TP7
SI4 – Liste chaînée

Ce TP est la suite du TP6. Dans cette nouvelle approche, l'objectif est de réaliser l'application à travers un langage objet.

Le langage C++ permettra de mettre au point notre application, qui intégrera, les éléments du TP6.

L'approche objet nécessite :

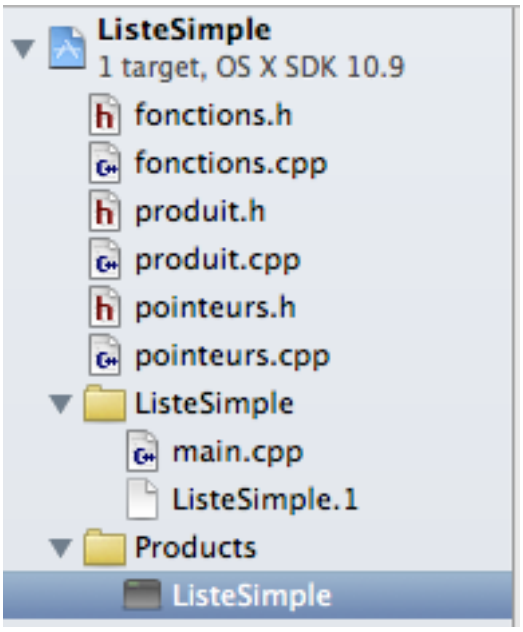
- Des classes d'objets avec des attributs et des méthodes
- Des classes qui définissent une visibilité de l'interface
- Une encapsulation qui protège les données
- Des méthodes spécifiques (constructeurs, destructeurs)
- Une interface qui fait l'objet d'une implémentation
- Une interface et une implémentation présentées dans des fichiers différents
- La création dynamique d'éléments (new en C++)
- La manipulation du pointeur this, qui pointe sur l'objet courant

Afin d'être plus concret je vous propose de mettre en pratique le code proposé plus loin. Attention, le code proposé a été réalisé avec Xcode en C++. Cela suppose quelques adaptations à travers VS.

Les standards présentés en cours sont mis en application dans ce code :

- Répartition de l'interface et de l'implémentation des fonctions dans des fichiers différents.
- Interface et implémentation des classes dans des fichiers différents
- Utilisation de procédures et de fonctions pour factoriser le code.
- Utilisation de méthodes pour encapsuler les données (visibilité privée).
- Passage de paramètres entre programme appelant et sous-programme.
- Pas de variables globales au travers du programme et des sous-programmes.
- Utilisation de constructeurs pour fabriquer les objets.
- Manipulation du pointeur this.

La hiérarchie du projet est la suivante :



Pour réaliser le travail, vous pouvez utiliser Xcode ou VS ou tout autre IDE, pouvant traiter le langage C++.

Interface des fonctions : fonctions.h

```
//
//  fonctions.h
//  ListeSimple
//
//  Created by Sébastien Gagneur on 01/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//
// INTERFACE DES STRUCTURES = DEFINITION DES TYPES PERSONNALISES
// INTERFACE DES FONCTIONS = MODE D'EMPLOI

#ifndef ListeSimple_fonctions_h
#define ListeSimple_fonctions_h

#include <string>
#include "produit.h"
#include "pointeurs.h"

// interface des fonctions. Le mode d'emploi

// création d'un nouveau produit
produit * creerProduit();

// insertion du produit dans la liste
pointeurs * inserProduit(produit * unProduit, produit * debut, produit * fin);

// affichage du produit courant
void afficProduit(produit * unProduit);

// affichage complet de la liste
void listeProduit(produit * debut, produit * fin);

// affichage du nombre de produit contenu dans la liste

#endif
```

