

Petar Pjanic - Prezentacija

prvi primer

1. Problem zadatka:

Rešavanje ovog primera podrazumeva skraćenu verziju celog zadatka, koja postavlja grafičke elemente (tekst, slike, blako, novi pasus) u redove zadate veličine. U ovom primeru ćemo, ipak, da se pozabavimo samo elementima oblika reč koji su razdvojeni elementom blanko. Ceo zadatak obuhvata i obradu po stranama.

2. Ideje za rešavanje zadatka:

Po tekstu zadatka treba da se niz znakova (elemenata) postavi u redove ili strane (u zavisnosti koji primer posmatramo). Taj problem možemo da rešimo na više načina. Jedan način je ovaj koji je ponuđen u daljnjem tekstu i koristi klase, nasleđivanje i vektorske nizove. Drugi način je na primer korišćenjem nizova ili matrica koji su trenutno jednostavniji za upotrebu i može se činiti da je to bolje rešenje (naravno ograničavajući se isključivo na prvi primer). Jednostavno bi se reči unosile bez ikakvih klasa u matricu koja bi predstavljala niz redova. Naravno kada se počne sa realizacijom na taj način, dolazi se do komplikovanja samog koda jer nisu u pitanju samo reči koje se ubacuju već i njihov font i veličina (veličina je posebno bitna jer određuje širinu i visinu reda). Naravno i ovaj problem je rešiv ali je bolje rešenje sigurno korišćenje hijerarhije klasa iz više razloga. Prva prednost je što je moguće da se program proširi sa raznim grafičkim elementima i rasporedom po stranama kao što je baš slučaj u drugom primeru bez velikih prepravki postojećeg koda, što nije slučaj sa rešenjem preko matrica. Druga prednost je i što je kod čitkiji i jednostavniji za upotrebu. Bilo bi takođe teško i napraviti bilo kakav različit element osim reči koje bi se postavljale u matricu. Tako da je očigledno zašto se pribegava metodama objektno orijentisanog programiranja, hijerarhija klasa itd. (postoje i druge metode kojima bismo mogli da rešimo zadatak ali one su daleko lošije od našeg rešenja).

Naše rešenje će da se bazira na tome da je moguće da se prepravi kod, tj da se proširi tako da može da prihvati razne zadate grafičke elemente kao što su slike, pasusi i blanko (koji će se implemetirati u drugom primeru). Stavljanje elementa u redove odgovarajuće širine je fleksibilne visine jer visina reda odgovara visini

najvećeg elementa u redu. Naravno, moraju da se i ti tako formirani redovi prelome u strane ali to je deo drugog primera. Mi ćemo se zadržati samo na postavljanju elemenata iz niza u redove i to samo reči. Glavna ideja je u tome da se napravi jedan najširi oblik klase, klasa `Element`, svih ostalih oblika klasa koji će da sadrži visinu i širinu koja je zajednička za sve klase. Dakle, planira se uvođenje sledećih klasa `Element`, `Reč`, `Red` (a za drugi primer još mnogo drugih klasa).

Reč je osnovni element reda i uvođenje klase reč omogućava nam da formiramo najjednostavnije rešenje zadatka. Reč nasleđuje osobine osnovne klase, tj klase `Element`.

Klasa `Red` je klasa koja sadrži niz pokazivača na elemente da bi mogla da prihvati sve elemente bilo kakvi oni bili, slike, pasusi, reči itd. Ovde se primenjuje princip virtuelnog nasleđivanja kojim se obezbeđuje da red prihvati razne elemente jer se radi sa pokazivačima. Da bi ta klasa imala smisla moramo da damo metode za ubacivanje u niz svih elemenata nekog drugog niza. Ta metoda nije jednostavna i razdvaja se na dve, jedna koja će da ubacuje samo jedan element i pravi korekcije širine i visine a druga koja će da ubacuje sve elemente nekog niza. Naravno uočava se mogućnost da se prva metoda iskoristi u drugoj. Tokom elementa u red treba paziti da se ne pređe maksimalna dužina reda.

3. Rešavanje problema:

Neophodno je konstruisati klasu **`Element`**, koja će da obuhvati najširi oblik svih ostalih klasa koje ćemo da implementiramo (njihove zajedničke osobine, visina, širina). Na osnovu prethodnog potrebno je u klasi `Element` da se implementiraju podaci o visini i širini elementa, nezavisno od toga šta on predstavlja (reč, red). Konstruktor klase je obavezan i jednostavan. Podrazumevaju se i metode vraćanja visine i širine (`Sirina()`, `Visina()`).

Neophodno je, da bi se program testirao, da se van klase definišu i dve funkcije koje će da dostavljaju informacije o visini i širini elementa (`lib_SirinaTeksta()`, `lib_VisinaTeksta()`) sa argumentima o tekstu, fontu i visini (implementaciju njihovu ćemo da vidimo na kraju jer postoji mnogo načina njihove imlementacije, a i te funkcije nisu suštinske za naš zadatak). Iz klase `Element` se izvodi nova klasa **`Rec`** koja predstavlja određeni element koji se postavlja u redove. Takođe i klasa **`Red`** će biti izvedena klasa od klase `Element`, ali o tome nešto kasnije.

