Problema 3: Especificació i implementació de la classe `String` (proposta de solució

Esteve Brugulat Josep M. Ribó

26 d'octubre de 2009

1 Objectius

- Continuar treballant amb apuntadors, memòria dinàmica i referències (1.6.2, 1.6.3, 1.6.4, apèndix A)
- Continuar treballant amb constructors i destructors (1.7, 1.8)
- Introduir la sobrecàrrega d'operadors (1.9)

Per fer aquest problema, cal que us llegiu:
- Els apunts (apartats 1.1-1.9)
- El document titulat: *Manaments de disseny de classes*

2 Especificació de la classe `MyString`

La classe `MyString` està dissenyada per tal de modelitzar cadenes de caràcters.

1. Quin fitxer usaràs per tal d'especificar la classe `MyString`?

   `MyString.txt`

2. Quines operacions són adients per la classe `MyString`? **(DOCUMENT: Manaments de disseny de classes)**.

   Pensa en quines operacions constructores, modificadores i consultores tenen sentit per aquesta classe i parametritza-les correctament.

   Exemples: Obtenir el caràcter de la cadena que ocupa la posició `i`. Obtenir la longitud de la cadena, comparar dues cadenes, concatenar-les....

   Nota important: Afegeix la sobrecàrrega dels operadors `=, ==, <=, +, <<` com a operacions de la classe (usa l'operador `+` per l'operació de concatenació de dos cadenes) **(APUNTS: 1.9)**

   - **Constructores:**

     ```
     MyString();
     MyString(char st[]);
     MyString(const MyString&);
     ```

   - **Modificadores:**

```
void insertChar(char c, unsigned int pos, bool& err);
void appendChar(char c);
void operator=(const MyString& c);
void concat(char* c);
void get(istream& c,char test);
```

- **Consultores:**

```
char getIthChar(unsigned int pos, bool& err) const;
unsigned int getLength() const;
unsigned int substring(const MyString& c);
void toMyString(char c[]);
bool operator==(const MyString& c);
bool operator<=(const MyString& c);
MyString& operator+(const MyString& c);

friend ostream& operator<<(ostream& c, const MyString& ca);
```

Aquestes operacions (la majoria) es troben especificades més avall.

3. Escriu capceleres alternatives pels operadors `==` i `+` **(APUNTS: 1.9.4)**

   - `friend bool operator==(const MyString& c, const MyString& d);`
     Notar que ara és una funció friend (no és, pròpiament, una operació de la classe) i, per tant, necessita que li passem per paràmetre els dos strings que cal comparar.
   - `friend MyString& operator+(const MyString& c, const MyString& d);`
     Mateix comentari que abans.

4. Especifica totes les operacions anteriors **(APUNTS: 1.4; DOCUMENT: Manaments de disseny de classes)**.

   - `MyString();` (constructora per defecte)
     - **Crida:** `MyString st;`
     - **Pre:** void
     - **Post:** `st` és una cadena de caràcters buida

     Notem que no fem cap hipòtesi de com estarà representada internament la cadena (per exemple: no diem si la cadena de caràcters buida estarà representada internament mitjançant una marca de final com ara `'\0'` o bé amb l'ajut d'un natural que indicarà la longitud de la cadena i que valdrà 0 en el cas de la cadena buida). **No ho diem. El client no ho sap... ni li interessa**.
   - `MyString(char c[]);`
     - **Crida:** `MyString st(c);`
     - **Pre:** `c` és un array de caràcters acabat en `'\0'`
     - **Post:** `st` té els mateixos caràcters que `c`

     Aquesta operació permet convertir un array de caràcters acabat en `'\0'` en un objecte de la classe `MyString`. Tinguem-ho clar: **un objecte de la classe MyString no és un array de caràcters ni el podem tractar com a tal** (potser internament està representat com a array de caràcters, però això el client no ho sap).

     Si `st` ha sigut declarat com a objecte de la classe `MyString` (`MyString st;`), fóra un error molt gran introduir en un programa una instrucció de l'estil:

     `strcpy(st,"popo");`

     L'error prové del fet que `strcpy` espera dos arrays de caràcters, però `st` no ho és.

