TD N°03

Type de données abstraits

Problème

La file est une structure de données organisée d'une manière à ce que le premier élément qui entre dans la file sera le premier élément qui en sort.
On veut définir un type de données abstrait pour modéliser une file tel que le type des éléments de la file est inconnu.
On utilisant le langage C++ :

1. Définir la classe **MAFILE** qui représente le type de données abstrait, avec un attribut « **nbElem** », un pointeur vers la tête de la file et un pointeur vers la queue de la file.
2. Définir une classe **Element** qui représente les éléments de la file. Un élément de la file qui est sous forme d'un couple <valeur, pointeur> tel que "valeur" représente la valeur de l'élément dont le type est inconnu, le "pointeur" pointe sur l'élément suivant.
3. Définir le constructeur de la classe **MAFILE** pour initialiser la file pour quelle soit vide à sa création.
4. Définir une fonction appelée **estVide()** qui permet de vérifier si la file est vide.
5. Définir une fonction appelée **premier()** qui permet de récupérer la valeur de la tête de la file.
6. Définir une fonction appelée **inserer()** qui permet d'insérer un élément à la fin de la file.
7. Définir une fonction appelée **enlever()** qui permet d'enlever le premier élément de la file.
8. Définir la fonction **main()** qui utilise la classe **MAFILE** comme suit :
    a. Créer une file avec des éléments de type entier.
    b. Insérer les 10 premiers entiers positifs dans la file.
    c. Supprimer 5 éléments.
    d. Créer une file avec des éléments de type réel.
    e. Insérer les réels suivants dans la file (-1.1, 1.5, 2.0).
    f. Supprimer un élément.

Correction du TD n°03

Type de données abstraits

Problème

Le code en C++ est comme suit :
Le programme est organisé en 3 fichiers : Element.h, MAFILE.h, MAFILE.cpp

Le fichier Element.h

```
#pragma once
#define NULL 0
template <typename Valeur>
class Element
	{
	public:
		Valeur val;
		Element *suiv;
	};
```

Le fichier MAFILE.h

```
#pragma once
#include "Element.h"
template <typename Valeur>
class MAFILE
	{
	public:
		int nbElem;
		Element<Valeur> * tete;
		Element<Valeur> * queue;
	public:
		MAFILE(void);
		bool estVide();
		Element<Valeur> *premier();
		void inserer(Element<Valeur> *e);
		void enlever();
		void afficher(Element<Valeur> *e);
	};
```

Le fichier MAFILE.cpp

```
#include "MAFILE.h"
#include <iostream>
using namespace std;

template <typename Valeur>
MAFILE<Valeur>::MAFILE(void)
	{
	nbElem = 0;
	tete = NULL;
```

```
	queue = NULL;
	}

template <typename Valeur>
bool MAFILE<Valeur>::estVide()
	{
	return (nbElem == 0);
	}

template <typename Valeur>
Element<Valeur> *MAFILE<Valeur>::premier()
	{
	return tete;
	}

template <typename Valeur>
void MAFILE<Valeur>::inserer(Element<Valeur> *e)
	{
	if (this->estVide())
		{
		tete = e;
		queue = e;
		}
	else
		{
		queue->suiv = e;
		queue = e;
		}
	nbElem++;
	}

template <typename Valeur>
void MAFILE<Valeur>::enlever()
	{
	if (!estVide())
		{
		Element<Valeur> *tmp = tete;
		tete = tete->suiv;
		delete tmp;
		nbElem--;
		}
	}

//Afficher le contenu de la file de l'élément e jusqu'à la fin
template <typename Valeur>
void MAFILE<Valeur>::afficher(Element<Valeur> *e)
	{
	if (e != NULL)
		{
		cout << e->val << endl;
		afficher(e->suiv);
		}
	}

int main(int argc, char **argv)
	{
	//Créer une file avec des éléments de type entier
	MAFILE<int> *intfile = new MAFILE<int>();

	//Insérer les 10 premiers entiers positifs dans la file
	for (int i = 0; i < 10; i++)
		{
		Element<int> *elem = new Element<int>();
		elem->val = i;
```

```
        elem->suiv = NULL;
        intfile->inserer(elem);
        }

cout << "Affichage de la file de 10 entiers" << endl;
intfile->afficher(intfile->tete);

// Supprimer 5 éléments
for (int i = 0; i < 5; i++)
        {
        intfile->enlever();
        }

cout << "Affichage de la file après la suppression de 5 éléments" << endl;
intfile->afficher(intfile->tete);

//Créer une file avec des éléments de type réel
MAFILE<float> *realfile = new MAFILE<float>();

//Insérer les réels suivants dans la file (-1.1, 1.5, 2.0)
Element<float> *elem = new Element<float>();
elem->val = (float)-1.1;
elem->suiv = NULL;
realfile->inserer(elem);
elem = new Element<float>();
elem->val = (float)1.5;
elem->suiv = NULL;
realfile->inserer(elem);
elem = new Element<float>();
elem->val = (float)2.0;
elem->suiv = NULL;
realfile->inserer(elem);

cout << "Affichage de la file des réels" << endl;
realfile->afficher(realfile->tete);

//Supprimer un élément
realfile->enlever();

cout << "Affichage de la file des réels après suppression d'un élément" << endl;
realfile->afficher(realfile->tete);

system ("pause");

return 0;
}
```