Klasu Rec treba da definišemo kao klasu naslednicu klase Element a u konačnoj verziji se implementiraju klase Slika, Novi_Pasus i Blanko. Klasa Rec pored širine i visine koji se postavljaju pomoću funkcija lib_SirinaTeksta(), lib_VisinaTeksta() sadrži i privatne podatke, pored nasleđenih, string _Tekst, string _Font i unsigned _VelFonta. Konstruktor klase reč:

```
Rec( string tekst, string font, unsigned velFonta ) :  
    Element( lib_SirinaTeksta( tekst + " ",  
font, velFonta ),  
            lib_VisinaTeksta( tekst + " ", font,  
velFonta ) ),  
        _Tekst( tekst ),  
        _Font( font ),  
        _VelFonta( velFonta )  
{ }
```

Sada dolazimo do glavne klase prvog primera **Red**, koja je takođe naslednica klase Element (ima širinu i visinu). Njen privatni podatak je

```
_Elemeti /* tipa vector<const Element*>_Elementi ; */
```

_Elementi predstavljaju niz pokazivača na elemente koji se tim putem ne mogu menjati već im se samo može pristupiti. Drugi privatni podatak je maksimalni širina (_MaxSirina). Konstruktor klase Red će da inicijalizuje samo podatak _MaxSirina.

Postoje dve metode koje se primenjuju u zadatku, a to su Dodaj() i PopuniRed(). Jedna ima zadatak da popuni taj red sa nizom reči, pazeći da se ne pređe _MaxSirina, a druga da doda jednu reč u red.

Dodaj() je osnovna funkcija koja se poziva u funkciji PopuniRed(). Funkcija Dodaj() dodaje element na kraj reda koristeći metod **push_back(e)** pri čemu se širina povećava za širinu elementa a visina postaje veća između visine reda i reči.

```
void Dodaj( const Element* e )  
{
```

```
_Elementi.push_back( e );  
_Sirina += e->Sirina();  
_Visina = max( _Visina, e->Visina() );  
}
```

Metod `PopuniRed()` je glavni metod klase `Red` i on mora da pazi da red ne pređe `_MaxSirina`. Njegov argument je niz elemenata koji se ubacuje u red, takođe postoji i argument koji nam govori od kog mesta se ubacuju reči. Njegova implementacija je:

```
unsigned PopuniRed( const vector<const Element*> el, unsigned prvi )  
{  
    unsigned sledeci = prvi;  
    do Dodaj( el[sledeci++] );  
    while(  
        sledeci < el.size()  
        && el[sledeci]->Sirina() <= _MaxSirina - Sirina()  
    );  
    return sledeci;  
}
```

4. Rešenje celog zadatka:

```
#include <vector>  
#include <string>  
#include <iostream>  
  
using namespace std;  
  
//      "bibliotecke" funkcije  
unsigned lib_SirinaTeksta(    string    tekst,    string  
fontname, unsigned size );  
unsigned lib_VisinaTeksta(    string    tekst,    string  
fontname, unsigned size );  
  
class Element  
{
```

```

public:
    Element( unsigned s, unsigned v )
        : _Sirina(s), _Visina(v)
        {}
    unsigned Sirina() const
        { return _Sirina; }
    unsigned Visina() const
        { return _Visina; }
protected:
    unsigned _Sirina;
    unsigned _Visina;
};

class Rec : public Element
{
public:
    Rec( string tekst, string font, unsigned velFonta )
        : Element(
            lib_SirinaTeksta( tekst + " ", font,
velFonta ),
            lib_VisinaTeksta( tekst + " ", font,
velFonta )
        ),
        _Tekst( tekst ),
        _Font( font ),
        _VelFonta( velFonta )
        {}
private:
    string _Tekst;
    string _Font;
    unsigned _VelFonta;
};

class Red : public Element
{
public:
    Red( unsigned maxSirina )
        : Element( 0, 1 ),
        _MaxSirina( maxSirina )
        {}

```

```

        unsigned PopuniRed( const vector<const Element*> el,
unsigned prvi )
    {
        unsigned sledeci = prvi;
        do Dodaj( el[sledeci++] );
        while(
            sledeci < el.size()
            && el[sledeci]->Sirina() <= _MaxSirina -
Sirina()
            );
        return sledeci;
    }
    void Dodaj( const Element* e )
    {
        _Elementi.push_back( e );
        _Sirina += e->Sirina();
        _Visina = max( _Visina, e->Visina() );
    }
private:
    unsigned                _MaxSirina;
    vector<const Element*>  _Elementi;
};

// probna implementacija
unsigned lib_SirinaTeksta( string tekst, string
fontname, unsigned size )
    { return tekst.length() * size; }
unsigned lib_VisinaTeksta( string tekst, string
fontname, unsigned size )
    { return size * 2; }

main()
{
    vector<const Element*> tekst;
    tekst.push_back( new Rec( "prva", "arial", 10 ));
    tekst.push_back( new Rec( "druga", "arial", 10 ));
    tekst.push_back( new Rec( "treca", "arial", 10 ));
    tekst.push_back( new Rec( "cetvrta", "arial", 10 ));

    Red red(120);

```

```
int sledeci = red.PopuniRed( tekst, 0 );
cout << sledeci << endl;
cout << red.Sirina() << " " << red.Visina() << endl;

Red red2(120);
sledeci = red2.PopuniRed( tekst, sledeci );
cout << sledeci << endl;
cout << red2.Sirina() << " " << red2.Visina() <<
endl;

return 0;
}
```