TP1 – Programmation Objet

Wairy Fabien – Pecro Kévin Groupe B

```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

using namespace std;

//https://github.com/kevinpecro/tp1_conception_objet.git

class MyString
{
private:
    char *tab;
    int n; // Taille de la chaine
    int stat[26]; //Decompte du nombre d'utilisation de chaque lettre de l'alphabet
    int spe;  //Nombre de caracteres speciaux


public:
    MyString();
    MyString(char*);
    MyString(char,int);
    ~MyString();
    MyString(const MyString&);
    MyString &operator=(const MyString &);
    MyString operator+=(MyString &);
    friend ostream& operator<<(ostream& os, const MyString& st);
    void affiche();
    void majstat();
    void concatenation(MyString);
    void supprimer_un_carac(char);
    void dedouble(char);
    void maj(char);
    char* GetTab();

};
```

1. **Définir la classe MyString en intégrant les attributs :**
    a. Un constructeur chargé d'initialiser à NULL le pointeur de chaîne

Le constructeur d'initialisation à NULL de la classe MyString va initialiser tous les éléments qui le compose à NULL ou 0.

```
MyString::MyString()
{
    tab=NULL;
    n=0;
    spe=0;
    majstat();
}
```

b. Un constructeur chargé d'initialiser la chaine avec un « tableau chaîne » (char*) passé en paramètre

Le constructeur initialise une chaine de caractère qui lui est transmise en paramètre. On compte le nombre de caractère, puis initialise un nouveau tableau redimensionné.

```
MyString::MyString(char* chaine)
{
    n=strlen(chaine);
    tab=new char[n+1];
    strcpy(tab,chaine);
    majstat();
}
```

c. Un constructeur permettant d'initialiser la chaine avec un même caractère répété en X fois (ex : Mystring s1('a',3) => « aaa »)

Le constructeur initialise la chaine tab avec un redimensionnement du tableau puis le rempli d'un caractère unique sur la longueur de la chaine souhaitée.

```
MyString::MyString(char caractere, int x)
{
    n=x;
    int i;
    tab=new char[n+1];
    for ( i = 0; i < x ; ++i)
    {
        tab[i]= caractere;
    }
    tab[i]='\0';
    majstat();
}
```

d. Un destructeur

Le destructeur permet de récupérer les ressources mémoire en détruisant les éléments déjà utilisé.

```
MyString::~MyString(){}
```

e. Un constructeur par recopie

Le constructeur initialise la chaine à partir d'un objet de type MyString déjà existant passer en paramètre.

```
MyString::MyString(const MyString& s)
{
    int i;
    n = s.n ;
    tab=new char[n];
    for(i=0; i<n; i++)
    {
        tab[i]=s.tab[i];
    }
    tab[i]='\0';
    majstat();
}
```

f. Des accesseurs :il conviendra de déterminer ceux qui sont nécessaires

L'accesseur GetTab permet d'accéder plus simplement au contenu de la chaine tab.

```
char* MyString::GetTab()
{
    return tab;
}
```

g. Une méthode permettant d'afficher la chaîne

Méthode d'affichage des différentes action et contenu des variables.

Affichage du tableau tab de la variable n, du nombre d'apparition de chaque caractère dans le tableau stat et du nombre de caractère spéciaux.

```
void MyString::affiche()
{
    if (tab!=NULL)
    {
        cout<<"chaine = "<<tab<<endl;
    }
    else
    {
        cout<<"chaine = "<<endl;
    }
    cout<<"n = "<<n<<endl;
    cout<<"stat : "<<endl;
    for(int i=0; i<26; i++)
    {
        if(stat[i]!=0)
        {
            cout<<"nb de car "<<(char)('A'+i)<<"="<<stat[i]<<endl;
        }
    }
    cout<<"spe = "<<spe<<endl;
}
```

h. Une méthode chargée de supprimer un caractère dans la chaine

La méthode permet de supprimer le caractère d'une chaine, ce caractère lui est envoyé en paramètre. Un tableau provisoire est créé puis on redimensionne le tableau final avant de transférer la chaine sans le caractère à supprimer.

```
void MyString::supprimer_un_carac(char carac_a_suppr)
{
    int j=0, taille, test=0;
    char *res;
    taille=strlen(tab)+1;

    res=(char*)malloc((taille+1)*sizeof(char));

    for(int i=0; i < n ; i++)
    {
        if((tab[i]!=carac_a_suppr) && (tab[i]!='\0'))
        {
            res[j]=tab[i];
            j++;
        }
    }
    res[j]='\0';
    delete tab;
    tab=res;
    n=0;
    for (int i = 0; tab[i]!='\0'; ++i)
    {
       n++;
    }
    majstat();
}
```

i. Une méthode chargée de dédoubler un caractère dans la chaine

Cette méthode est chargée de dédoubler un caractère en effectuant un redimensionnement de tableau en utilisant un malloc puis de transférer le nouveau contenu de la chaine dans le nouveau tableau.

```
void MyString::dedouble(char carac_a_doubler)
{
    int j=0, taille;
    char  *res;
    taille=strlen(tab)+1;

    res=(char*)malloc((taille+1)*sizeof(char));

    for (int i=0; i < n ; i++)
    {
        if(tab[i]==carac_a_doubler)
        {
            res[j]=tab[i];
            j++;
            res[j]=tab[i];
            j++;
        }
        else
        {
            res[j]=tab[i];
            j++;
        }
    }
    res[j]='\0';
    delete tab;
    tab=res;
    n=0;
    for (int i = 0; tab[i]!='\0'; ++i)
    {
       n++;
    }
    majstat();
}
```

j. Une méthode chargée de concaténer la chaîne avec une seconde passée en paramètre (paramètre objet)

