Definició de classes: La cadena de caràcters

Josep Maria Ribó

Maig-2004

1 La classe Cadena

1.1 El fitxer "cadena.txt"

```
#ifndef CADENA_H
#define CADENA_H

/*****************************************

CLASSE: Cadena

Classe que implementa les cadenes de caracters.

OPERACIONS:

Cadena();

Cadena c;

Pre:
Post: c es una cadena buida.

*************************************
Cadena(char st[]);

Cadena  c(st);

Pre: st es un vector de caracters acabat en '\0'

Post: Si la longitud de st<=NCAR, c=st
      Si no, c conte els primers NCAR caracters de st.
*************************************
Cadena(const Cadena& ca);

Cadena c(ca);

Pre:
Post: c=ca
*************************************
char posicio(unsigned int pos, bool& err) const;

car=c.posicio(pos,err);

Pre:
```

```
Post: Si 0<=poscar<cad.longitud(), err=fals i
         car es el caracter que ocupa la posicio pos de c
      Si pos>=cad.longitud() err=cert i car es indefinit.

**************************************
unsigned int longitud() const;

n=c.longitud();

Pre:
Post: Retorna el nombre de caracters de c.

**************************************
bool posicioFinal(unsigned int pos) const;

b=c.posicioFinal(pos);

Pre:
Post: b=cert sii pos es  la posicio ocupada pel
      darrer caracter de c.

**************************************
void afegirCarFinal(char car, bool& err);

c.afegirCarFinal(car,err);

Pre:
Post: Si hi ha prou espai a c' per afegir-hi un nou
      caracter, c=c'+{car} (car s'afegeix al final de
      c') i err=fals

     Si no hi ha prou espai: err=cert i c=c'.
**************************************
void buidarCadena();

c.buidarCadena();

Pre:
Post: c es una cadena buida.

**************************************
unsigned int subcadena(const Cadena& c2);

n=c.subcadena(c2);

Pre:
Post: Si c2 esta continguda dins de c, n es l'index de
      c on comenca la primera aparicio de c2.
      Si no, n=-1

**************************************
bool operator==(const Cadena& c2);

b=(c==c2);

Pre:
Post: Si c=c2, b=cert. Altrament b=fals
```

```
**********************************
void operator=(const Cadena& c2);

c=c2;

Pre:
Post: c=c2
**********************************
bool operator<=(const Cadena& c2);

b=c<=c2;

Pre:
Post: b=cert sii c es menor o igual alfabeticament
      que c2
**********************************
void operator+(const Cadena& c2);

c+c2;

Pre:
Post: Si c'.longitud()+c2.longitud()<=NCAR,
      c conte la concatenacio de c' i c2.

      Si no, c' conte els NCAR primers caracters
      de la concatenacio de c' i c2.

**********************************
void llegir(char final);

c.llegir(final);

Pre:
Post: c conte la cadena de caracters que s'ha obtingut per
      l'entrada estandar.

      El caracter que marca el final de la lectura de la
      cadena de l'entrada estandar es final, que no
      s'incorpora a c.

      La cadena contindra com a maxim NCAR caracters.

**********************************
void escriure();

c.escriure();

Pre:
Post: Escriu la cadena c per la sortida estandar.

**********************************

ostream& operator<<(ostream& csort, const Cadena& c);

cout<<c;

Versio estandar de c.escriure();
```

```
*******************************
istream& operator>>(istream& csort, Cadena& c);

cin>>c;

Versio estandar de c.llegir('\n');

************************************************/
```

2 El fitxer “cadena.h”

```
/************************************

Representacio de la classe Cadena.

Llista de les capceleres de les operacions


************************************/


#include <iostream.h>

const int NCAR=50;

class Cadena{

  char cad[NCAR];
  int numcar;

  bool cadenaMenorOIgual(const char c1[],int l1,
                         const char c2[],int l2,
                         int pos);

public:

  Cadena();
  Cadena(char st[]);
  Cadena(const Cadena&);

  char posicio(unsigned int pos, bool& err) const;
  unsigned int longitud() const;
  bool posicioFinal(unsigned int pos) const;
  void afegirCarFinal(char c, bool& err);
  void buidarCadena();
  unsigned int subcadena(const Cadena& c);
  void assignCars(char c[]);
  bool operator==(const Cadena& c);
  void operator=(const Cadena& c);
  bool operator<=(const Cadena& c);
  void operator+(const Cadena& c);
  void llegir(char final);
  void escriure();
```

```
  friend ostream& operator<<(ostream& c, const Cadena& ca);
  friend istream& operator>>(istream& c, Cadena& ca);
};

#endif
```