Implémentation des fonctions : fonctions.cpp

```
//
//  fonctions.cpp
//  ListeSimple
//
//  Created by Sébastien Gagneur on 01/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//
// IMPLEMENTATION DES FONCTIONS = CODE DES FONCTIONS

// Ce dont j'ai besoin ici !
#include "fonctions.h"
#include "pointeurs.h"
#include "produit.h"

#include <iostream>
#include <string>


// Directives du préprocesseur pour gérer les inclusions multiples
#ifndef ListeSimple_fonctions_cpp
```

```
#define ListeSimple_fonctions_cpp

//implémentation des fonctions de la classe produit

//créer un produit
produit * creerProduit()
{
    // allocation dynamique qui permet d'allouer un nouvel
emplacement pour chaque nouveau produit créé
    // si vous ne mettez pas l'allocation dynamique, le produit est
une variable statique, et l'emplacement,
    // l'adresse du produit ne change jamais, si bien, que chaque
produit créé remplace le précédent.
    // Dans ce cas les nouveaux produits créés ne remplacent pas les
précédents, ils sont placés à des
    // adresses différentes.

    produit * unProduit = new produit();
    int num;
    std::string lib;
    float pri;
    int cat;

    std::cout<<"Code produit ?:";
    std::cin>>num;
    unProduit->setNum(num);
    std::cout<<"Libellé produit ?:";
    std::cin>>lib;
    unProduit->setLib(lib);
    std::cout<<"Prix produit ?:";
    std::cin>>pri;
    unProduit->setPri(pri);
    std::cout<<"Catégorie produit ?:";
    std::cin>>cat;
    unProduit->setCat(cat);
    // Le procuit suivant n'est pas encore connu !
    unProduit->setSuiv(NULL);
    return unProduit;
}

// Afficher un produit
void affichProduit(produit * unProduit)
{
    std::cout<<"num : "<<unProduit->getNum()<<"lib : "<<unProduit-
>getLib()<<"prix : "<<unProduit->getPri()<<"cat : "<<unProduit-
>getCat()<<"\n";
}

// Insérer un produit
pointeurs * inserProduit(produit * unProduit, produit * debut,
produit * fin)
{
    // structure pour récupérer mes pointeurs, la fonction ne
retournant pas mes modifications
```

```
    pointeurs * mesPointeurs = new pointeurs();
    // Si la liste ne contient pas de produit
    if ( debut == NULL)
    {
        // le seul est unique produit, constitue la début et la fin
de la liste
        debut = unProduit;
        fin = unProduit;
    }
    else
        // La liste contient déjà au moins un produit, et donc cette
fois il faut faire le chaînage
    {
        // le dernier produit de liste à pour suivant le nouveau
produit
        fin->setSuiv(unProduit);
        // le dernier produit, constitue la nouvelle fin
        fin = unProduit;
    }
    // je mets à jour ma structure pour faire remonter dans le
programme principal mes modifications sur
    // mes pointeurs
    mesPointeurs->setDeb(debut);
    mesPointeurs->setFin(fin);
    return mesPointeurs;
}

// Parcourir la liste
void listeProduit(produit * debut, produit * fin)
{
    // le pointeur sur le produit courant
    produit * courant = new produit();

    // le produit courant est le premier
    courant = debut;

    // tant que l'adresse du produit que je traite est différente de
l'adresse du dernier produit je traite
    while (courant != fin)
    {
        // j'affiche le produit courant
        affichProduit(courant);
        // le nouveau produit courant, est le produit suivant du
produit courant !
        courant = courant->getSuiv();
    }
    // Si vous voulez voir le dernier !
    affichProduit(courant);

}

#endif
```

Interface classe produit : produit.h

```
//
//  produit.h
//  ListeSimple
//
//  Created by Sébastien Gagneur on 02/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//

#ifndef ListeSimple_produit_h
#define ListeSimple_produit_h

#include <string>

// classe d'un produit
class produit
{
    private :

    int numProduit;
    std::string libProduit;
    float priProduit;
    int catProduit;
    struct produit * suivProduit;

    public :

    produit(int num = 0, std::string lib = "", float pri = 0, int cat = 0 );
    produit setNum(int num);
    produit setLib(std::string lib);
    produit setPri(float pri);
    produit setCat(int cat);
    produit setSuiv(produit * pro);
    int getNum();
    std::string getLib();
    float getPri();
    int getCat();
    produit * getSuiv();

};


#endif
```

