# Napredno programiranje i programski jezici

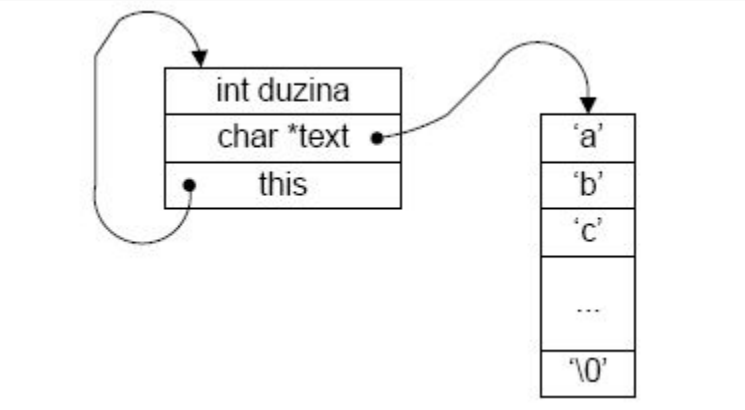
## 04 C++ (preklapanje operatora)

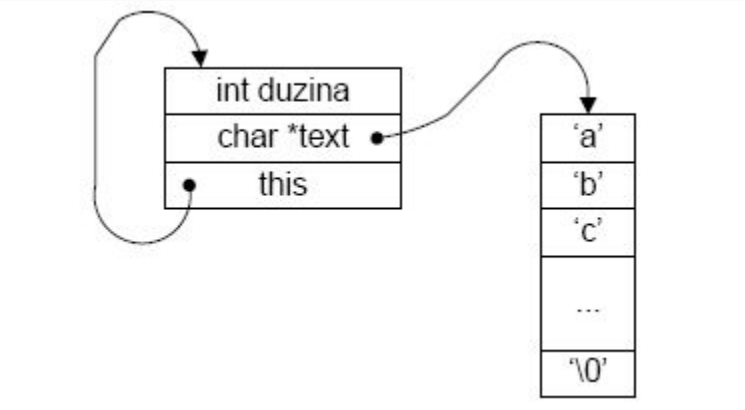
Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

## DinString

Klasa koja obezbeđuje rad sa dinamički kreiranim nizovima karaktera.
Modelira rad sa C-stringovima (null terminated).

*DinString - edukativno, koristiti na kursu*
*ubuduće - uvek koristiti standardnu biblioteku*


int duzina
char *text
this
'a'
'b'
'c'
...
'\0'




c++ biblioteka - tip string, interno skladišti dužinu, string nema null
c nema klase, "string" je niz karaktera (pokazivač na niz), završava se sa null
DinString - klasa, interno pamti dužinu, ima null

## Zašto pamtiti dužinu?

```
class DinString {
private:
 int duzina;
 char *text;
public:
 DinString();
 DinString(const char[]);
 DinString(const DinString&);
 ~DinString();

 int length() const; // return duzina

...
```

**dinstring.hpp**

```
 ...
 char& operator[](int);
 char operator[](int) const;
 DinString& operator=(const DinString&);
 DinString& operator+=(const DinString&);

 friend bool operator==(const DinString&, const DinString&);
 friend bool operator!=(const DinString&, const DinString&);
 friend DinString operator+(const DinString&, const DinString&);
 friend ostream& operator<<(ostream&, const DinString&); };
};
```

```
DinString::DinString() {
  duzina = 0;
  text = NULL;
}
```

```
DinString::DinString(const char ulazniStr[]) {
   ...
}
```

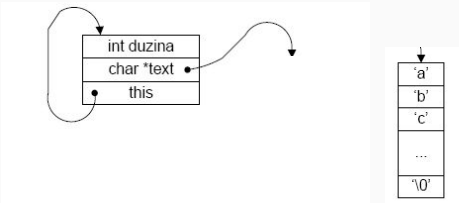
```
DinString::DinString(const DinString &ds) {
   ...
}
```

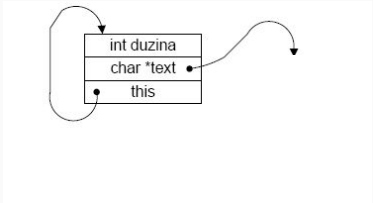
```
DinString::~DinString() {
   delete[] text;
}
```

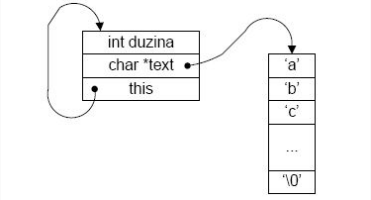
*Kad se, recimo, završi funkcija gde je ds lokalna, zauzeta memorija za niz neće biti oslobođena osim ako je eksplicitno ne oslobodimo u destruktoru*

```
DinString::DinString(const char ulazniStr[]) {
   duzina = 0;

   while(ulazniStr[duzina] != '\0')
      duzina++;

   text = new char[duzina + 1];
   for (int i = 0; i < duzina; i++)
      text[i] = ulazniStr[i];
   text[duzina] = '\0';
}
```

```
DinString::DinString(const DinString &ds) {
   duzina = ds.duzina;

   text = new char[duzina+1];
   for (int i = 0; i < duzina; i++)
      text[i] = ds.text[i];
   text[duzina] = '\0';
}
```



