TP 4 - Forme Canonique de Coplien

Programmation Orientée Objet

Objectif

- Comprendre le principe de la forme canonique de Complien

1 La Classe chaîne

On vous demande de créer une nouvelle classe selon la forme canonique de Coplien. Celle-ci est une chaîne simple dotée de 3 attributs :

- un tableau de caractères (`char *`) contenant la chaîne proprement dite.
- Un entier dénotant la capacité de la chaîne, c'est à dire le nombre de caractères qu'elle peut physiquement contenir à un instant donné. Il s'agit donc de la capacité physique du tableau de caractères.
- Un entier dénotant la longueur de la chaîne, c'est à dire le nombre de caractères significatifs qu'elle contient. Il s'agit de l'information renvoyée par l'application de la fonction `strlen` sur le tableau de caractères.

Le constructeur par défaut créera une chaîne de longueur nulle mais de capacité 10 caractères. Le constructeur par recopie devra positionner correctement la capacité et la longueur mais également allouer la mémoire du tableau de caractères et recopier le modèle. L'opérateur d'affectation, après nettoyage de la mémoire, devra allouer la mémoire puis recopier la chaîne du modèle. Attention, il ne faut surtout pas utiliser `strdup` car ce dernier utiliser `malloc`, incompatible avec `new` et `delete`.

```
//-------------------------------------------------------
// Classe exemple des chaines de caractères JMB 2007 MODIF NB 2009
//-------------------------------------------------------
#include <iostream>
using namespace std;
#include <string.h>
#define DEFAULT_CAP 10

class Chaine {
        // implémentation
        int capacite;
        int longueur;
        char *tab;

  public:
        // constructeurs & destructeur
        Chaine (int cap = DEFAULT_CAP); // par défaut
        Chaine (char *);                                    // à partir d'un char *
        Chaine (const Chaine &);                  // par recopie
        ~Chaine();

        // accesseurs
        int get_capacite();
        int get_longueur();
        char operator[](int);

        // autres
        Chaine & operator=(const Chaine &);
};//-------------------------------------
// constructeurs & destructeur
Chaine::Chaine(int cap) {
        if (cap <= 0)   //cas du zero
                capacite = DEFAULT_CAP;
        else if (cap < 10) //cas du inferieur a 10
                capacite = DEFAULT_CAP;
        else{
                int pages = cap % DEFAULT_CAP; // plus d'une page requise si > 0
                capacite = ((cap / DEFAULT_CAP)+1)*DEFAULT_CAP;
        }
```

```
        longueur = 0;
        tab = new char[capacite];
        tab[0]='\0'; // chaine vide
}
Chaine::Chaine(char* ch) {
        int longch = strlen(ch);
        if (longch<=0){
                capacite = DEFAULT_CAP;
                longueur=0;
        }
        else{
                int pages = longch % DEFAULT_CAP; // plus d'une page requise si > 0
                if (pages != 0) capacite = ((longch / DEFAULT_CAP)+1)*DEFAULT_CAP;
                else capacite = longch;
                longueur = longch;
                }
        tab = new char[capacite];
        strcpy(tab,ch);
}
Chaine::Chaine(const Chaine &ch) {
        capacite = ch.capacite;
        longueur = ch.longueur;
        tab = new char[capacite];
        strcpy(tab,ch.tab);
}
Chaine::~Chaine() {
        if (tab) delete[] tab; //on ne delete que si alloué
}
//accesseurs
int Chaine::get_capacite() {
        return capacite;
}
int Chaine::get_longueur() {
        return longueur;
}
char Chaine::operator[](int rang) {
        if ((rang>0) && (rang<longueur))
                return tab[rang];
        else return 0;
}
// autres
Chaine & Chaine::operator=(const Chaine &ch) {
        if (tab) // déjà alloué
                delete [] tab;
        capacite = ch.capacite;
        longueur = ch.longueur;
        tab = new char[capacite];
        strcpy(tab,ch.tab);
        return *this;
}


int main (){
        Chaine C1;
        char toto[]="Hello";
        Chaine C2(toto);
        C1=C2;
        cout << C1[0] << endl;
        return 0;
}
```