Implémentation classe produit : produit.cpp

```
//
//  produit.cpp
//  ListeSimple
//
//  Created by Sébastien Gagneur on 02/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//

#include "produit.h"

produit::produit(int num, std::string lib, float pri, int cat)
```

```
{
    this->numProduit = num;
    this->libProduit = lib;
    this->priProduit = pri;
    this->catProduit = cat;
    this->suivProduit = NULL;
}

produit produit::setNum(int num)
{
    this->numProduit = num;
    return *(this);
}
produit produit::setLib(std::string lib)
{
    this->libProduit = lib;
    return *(this);
}
produit produit::setPri(float pri)
{
    this->priProduit = pri;
    return *(this);
}
produit produit::setCat(int cat)
{
    this->catProduit = cat;
    return *(this);
}

produit produit::setSuiv(produit * pro)
{
    this->suivProduit = pro;
    return *(this);
}


int produit::getNum()
{
    return this->numProduit;
}

std::string produit::getLib()
{
    return this->libProduit;
}

float produit::getPri()
{
    return this->priProduit;
}

int produit::getCat()
{
    return this->catProduit;
```

```
}

produit * produit::getSuiv()
{
    return this->suivProduit;
}
```

Interface classe pointeurs : pointeurs.h

```
//
//  pointeurs.h
//  ListeSimple
//
//  Created by Sébastien Gagneur on 02/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//

#ifndef ListeSimple_pointeurs_h
#define ListeSimple_pointeurs_h

#include "produit.h"


class pointeurs
{
    private :

    produit * debut;
    produit * fin;

    public :

    pointeurs();
    pointeurs * setDeb(produit * deb);
    pointeurs * setFin(produit * fin);
    produit * getDeb();
    produit * getFin();

};

#endif
```

Implémentation classe pointeurs : pointeurs.cpp

```
//
//  pointeurs.cpp
//  ListeSimple
//
//  Created by Sébastien Gagneur on 02/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//

#include "pointeurs.h"

pointeurs::pointeurs()
{
    this->debut = NULL;
    this->fin = NULL;
}


pointeurs * pointeurs::setDeb(produit * debut)
{
    this->debut = debut;
    return (this);
}

pointeurs * pointeurs::setFin(produit * fin)
{
    this->fin = fin;
    return (this);
}

produit * pointeurs::getDeb()
{
    return (this->debut);
}

produit * pointeurs::getFin()
{
    return (this->fin);
}
```

Programme principal : main.cpp

```
//
//  Created by Sébastien Gagneur on 01/11/2013.
//  Copyright (c) 2013 Sébastien Gagneur. All rights reserved.
//

#include <iostream>
#include "fonctions.h"
#include "produit.h"
#include "pointeurs.h"

// Programme principal
int main(int argc, const char * argv[])
{

    // insert code here...
    produit * unProduit;
    pointeurs * mesPointeurs = new pointeurs();
    produit * debut = NULL;
    produit * fin = NULL;
    char reponse[4] = "oui";

    // Codes couleurs présentés dans le fichier d'interface
    std::cout<<"\033[32;01mListe Chaine Simple\033[00m\n";

    // la boucle pour gérer n produit
    while (strcmp(reponse,"oui") == 0)
    {
        // création d'un produit
        unProduit = creerProduit();
        // insertion d'un produit
        mesPointeurs = inserProduit(unProduit, debut, fin);
        // mise à jour des pointeurs, pour avoir connaissance des nouvelles valeurs
        debut = mesPointeurs->getDeb();
        fin = mesPointeurs->getFin();
        std::cout<<"Autre produit (oui/non) : ?";
        std::cin>>reponse;
    }
    // Affichage de la liste constituée
    listeProduit(debut,fin);
    return 0;
}
```

Remarque : Le projet ne contient pas de classe Liste, mais il est possible de créer cette classe et de faire en sorte qu'une classe contienne des produits. La classe pointeurs pourrait illustrer ce rôle.

Pour cela, il est nécessaire de mener une réflexion lorsque l'on commence un projet. Un langage (UML) permet d'illustrer cette réflexion avec un diagramme de classe.

Travail à faire :

1. Reconstruire le projet à l'aide du code proposé.

2. Intégrer vos apports du TP6 dans cette nouvelle version.