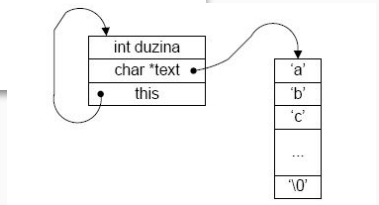



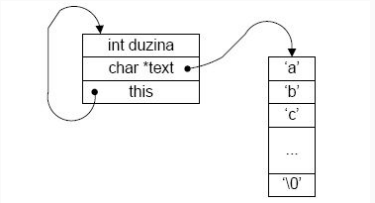
```
char& DinString::operator[](int i) {
   return text[i];
}
```

```
char DinString::operator[](int i) const {
   return text[i];
}
```

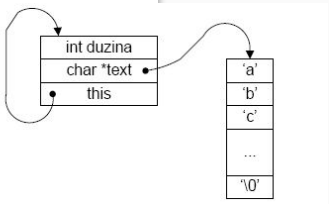
Koji je za čitanje `(ch = ds[3])`, a koji za pisanje `(ds[3] = ‘a’)`?

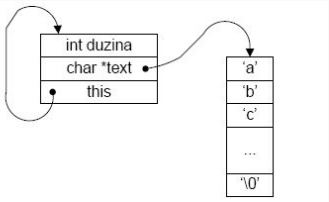
```
DinString& DinString::operator=(const DinString &ds) {
   if (this != &ds){
      delete[] text;

      duzina = ds.duzina;
      text = new char[duzina+1];

      for (int i = 0; i < duzina; i++)
         text[i] = ds.text[i];

      text[duzina] = '\0';
   }

   return *this;
}
```



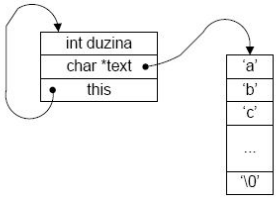
```
DinString& DinString::operator+=(const DinString &ds) {
   int i;
   char *tempText = new char[duzina + ds.duzina + 1];

   // popunjavamo tekst
   for (i = 0; i < duzina; i++)
      tempText[i] = text[i];

   for (i = 0; i < ds.duzina; i++)
      tempText[duzina + i] = ds.text[i];

   tempText[duzina + ds.duzina]='\0';

   // postavljamo duzinu
   duzina += ds.duzina;

   // postavljamo tekst
   delete []text;
   text = tempText;
   return *this;
}
```

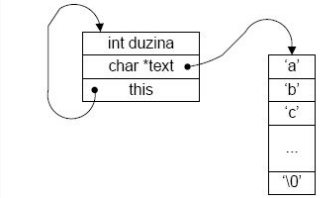


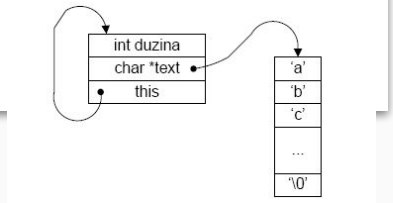
```
bool operator==(const DinString &ds1, const DinString &ds2) {
   if (ds1.duzina != ds2.duzina)
      return false;

   for (int i = 0; i < ds1.duzina; i++)
      if (ds1.text[i] != ds2.text[i])
         return false;

   return true;
}
```

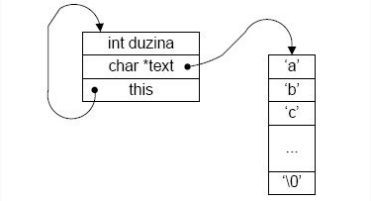
```
bool operator!=(const DinString &ds1, const DinString &ds2) {
   if (ds1.duzina != ds2.duzina)
      return true;

   for (int i = 0; i < ds1.duzina; i++)
      if (ds1.text[i] != ds2.text[i])
         return true;

   return false;
}
```



```
DinString operator+(const DinString &ds1, const DinString &ds2) {
   DinString temp;
   temp.duzina = ds1.duzina + ds2.duzina;
   temp.text = new char[temp.duzina+1];

   int i;
   for(i = 0; i < ds1.duzina; i++)
      temp.text[i] = ds1.text[i];

   for(int j = 0; j < ds2.duzina; j++)
      temp.text[i + j] = ds2.text[j]; // i je ds1.duzina

   temp.text[temp.duzina] = '\0';

   return temp;
}
```



```
ostream& operator<<(ostream &out, const DinString &ds) {
   if(ds.duzina>0)
      out << ds.text;
   return out;
}
```

```
#include "dinstring.hpp"
int main(){
   DinString a, b("Dobar"), c("dan");
   cout << "b: " << b << endl;

   a = b;
   cout << "a: " << a << endl;

   a += c;
   cout << "a: " << a << endl;

   a = b + c;
   cout << "a: " << a << endl;

   char x = a[5];
   cout << "x: " << x << endl;

   a[5] = 'Z';
   cout << "a: " << a << endl;

   cout << "a==b? " << (a == b) << endl;
   cout << "a!=b? " << (a != b) << endl;
   …
}
```

## ZADATAK

*(neobavezno)*

Isprobati sve iz prezentacije.

Implementirati samostalno klasu DinString.
Testirati sve operatore.

Razmisliti i razmotriti šta se izvršava prilikom sledeće naredbe (koje metode se pozivaju) i kako se to dešava.

```
a = “dobar dan”;
```

*Staviti cout << na odgovarajuća mesta (kraj metode ili funkcije).*

Knjiga: primeri za Polinom, Vreme.