2.1 El fitxer “cadena.cpp”

```
#include "cadena.h"
#include <iostream.h>

/************************************************

CLASSE Cadena. Implementacio de les operacions
de la classe

Implementada amb el nombre de caracters,
sense la marca de final.


************************************************/



/*****************
OPERACIONS PRIVADES
****************/

bool Cadena::cadenaMenorOIgual(const char c1[],int l1,
                               const char c2[],int l2,
                               int pos)
{

 if (pos==l1) return true;
 else if (pos==l2) return false;
 else if (c1[pos]<c2[pos]) return true;
 else if (c1[pos]==c2[pos]) return
                     cadenaMenorOIgual(c1,l1,c2,l2,pos+1);
 else return false;

}



/*****************
OPERACIONS PUBLIQUES
****************/

Cadena::Cadena()
{
    numcar=0;
}


Cadena::Cadena(char st[])
{
  int i=0;
```

```
  while(st[i]!='\0' && i<NCAR){
   cad[i]=st[i];
   i=i+1;
  }
  numcar=i;
}

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

  numcar=c.numcar;
  for(i=0;i<numcar;i++){
   cad[i]=c.cad[i];
  }
}

char Cadena::posicio(unsigned int pos, bool& err) const
{
  if (pos<numcar){
    err=false;
    return cad[pos];
  }
  else{
    err=true;
    return '\0';
  }
}


void Cadena::assignCars(char st[])
{
  int i=0;

  while(st[i]!='\0' && i<NCAR){
   cad[i]=st[i];
   i=i+1;
  }
  numcar=i;
}


bool Cadena::posicioFinal(unsigned int pos) const
{
  return (pos==numcar-1);
}

void Cadena::buidarCadena()
{
  numcar=0;
}

unsigned int Cadena::longitud() const
{
  return numcar;
}
```

```
void Cadena::afegirCarFinal(char c,bool& err)
{
  if (numcar==NCAR) err=true;
  else {
   err=false;
   cad[numcar]=c;
   numcar=numcar+1;
 }
}

unsigned int Cadena::subcadena(const Cadena& c)
{

///EXERCICI: IMPLEMENTAR-LA

}

bool Cadena::operator==(const Cadena& c)
{
  int i=0;
  if (c.numcar!=numcar) return false;
  else{
   while (c.cad[i]==cad[i] && i<numcar-1){
    i=i+1;
   }
   return (c.cad[i]==cad[i]);
  }
}


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

  int i=0;

  for(i=0;i<c.numcar;i++){
   cad[i]=c.cad[i];
  }
  numcar=c.numcar;

}

bool Cadena::operator<=(const Cadena& c)
{
 return cadenaMenorOIgual(cad,numcar,c.cad,c.numcar,0);
}

void Cadena::operator+(const Cadena& c)
{
  int i,j;

  j=0;

  i=numcar;
  while(i<NCAR && j<c.numcar){
    cad[i]=c.cad[j];
    i=i+1;j=j+1;
  }
```

```
  numcar=i;
}

void Cadena::llegir(char final)
{
   numcar=0;
   cin.get(cad[0]);
  while (cad[numcar]!=final && numcar<NCAR){
   numcar=numcar+1;
   cin.get(cad[numcar]);
  }

}

void Cadena::escriure()
{
  int i=0;
  while (i<numcar){
    cout<<cad[i];
    i=i+1;
  }
  cout<<" ";


}

ostream& operator<<(ostream& c, const Cadena& ca)
{
  int i=0;
  while (i<ca.numcar){
    c<<ca.cad[i];
    i=i+1;
  }
  c<<" ";

  return c;
}


istream& operator>>(istream& c, Cadena& ca)
{
   ca.numcar=0;
   c.get(ca.cad[0]);
  while (ca.cad[ca.numcar]!='\n' && ca.numcar<NCAR){
   ca.numcar=ca.numcar+1;
   c.get(ca.cad[ca.numcar]);
  }

  return c;
}
```

2.2 El fitxer “prova1.cpp”

```
#include "cadena.h"
#include <iostream.h>
```

```
void main()
{


Cadena c1("popopopo");
Cadena c2("popopopo");
Cadena c3("popo");
Cadena c4("");
Cadena c5("");
Cadena c6("");
Cadena c7("");

int i;
bool err;

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

for(i=0;i<10;i++)
{

 c6.llegir();
 c7.llegir();

 c4.afegirCarFinal('q',err);

 cout<<c4<<endl;

 if (c6<=c7) cout<<"primera <= segona\n";

 c6+c7;

 cout<<c6<<c7<<endl;

}
}
```

2.3 El fitxer "Makefile"

Aquest fitxer essencialment executa les comandes següents:

```
$g++ -c prova1.cpp

$g++ -c cadena.cpp

$g++ cadena.o prova1.o -o prova1
```

I genera el fitxer `prova1` que és executable.

El contingut del fitxer `Makefile` és el següent:

```
# Pels RedHat de la Sala d'Usuaris
CC=g++
CPPFLAGS=
CFLAGS= -g
# Per jupiter
ifeq ($(HOSTNAME),jupiter)
```

```
CC=g++
CPPFLAGS= -I/usr/include/g++-include/
CFLAGS=  -g -fhandle-exceptions
endif

all:
make prova1

prova1: prova1.o cadena.o
$(CC) $(CFLAGS) -o prova1 prova1.o cadena.o

prova1.o: prova1.cpp cadena.h
$(CC) $(CPPFLAGS) $(CFLAGS) -c prova1.cpp

cadena.o: cadena.cpp cadena.h
$(CC) $(CPPFLAGS) $(CFLAGS) -c cadena.cpp
```