     Igualment fóra un error fer:

```
char s[10];

s.getLength();  //ERROR!!! s no es un objecte
                //de la classe MyString
```

- `MyString(const MyString&);` (Constructora copiadora)
  - **Crida:** `MyString st(st2);`
  - **Pre:** void
  - **Post:** `st` ha sigut inicialitzada amb els caràcters de la cadena `st2`
- `void insertChar(unsigned int, char, bool& );`
  - **Crida:** `st.putChar(pos,c,err);`
  - **Pre:** void
  - **Post:** `st` és la mateixa cadena que `st'` a la que s'ha afegit el caràcter `c` a la posició `pos`. Els caràcters successius s'han desplaçat una posició a la dreta. `err=fals`. Si `pos` és més gran o igual que la longitud de `st'`, aleshores `err=cert` i `st=st'`
- `void appendChar( char);`
  - **Crida:** `st.putChar(c);`
  - **Pre:** void
  - **Post:** `st` és la mateixa cadena que `st'` a la que s'ha afegit el caràcter `c` al final.
- `MyString operator+(const MyString&);`
  - **Crida:** `st3=st+st2;`
  - **Pre:** void
  - **Post:** `st3` conté la concatenació de les cadenes `st` i `st2`. Ni `st` ni `st2` es modifiquen.

  Conceptualment aquesta crida equival a fer:
  `st3=st.operator+(st2);`
  Però no s'ha de fer la crida d'aquesta manera sinó en forma d'operador infix: `st3=st+st2;`
  void concat(char* c);
  - **Crida:** `st.concat(c);`
  - **Pre:** `c` és un array de caràcters acabat en `\0`
  - **Post:** `st` conté la concatenació de `st'` amb els caràcters de l'array `c`.
- `char getIthChar(unsigned int, bool&) const;`
  - **Crida:** `c=st.getIthChar(pos,err);`
  - **Pre:** void
  - **Post:** `c` és el caràcter que ocupa la posició `pos` de la cadena `st` i `err=fals`. Si `pos` és més gran o igual que la longitud de `st` `err=cert` i `c` és indefinit.
- `unsigned int getLength() const;`
  - **Crida:** `lon=st.getLength()`
  - **Pre:** void
  - **Post:** `lon` és la longitud de la cadena `st`
- `bool operator==(const MyString&) const;`
  - **Crida:** `b=(st==st2)`
  - **Pre:**
  - **Post:** `b` és cert si `st` i `st2` tenen els mateixos caràcters i en el mateix ordre i fals altrament.
- `void operator=(const MyString&);`

  - **Crida:** `st=st2;`
  - **Pre:**
  - **Post:** `st` conté una còpia de `st2`
- `bool operator<=(const MyString& st);`
  - **Crida:** `b=(st<=st2);`
  - **Pre:**
  - **Post:** `b` és cert si `st` és lexicogràficament menor o igual que `st2`
- `friend ostream& operator<<(ostream& c, const MyString& ca);`

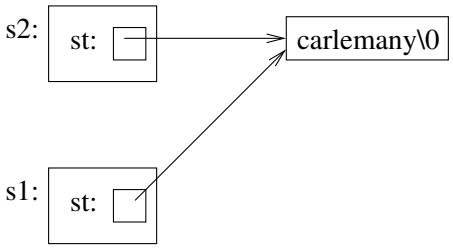
  ...
- `void get(istream& c,char test);`
  - **Crida:** `st.get(c,test);`
  - **Pre:**
  - **Post:** `st` és l'objecte de la classe `MyString` que s'obté de la lectura dels propers caràcters de l'entrada estàndar fins llegir el caràcter `test`. El caràcter `test` no s'inclou a `st`.

5. L'especificació que has fet, pot contenir alguna referència a la representació de la classe? **(APUNTS: 1.4; DOCUMENT: Manaments de disseny de classes)**.

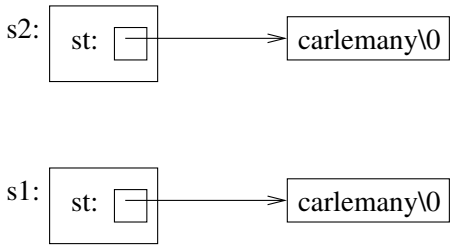
   No, no i mil cops no. *El client no ha de saber com està representada internament un objecte d'una classe ni com estan implementades les seves operacions.*

6. Què passaria si no haguessis especificat (i, posteriorment, no implementessis) l'operador d'assignació (`=`)? Es podrien fer coses de l'estil `s1=s2;`? (se suposa que `s1` i `s2` són dos objectes de la clase `MyString`) **(APUNTS:1.9.3, 1.11.5)**