2 La Classe chaîne intelligente

Modifiez la classe précédente de manière à prendre en compte les points suivants. On souhaite rendre ces chaînes "intelligentes" en leur permettant de ne stocker qu'une fois les chaînes identiques. L'idée est de faire en sorte que si une chaîne `ch1` contient "coucou", alors l'affectation `ch2 = ch1` provoque la situation illustré par la figure FIG. 1 en mémoire :

Ainsi la destruction de ch2 ne provoque pas la libération de la zone mémoire contenant les caractères, tandis que la destruction de ch1 (en considérant qu'elle est la " dernière " à contenir ces caractères) provoque la libération de la zone

Ce changement de comportement va impliquer un changement de structure de la classe. Modifiez la structure en conséquence, et donc aussi toutes les méthodes associées.

```
//-------------------------------------------------
```

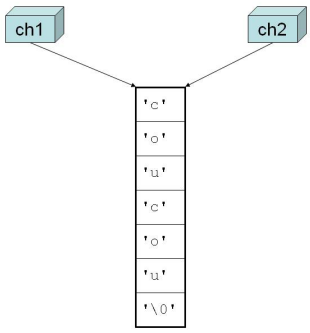

FIGURE 1 – Exemple de chaîne intelligente

```
// Classe exemple des chaines de caractères JMB 2007
//-------------------------------------------------------
#ifndef ChaineI_H
#define ChaineI_H

#include <iostream>
#include <string.h>
using namespace std;
#define DEFAULT_CAP 16

typedef struct{
        int capacite;
        int longueur;
        char * tab;
        int cpt; // COMPTEUR DE REFERENCES (clé du problème)
} Chaine;

class ChaineInt {
        // implémentation
        Chaine *chaine;
  public:
        // constructeurs & destructeur
        ChaineInt (int cap = DEFAULT_CAP); // par défaut
        ChaineInt (char *);                          // à partir d'un char *
        ChaineInt (const ChaineInt &);          // par recopie
        ~ChaineInt();

        // accesseurs
        int get_capacite();
        int get_longueur();
        char operator[](int);

        // autres
        ChaineInt & operator=(const ChaineInt &);
        friend  ostream & operator<<(ostream &, ChaineInt &);
};//-------------------------------------------------

#endif
```

```
//-------------------------------------------------------
// Classe exemple des chaines de caractères JMB 2007
//-------------------------------------------------------
#include "ChaineInt.hpp"

ChaineInt::ChaineInt(int cap) {
        chaine = new Chaine;
        chaine->capacite = cap;
        chaine->longueur = 0;

        chaine->tab = new char[chaine->capacite];
        chaine->tab[0]='\0'; // chaine vide
        chaine->cpt=0;
        #ifdef DEBUG
                cout << "  ***** ChaineInt - Constructeur par defaut\n";
```

```
                cout << "  *****                    capacite = " << chaine->capacite << "\n";
                cout << "  *****                    longueur = " << chaine->longueur << "\n";
                cout << "  *****                    cpt      = " << chaine->capacite << "\n";
                cout << "  *****                    tab      = " << chaine->tab << "\n";
        #endif
}
ChaineInt::ChaineInt(char* ch) {
        chaine = new Chaine;
        int longch = strlen(ch);
        int pages = longch % DEFAULT_CAP; // plus d'une page requise si > 0
        if (pages != 0)
                chaine->capacite = ((longch / DEFAULT_CAP)+1)*DEFAULT_CAP;
        else
                chaine->capacite = longch;
        chaine->longueur = longch;
        chaine->tab = new char[chaine->capacite];
        strcpy(chaine->tab,ch);
        chaine->cpt=0;
        #ifdef DEBUG
                cout << "  ***** ChaineInt - Constructeur par chaine passee en parametre\n";
                cout << "  *****                    parametre = " << ch << "\n";
                cout << "  *****                    capacite = " << chaine->capacite << "\n";
                cout << "  *****                    longueur = " << chaine->longueur << "\n";
                cout << "  *****                    cpt      = " << chaine->capacite << "\n";
                cout << "  *****                    tab      = " << chaine->tab << "\n";
        #endif

}
ChaineInt::ChaineInt(const ChaineInt &ch) {
        chaine = new Chaine;
        chaine->longueur = ch.chaine->longueur;
