; cin >> a;
   cout << "Tapez la valeur de b : "; cin >> b;
12
   minmax(a, b, REEMPLIR, REEMPLIR);
14  cout << "Le plus petit vaut : " << w << endl;
   cout << "Le plus grand vaut : " << x << endl;
16  return 0; }
```

- 3 (Pointeurs versus références 2)** a) *Le programme suivant va-t-il fonctionner ? On choisit alors de ne laisser qu'une des trois définitions de ajouter. Pour laquelle/lesquelles des définitions le programme fonctionnera-t-il ? Qu'affichera-t-il alors sur la console ?*
 b) *On remplace, dans la fonction main, ajouter(a,b,c); par ajouter(a,b,&c);. Répondre à nouveau aux questions du a).*

```

#include<iostream>
2 using namespace std;

4 void ajouter (int a, int b, int c) { c = a+b; }
void ajouter (int a, int b, int * c) { *c = a+b; }
6 void ajouter (int a, int b, int & c) { c = a+b; }

8 main() {
    int a=1; int b=2; int c=0;
10    cout << "avant appel de ajouter" << endl;
    cout << "a = " << a << endl;
12    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
14    ajouter(a,b,c);
    cout << "apres appel de ajouter" << endl;
16    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
18    cout << "c = " << c << endl;}

```

- 4 (Référence vers une variable comme un alias de cette variable)** *Que va rendre le programme suivant ?*

```

#include <cmath>
2 #include <iostream>
using namespace std;

4 int& max( int& m, int& n ) {return ( m > n ? m : n );}

6 int main() {
8     int m = 2;
    int n = 3;
10    max(m,n) = 1;
    cout << "m = " << m << ", n = " << n << endl;
12    return 0;}

```

- 5** *Les programmes suivants vont-ils pouvoir être compilés ? Pourquoi ? Les modifier de façon à ce qu'ils fonctionnent.*

```

#include <iostream>
2 using namespace std;

4 int * pointeur_somme(int x, int y) {
    int z=x+y;
6    return &z;}

8 int main () {
    int x=1, y=2;
10    cout<<pointeur_somme(x, y)<<endl;
    return 0;}

```

```

#include <iostream>
2 using namespace std;

4 int * pointeur_somme(int x, int y) {
    int z=x+y;
6     return &z;}

8 int main () {
    int x=1, y=2;
10    cout<<*pointeur_somme(x, y)<<endl;
    return 0;}

```

6 (Tableaux 1) Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V . Le programme doit rechercher si V se trouve dans le tableau et doit supprimer la dernière occurrence de V en décalant d'une case vers la gauche les éléments suivants et en rajoutant un 0 à la fin du tableau. Le programme doit ensuite afficher le tableau final.

7 (Tableaux 2) Écrire un programme qui demande à l'utilisateur un entier $n \geq 2$, de taper n entiers qui seront stockés dans un tableau. Le programme doit ensuite afficher soit "le tableau est croissant (au sens large)", soit "le tableau est décroissant (au sens large)", soit "le tableau est constant", soit "le tableau est quelconque".

8 (Tableaux dynamiques et références) A rendre, par mail, en binôme, à florent.benaych@gmail.com. **Sujet du mail : C++IFMA, Prénom 1 NOM 1, Prénom 2 NOM 2.**

Écrire une fonction f de prototype

```
bool f(int t[], int n, int &v)
```

(t est un tableau d'entiers de taille n). f doit renvoyer par un booléen b indiquant s'il existe une valeur comprise (au sens large) entre 1 et 4 dans les n premières cases du tableau t . Si f renvoie `true`, v est égal à la valeur de la première case du tableau comprise entre 1 et 4. Tester cette fonction.

9 (Tableaux dynamiques et pointeurs) Écrire (et tester dans un programme) une fonction qui a comme paramètres un tableau d'entiers de taille quelconque, la taille du tableau, et 2 pointeurs vers des entiers `min` et `max`. La fonction doit renvoyer dans les entiers pointés par `min` et `max` respectivement les plus petits et les plus grands entiers du tableau.

10 (Tableaux multidimensionnels, incrémentation des pointeurs et précision de l'affichage) Écrire une fonction de prototype