   Si que es podria fer perquè el compilador de C++ proporciona per defecte una implementació de l'operador d'assignació (=). Però aquesta implementació per defecte només fa una còpia superficial de l'objecte (vegeu apunts, 1.11.5). Si volem fer una còpia profunda hem de proporcionar nosaltres una implementació per aquest operador (sobrecarregar-lo). La figura mostra la diferència entre la còpia superficial proporcionada per l'operador = definit per defecte pel compilador i el que proposem nosaltres.



s1=s2;
(usant l'operador = per defecte de C++)



s1=s2;
(usant l'operador = sobrecarregat pel programador)

3 Representació de la classe `MyString`

1. Quins atributs haurá de contenir aquesta classe considerant que volem fer la representació de manera que la cadena de caràcters s'emmagatzemi en memòria dinàmica?

   ```
   char* st;
   ```

   L'espai per encabir la cadena es reservarà en temps d'execució en memòria dinàmica.

2. En aquesta representació, com determines el final de la cadena?

   Tenim dues opcions:

   (a) *Mitjançant un natural lenString que indica el nombre de caràcters que conté:*
   ```
   char* st;
   unsigned int lenString;
   ```
   (b) *Mitjançant una marca de final*
   En aquest cas no cal afegir cap nou atribut a la representació

   Triarem la representació de la cadena amb marca de final i usarem `'\0'` *com a marca de final per tal de poder aprofitar algunes operacions de la biblioteca* `cstring` *(i.e., strcpy, strcmp, strlen...; aquestes operacions confien en què l'array de caràcters acaba en* `'\0'`*)*

3. L'usuari de la classe `MyString`, ha de conèixer com està aquesta representada per tal de poder-la usar? **(APUNTS: 1.2.2)**

   En particular, li cal conèixer si, en la seva representació interna, un objecte de la classe `MyString` acaba en `'\0'` (si és que s'ha triat aquesta representació)?

   No, no i no. *L'usuari no necessitarà saber en cap moment si un objecte de la classe* `MyString` *acaba o no en una marca de final i quina és aquesta marca de final, si se n'usa cap. L'usuari es limitarà a usar les operacions tal i com estan especificades i cap d'elles parla de cap marca de final.*

4. Escriu el contingut del fitxer que contindrà la representació de la classe `MyString`

```
#ifndef CADENA_H
#define CADENA_H

#include <iostream>

using namespace std;

class MyString{

  char* st;

public:

  MyString();
  ~MyString();
  MyString(char st[]);
  MyString(const MyString&);

  char getIthChar(unsigned int pos, bool& err) const;
  void insertChar(char c, unsigned int pos, bool& err);
  void appendChar(char c);
  unsigned int getLength() const;
  unsigned int substring(const MyString& c);
  void toString(char c[]);
  bool operator==(const MyString& c);
  void operator=(const MyString& c);
  bool operator<=(const MyString& c);
  MyString& operator+(const MyString& c);
  void concat(char* c);

  friend ostream& operator<<(ostream& c, const MyString& ca);
  void get(istream& c,char test);
```

```
};

#endif
```

