TP C++: Serie 4

1. Fonction template:

```
template <class T> // ou template <typename T>
type_retour nomFonction(T p1)
{
   ... .. ...
}
```

Exemple:

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

template <class T>
T maxi(T n1, T n2)
{
   return (n1 > n2) ? n1 : n2;
}

// programme principal
int main()
{
   cout<< maxi(5,2)<<endl; // max de deux entiers
   cout<< maxi(3.5,2.75)<<endl; // max de deux réels
   cout<< maxi('Z','M')<<endl; // max de deux réels
   return 0;
}
```

2. Class template:

```
template <class T>
class nomClasse
{
   ... .. ...
   public:
      T attr;
      T nomfonction(T arg);
   ... .. ...
};
```

Exemple:

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

// définition de la classe
template <class T>
class Tableau {
    private:
        T *tab;
        int taille;
    public:
        Tableau(T t[], int n);
        void afficher();
};

// implémentation des méthodes
template <class T>
Tableau<T>::Tableau(T t[], int n) {
    tab = new T[n];
    taille = n;
    for(int i = 0; i < taille; i++)
        tab[i] = t[i];
}
```

Exercice 1: les templates

Requis:

savoir utiliser les pointeurs et l'allocation dynamique de mémoire,
savoir créer une classe template,
savoir créer une structure.

Énoncé:

Créez une **classe liste** simplement chaînée, avec une classe liste. Cette classe a un pointeur sur le premier élément de la liste. Elle a une méthode pour ajouter ou supprimer un élément au début de la liste et une pour afficher la liste en entier. Évitez toute fuite mémoire. Les éléments de la liste seront contenu dans la structure element.

Correction: https://github.com/hm43/ExercicesTpCpp.git