```
void aff_ligne(double T[],int p, int prec_dig)
```

qui a pour effet d'afficher (en ligne) le tableau T de taille p avec `prec_dig` chiffres après la virgule, en utilisant le principe d'incrémention des pointeurs utilisé dans le programme suivant. S'en servir dans un programme qui demande à l'utilisateur un entier m , un nombre de chiffres après la virgule, et affiche (en deux dimensions) la matrice $[i/j]_{1 \leq i, j \leq m}$.

```

#include<iostream>
2 #include <iomanip> //bibliothèque permettant d'affiner la présentation
    // des résultats sur la console
4 #include <cstdlib> //bibliothèque permettant d'affiner la présentation
    // des résultats sur la console
6 #include <cmath>
using namespace std;

8
int main(){
10     double t[10];

12     for ( int i=0 ; i<5 ; i++) {t[i] = i*i + 1/3.;}
    for ( int j=5 ; j<10 ; j++) {t[j] = -j * j-1/3.;}

14
    double* x;
16     x=t; // pointeur vers le premier élément
    cout << fixed << setprecision(4); //"fixed" permet que les nombres
18     // apparaissent sur la même colonne qu'ils soient positifs ou négatifs,
    // "setprecision": nombre de chiffres après la virgule

20
    for ( int k=0 ; k<10 ; k++){
22         cout << setw(8) << *x << endl;//"setw(8) " nombre max de caractères
        // dans l'affichage
24         x++; // pointeur vers l'élément suivant
    }

26     cout << endl;
    return 0;}

```

Explications

- Dans cet exemple, t est un tableau statique de 10 entiers et x est un pointeur vers un entier.
- L'instruction x=t; permet de faire pointer x vers la première case du tableau t. Il est équivalent d'écrire x=t; que d'écrire x=&t[0];
- À l'intérieur d'une boucle for, on va afficher *x, c'est-à-dire l'entier pointé par t et à chaque étape, on écrit x++, ce qui incrémente la valeur de t de la taille d'un entier.
- x va donc pointer successivement t[0], t[1],... t[9]. On va donc afficher une à une toutes les cases du tableau.

11 (Tableaux de caractères) A rendre, par mail, en binôme, à florent.benaych@gmail.com. Sujet du mail : C++IFMA, Prénom 1 NOM 1, Prénom 2 NOM 2.

a) Écrire une fonction de prototype

```
void affichage_tableau_char(char t[], int n)
```

qui affiche le tableau de caractères t. La tester en écrivant un programme qui demande à l'utilisateur le nombre de lettres son prénom, son prénom et lui écrit un message personnalisé sur la console.

b) Améliorer cette fonction et ce programme de façon à ne pas avoir à demander le nombre de lettres, en vous inspirant du programme suivant (on supposera que les prénoms n'ont jamais plus de 100 lettres et qu'ils ne comportent pas d'espace). Le prototype de la fonction sera void affichage_tableau_char(char t[])

```

#include<iostream>
2 using namespace std;

4 int main() {
    char c[100];
6     cout << "Tapez une chaîne : ";
    cin >> c;

8
    char *p;
10     p=c; // pointeur vers le premier caractère
    while(*p!='\0'){
12         cout << *p << endl;
        p++;} // pointeur vers le caractère suivant

14     return 0;}

```

12 Simplifier le programme précédent en utilisant la bibliothèque `string` (donc en ajoutant `#include <string>` dans le préambule).

```
#include <iostream>
2 #include <string> // Obligatoire pour pouvoir utiliser les objets string
  using namespace std;
4
6 int main() {
  string maChaine; // Creation d'un objet "maChaine" de type string
8  cout<<"Tapez une chaine "<<endl;
  cin>>maChaine;
10  cout<<"Voici votre chaine : "<<maChaine<<endl;
  return 0;}
```