4 Implementació de les operacions de la classe MyString

1. En quin fitxer hauràs de posar aquesta implementació? **(APUNTS: 1.5.4)**

   MyString.cc

2. Implementa les operacions de la classe MyString

```
#include "MyString.h"
#include <iostream>
#include <cstring>

using namespace std;

MyString::MyString()
{
  st=new char[1];
  st[0]='\0';
}

MyString::MyString(char pst[])
{
  st=new char[strlen(pst)+1];
  strcpy(st,pst);
}

MyString::MyString(const MyString& c)
{
  int i=0;

  st=new char[c.getLength()+1];
  for(i=0;i<=c.getLength();i++){
   st[i]=c.st[i];
  }
  //aquest bucle s'hagues pogut substituir per:
  //strcpy(st,c.st);

}

MyString::~MyString()
{
  delete [] st;
}

char MyString::getIthChar(unsigned int pos, bool& err) const
{
  if (pos<getLength()){
    err=false;
    return st[pos];
  }
  else{
    err=true;
```

```
    return '\0';
  }
}


void MyString::insertChar(char c, unsigned int pos, bool& err)
{
  char* aux;
  int len=getLength();
  int i;

  if (pos<=len){
    err=false;

    aux=new char[len+2];

    for (i=0;i<pos;i++){
      aux[i]=st[i];

    }
    aux[pos]=c;
    for (i=pos+1;i<=len;i++){
      aux[i]=st[i-1];
    }
    aux[len+1]='\0';
    delete [] st;
    st=aux;
  }
  else{
    err=true;
  }

}

void MyString::appendChar(char c)
{
  bool err;

  insertChar(c,getLength(),err);
}



void MyString::toString(char c[])
{
  int i=0;

  delete [] st;
  st=new char[strlen(c)+1];
  strcpy(st,c);
}

unsigned int MyString::getLength() const
{
  return strlen(st);
}

unsigned int MyString::substring(const MyString& c)
```

```
{

///EXERCICE: IMPLEMENT IT
  return 0;
}

bool MyString::operator==(const MyString& c)
{
  int i=0;
  int l;
  l=getLength();
  if (c.getLength()!=l) return false;
  else{
   while (c.st[i]==st[i] && i<l){
    i=i+1;
   }
   return (c.st[i]==st[i]);
  }
}


void MyString::operator=(const MyString& c)
{

  int i=0;

  delete[] st;

  st=new char[c.getLength()+1];
  strcpy(st,c.st);
}

bool MyString::operator<=(const MyString& c)
{
  int i=0;
  while (st[i]==c.st[i] && st[i]!='\0'){

    i++;
  }

  return (st[i]<=c.st[i]);
}

MyString& MyString::operator+(const MyString& c)
{
  int i,j,l1,l2;

  MyString* aux;

  aux=new MyString;

  l1=getLength();
  l2=c.getLength();
  aux->st=new char[l1+l2+1];
  for (i=0;i<l1;i++){
    aux->st[i]=st[i];
  }
  for (j=0;j<l2;j++){
```

```
    aux->st[i]=c.st[j];
    i++;
  }
  aux->st[i]='\0';

  return *aux;
}

void MyString::concat(char* c)
{
  int i,j,l1,l2;

  char* aux;

  aux=new char[getLength()+strlen(c)+1];
  i=0;j=0;
  while (st[i]!='\0'){
    aux[i]=st[i];
    i++;
  }
  while(c[j]!='\0'){
    aux[i]=c[j];
    j++; i++;
  }
  aux[i]='\0';
  delete [] st;
  st=aux;
}


ostream& operator<<(ostream& c, const MyString& ca)
{

  c<<ca.st;

  return c;
}


void MyString::get(istream& c, char test)
{
  const int NCAR=10;
  int i;
  bool end=false;
  char aux[NCAR+1];

  delete [] st;
  st=new char[1]; st[0]='\0';

   while (!end){
     c.get(aux[0]);
     i=0;
     while (aux[i]!=test && i<NCAR-1){
       i++;
       c.get(aux[i]);
     }

     if (aux[i]==test){
```

```
        aux[i]='\0';
        end=true;
      }
      else aux[i+1]='\0';

      this->concat(aux);
    }
}
```

5 Ús de la classe MyString

1. En quin fitxer has de posar un programa que usi la classe MyString? **(APUNTS: 1.5.4, 1.5.5)**
   userstring.cc (per exemple)
2. Quins fitxers has d'incloure per tal de poder usar la classe MyString? **(APUNTS: 1.5.4)**
   MyString.h
3. Implementa un petit programa que usi la classe MyString **(APUNTS: 1.5.4)**

```
#include "MyString.h"


int main()
{
  MyString ss1, ss3;
  MyString ss2("kjkj");

  ss3.get(cin,'\n');
  MyString ss4(ss2);
  ss1=ss2+ss3;

  cout<<ss1<<endl;
  cout<<"ss4="<<ss4<<endl;
  if (ss1<=ss2) cout<<"error"<<endl;
  else cout<<"ok"<<endl;

  if (ss1<=ss1) cout<<"ok"<<endl;
  else cout<<"error"<<endl;

  if (ss2<=ss1) cout<<"ok"<<endl;
  else cout<<"error"<<endl;

  if (ss1==ss1) cout<<"ok"<<endl;
  else cout<<"error"<<endl;

  if (ss1==ss3) cout<<"error"<<endl;
  else cout<<"ok"<<endl;

  if (ss3==ss1) cout<<"error"<<endl;
  else cout<<"ok"<<endl;

  bool err;
  cout<<ss3<<endl;
  cout<<ss3.getIthChar(4,err);
```

```
  return 0;
}
```

4. Què cal fer per tal d'executar aquest programa? **(APUNTS: 1.5.4, document compilació separada)**

```
$g++ -c MyString.cc
$g++ -c userstring.cc
$g++ userstring.o MyString.o -o userstring
```

Aquestes instruccions generen un fitxer directament executable anomenat `userstring`.