Cette méthode concatène deux chaines de caractère entre elles, on insère la chaine passé en paramètre dans un nouveau tableau à laquelle on avait déjà insérer la première chaine de caractère.

```
void MyString::concatenation(MyString chaine)
{
    char* fusion;
    int taille,i,j;
    taille = n+chaine.n+1;
    fusion =new char[taille];
    for( i=0; i < n ;i++)
    {
        fusion[i]=tab[i];
    }

    for(j=0; j < chaine.n ;j++,i++)
    {
        fusion [i]=chaine.tab[j];
    }
    fusion[i]='\0';
    delete tab;
    tab=fusion;
    n = taille-1;
    majstat();

}
```

k. Une méthode chargée de transformer la chaîne en son équivalent majuscule

Cette méthode utilise la fonction toupper qui va convertir chaque caractère d'une chaine en son équivalent majuscule.

```
void MyString::majstat()
{
    int i;
    char car;
    spe=0;
    for (int i = 0; i < 26; ++i)
    {
        stat[i]=0;
    }
    for (int i = 0; i < n; ++i)
    {
        car=toupper(tab[i]);
        if ((car>='A')&&(car<='Z'))
        {
            stat[car-'A']++;
        }
        else
        {
            if (car!='\0')
            {
                spe++;
            }
        }
    }
}
```

2. **Surcharger les trois opérateurs suivants :**
    a. L'opérateur = : il devra permettre de gérer des affectations tu type s1=s2

L'opérateur égale compare l'objet d'origine avec celui passé en paramètre, s'ils sont différents, on affecte celui passé en paramètre dans l'objet.

```
MyString& MyString::operator=(const MyString &b)
{
    int i;
    if(this !=&b)
    {
        delete tab;
        tab = new char[n = b.n];
        for( i=0; i<n; i++)
            tab[i] = b.tab[i];
    }
    tab[i]='\0';
    majstat();
    return *this;

}
```

    b. L'opérateur + : il devra permettre de gérer la concaténation de deux chaînes dans une troisième => s3=s1+s2

L'opérateur réalise une concaténation de st1 et st2 dans res.

La première entité st1 est affecté puis on ajoute st2 à la suite en utilisant la méthode concaténation.

```
MyString operator+(MyString st1, MyString st2)
{
    MyString res = st1;
    res.concatenation(st2);
    return res ;
}
```

    c. L'opérateur de flux de sortie pour permettre d'afficher des objets Mystring avec le cout

Dans un objet MyString on affecte la valeur de notre objet puis on retourne notre objet pour l'affichage.

```
ostream& operator<<(ostream& os, const MyString& st)
{
    os<< st.tab;
    return os;
}
```

3. **Ecrire un programme principal permettant d'instancier et de manipuler plusieurs objets « string »**

```cpp
int main()
{
    cout<<"s:"<<endl;
    MyString s; s.affiche();//Initialisation d'un objet MyString vierge
    cout<<"S1:"<<endl;
    MyString s1("aac"); s1.affiche();//Initialisation d'un objet MyString avec définition de la chaine
    cout<<"S2:"<<endl;
    MyString s2(s1); s2.affiche();//Initialisation d'un objet MyString via constructeur de recopie
    cout<<"S3:"<<endl;
    MyString s3('z',4); s3.affiche();
    //Initialisation d'un objet MyString comprenant 4 fois le caractère z

    s3.concatenation(s1); //ajout de la chaine s1 dans s3
    s3.affiche();
    s1.dedouble('c'); // Doublage du caractère c dans s1
    cout<< "S1 : " << endl;
    s1.affiche();
    s1.supprimer_un_carac('c');//Suppression des caractères c dans s1
    s1.affiche();

    MyString s4("abcdefghijklmnopqrtusvwxyz");
    s4.supprimer_un_carac('a');s4.supprimer_un_carac('i');
    cout << "S4 : " << endl;
    s4.affiche();
    s4.dedouble('e');
    s4.affiche();
    //vérifications de l'application de plusieurs méthodes a la suite les unes des autres

    s4 = s2; // Vérication de la surcharge d'opérateur =
    s4.affiche();

    s3.GetTab();s3.affiche();
    cout << "Operateur +" <<endl;
    cout << s4 << endl;// Vérication de la surcharge d'opérateur <<
    MyString s5;
    s5 = s2+s2; // Vérication de la surcharge d'opérateur +
    cout << s5 <<endl;

}
```