TP 1 - Quelques particularités du C++

Programmation Orientée Objet

1 Classe HelloWorld

1) Ecrivez un programme C++ affichant "Hello World !".
2) Ecrivez une classe HelloWorld dont une méthode affiche le message "Hello World !". Donnez le code du programme principal appelant la méthode de la classe HelloWorld.

```
#include<iostream>
#include<cstring>

using namespace std;
////////////////////////////////////////////////////////////////////////////////
// Definition de la classe HelloWorld
////////////////////////////////////////////////////////////////////////////////


class HelloWorld{
        public:
                void affiche() const;
};

void HelloWorld::affiche() const{
        cout << "Hello World" << endl;
}

////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////

int main()
{
        cout << "Hello World" << endl;

        HelloWorld hw;
        hw.affiche();

        return (EXIT_SUCCESS);
}
```

2 Opérateurs de chaîne

Ecrire un programme saisissant le nom et le prénom d'une personne respectivement dans une variable *strNom* et *strPrenom*. Puis le programme effectuera une concaténation des variables *strNom* et *strPrenom* dans une variable *sChaineConcatenee*. Vous utiliserez un tableau de caractères de taille fixe. Utilisez dans un premier temps les fonction *strcpy*, *strcat*. Puis, retester les même fonctionnalités à l'aide des opérateurs + et =.

```
#include<iostream>

#include<string>

using namespace std;

const int TAILLE = 25;

int main(){

char strNom [TAILLE];
char strPrenom [TAILLE];
char sChaineConcatenee [TAILLE];

cout << "strcpy, strcat" << endl;
cout << "**************" << endl;
```

```
cout << "Donnez le nom : ";
cin >> strNom;

cout << "Donnez le prenom : ";
cin >> strPrenom;

strcpy(sChaineConcatenee,strNom);
strcat(sChaineConcatenee," ");
strcat(sChaineConcatenee,strPrenom);

cout << "chaine concatenee : "<< sChaineConcatenee << endl;

cout << "   +   et  =  " << endl;
cout << "**************" << endl;

string stringNom, stringPrenom, stringChaineConcatenee;

stringNom = strNom;
stringPrenom = strPrenom;

stringChaineConcatenee=strPrenom;
stringChaineConcatenee+=" ";
stringChaineConcatenee+=strNom;

cout << "chaine concatenee : "<< stringChaineConcatenee << endl;
        return 0;
}
```

3 Echange

Ecrire une fonction échangeant deux entiers. Cette fonction aura successive les entêtes suivantes :

```
void echange (int a, int b);
void echange (int & a, int & b); void
echange (int * a, int *b);
```

Commentez.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// On ne peut pas avoir les trois fonctions echange comme données par l ennonce.
// en effet, le compilateur ne peut pas choisir lors de l appel echange (x,y) entre les
// fonctions void echange (int a, int b) et void echange (int & a, int & b)
////////////////////////////////////////////////////////////////////////////////////////////////////
#include<iostream>
using namespace std;

////////////////////////////////////////////////////////////////////////////////////////////////////
// affichage de 2 entiers
////////////////////////////////////////////////////////////////////////////////////////////////////
void affiche (int a, int b){
        cout << "( " << a << "," << b << " )" <<endl;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// Passage de parametre par recopie
////////////////////////////////////////////////////////////////////////////////////////////////////

void echange (int a, int b){
        int c;
        c = a;
        a = b;
        b = c;
}


////////////////////////////////////////////////////////////////////////////////////////////////////
// Passage de parametre par reference
////////////////////////////////////////////////////////////////////////////////////////////////////
void echangeB (int &a, int &b){
        int c;
        c = a;
        a = b;
        b = c;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// Passage de parametre par pointeurs
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
void echange (int *a, int * b){
        int c;
        c = *a;
        *a = *b;
        *b = c;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

int main()
{
        int x, y;

        x=5; y=6;

        affiche (x,y);
        echange(x,y);
        affiche (x,y);

        affiche (x,y);
        echangeB(x,y);
        affiche (x,y);

        affiche (x,y);
        echange(&x,&y);
        affiche (x,y);

        return (EXIT_SUCCESS);
}
```

4 Références et pointeurs

Soit le type structure défini ainsi :

```
typedef struct {
 int stock;
 float prix;
 int ventes [NMOIS];
} enreg;
```

Ecrire une fonction `raz` qui "remet à zéro" les champs `stock` et `ventes` d'une structure de ce type, transmise en argument. La fonction ne comportera pas de valeur de retour. Ecrire une fonction passant la structure par référence et une autre la passant par pointeur.

Ecrire un petit programme d'essai qui affecte tout d'abord les valeurs aux différents champs d'une telle structure, avant de leur appliquer la fonction `raz`. On affichera les valeurs de la structure, avant et après appel (on pourra s'aider d'une fonction d'affichage).

```
#include<iostream>
using namespace std;

const int NMOIS=12;

struct
{
        int stock;
        float prix;
        int ventes [NMOIS];
} Enreg;

////////////////////////////////////////////////////////////////////////////////////////////////////
// fonction d initialisation arbitraire pour tests
////////////////////////////////////////////////////////////////////////////////////////////////////
void init (Enreg & e){
        e.stock = 12;
        e.prix = 5.25;
        for (int i=0; i< NMOIS; i++)
        {
                e.ventes[i]=i;
        }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
// Remise a zero de tous les champs de la structure avec passage par reference
////////////////////////////////////////////////////////////////////////////////////////////////////
void raz (Enreg & e)
{
        e.stock = 0;
        e.prix = 0.0;
        for (int i=0; i< NMOIS; i++)
        {
                e.ventes[i]=0;
        }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// Remise a zero de tous les champs de la structure avec passage par pointeur
////////////////////////////////////////////////////////////////////////////////////////////////////
void raz (Enreg * e)
{
        e->stock = 0;
        e->prix = 0.0;
        for (int i=0; i< NMOIS; i++)
        {
                e->ventes[i]=0;
        }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// affichage de la structure.
// l enregistrement est passe par reference par soucis de performance (non recopie de tous les champs)
////////////////////////////////////////////////////////////////////////////////////////////////////
void affiche (Enreg & e){
        cout << "Stock : " << e.stock << endl;
        cout << "Prix : " << e.prix << endl;
        for (int i=0; i<NMOIS; i++)
        {
                        cout << " " << e.ventes[i];
        }
        cout << endl<< endl;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

int main()
{

        Enreg enreg;

        // avec passage par reference
        init (enreg);
        affiche (enreg);
        raz (enreg);
        affiche (enreg);

        // avec passage par pointeur
        init (enreg);
        affiche (enreg);
        raz (&enreg);
        affiche (enreg);

        return (EXIT_SUCCESS);
}
```