        chaine->capacite = ch.chaine->capacite;
        chaine->tab = new char[chaine->capacite];
        strcpy(chaine->tab,ch.chaine->tab);
        chaine->cpt=0;
        #ifdef DEBUG
                cout << "  ***** ChaineInt - Constructeur par recopie\n";
                cout << "  *****                    capacite = " << chaine->capacite << "\n";
                cout << "  *****                    longueur = " << chaine->longueur << "\n";
                cout << "  *****                    cpt      = " << chaine->capacite << "\n";
                cout << "  *****                    tab      = " << chaine->tab << "\n";
        #endif
}

ChaineInt::~ChaineInt() {
        #ifdef DEBUG
                cout <<  "**** ChaineInt - destructeur\n";
                cout <<  "****                          cpt= " << chaine->cpt <<"\n";
                cout <<  "****                          longueur= " << chaine->longueur <<"\n";
                cout <<  "****                          capacite= " << chaine->capacite <<"\n";
                cout <<  "****                          tab= " << chaine->tab <<"\n";
        #endif
        if (chaine!=NULL)
        {
                if (chaine->cpt!=0)
                {
                        /* la chaine est pointe par une autre classe */
                        #ifdef DEBUG
                                cout << "****           --> Pointee par une autre classe - Pas de liberation d
                        #endif
                        chaine->cpt--;
                }
                else{
                        /* la chaine n est pointee que par cette classe */
                        #ifdef DEBUG
                                cout << "****            --> Non Pointee par une autre classe - liberation de
                        #endif
                        delete [] (chaine->tab);
                        delete chaine;
                }
        }
}

int ChaineInt::get_capacite() {
        return chaine->capacite;
}
int ChaineInt::get_longueur() {
        return chaine->longueur;
}
char ChaineInt::operator[](int rang) {
        if ((rang>0) && (rang<chaine->longueur))
```

```
                return chaine->tab[rang];
        else return 0;
}

// autres

ChaineInt & ChaineInt::operator=(const ChaineInt &ch) {
        #ifdef DEBUG
                cout << "Trace : ChaineInt - affectation\n";
        #endif
        if (chaine!=NULL){
                if (chaine->capacite!=0) {
                        #ifdef DEBUG
                                cout << "\t chaine recevante a deja un tab\n";
                        #endif
                        if (chaine->cpt!=0) {
                                /* la chaine est pointe par une autre classe */
                                #ifdef DEBUG
                                        cout << "\t chaine recevante a deja un tab pointee par autre classe\n"
                                #endif
                                (chaine->cpt)--;
                                chaine=NULL;
                        }else{
                                /* la chaine n est pointee que par cette classe */
                                #ifdef DEBUG
                                        cout << "\t chaine recevante a deja un tab pointee seulement par elle\
                                #endif
                                delete [] (chaine->tab);
                                delete chaine;
                        }
                }
        }
        /* affectation */
        chaine = ch.chaine;
        chaine->cpt++;
        return *this;
}

ostream& operator<<(ostream& flot,ChaineInt &ch){
        flot << "Chaine : " << ch.chaine->tab << " de longueur : " << ch.chaine->longueur ;
        flot << " et de capacite : " << ch.chaine->capacite << " avec un compteur de : "<< ch.chaine->cpt <<"\
        return flot;
}

int main(){
        ChaineInt c1;
        ChaineInt c2("toto");
        ChaineInt c3("titi");

        cout << c1;
        cout << c2;
        cout  << c3;

        c1 = c2;
        c1 = c3;

        cout <<c1;
        cout <<c2;
        cout <<c3;
}
```