Projekt iz predmeta *“Uvod u kompjutersku geometriju”*

Student: Edin Smajić

Smjer: Teorijska kompjuterska nauka (TKN)

Godina: Treća (III)

Status: Redovan

Broj indeksa: 5810/M

Tema 7

Predmetni asistent: mr. Admir Beširević

Uvod:

U prilogu sa ovom dokumentacijom poslat je i folder pod imenom “vjezbe2022 - 11. Maj” u kojem se nalaze svi fajlovi neophodni za projekat koji se može pokrenuti u C++ Builder-u.

Između ostalih u projektu se nalaze sljedeći, nama bitni fajlovi:

“**pomocna.h**” - header fajl, u kojem su navedene pomoćne strukture, navedeni njihovi atributi i prototipovi funkcija članica, te su u njemu uključeni neki drugi headeri koji su neophodni za ispravan rad projekta.

“**pomocna.cpp**” - source fajl, u kojem su implementirane sve funkcije članice struktura čiji su prototipovi navedeni u “*pomocna.h*”.

“**Forma.h**” - header fajl, u kojem su navedeni prototipovi svih glavnih funkcija, bez implementacija istih, te su u njemu uključeni neki drugi headeri koji su neophodni za ispravan rad projekta.

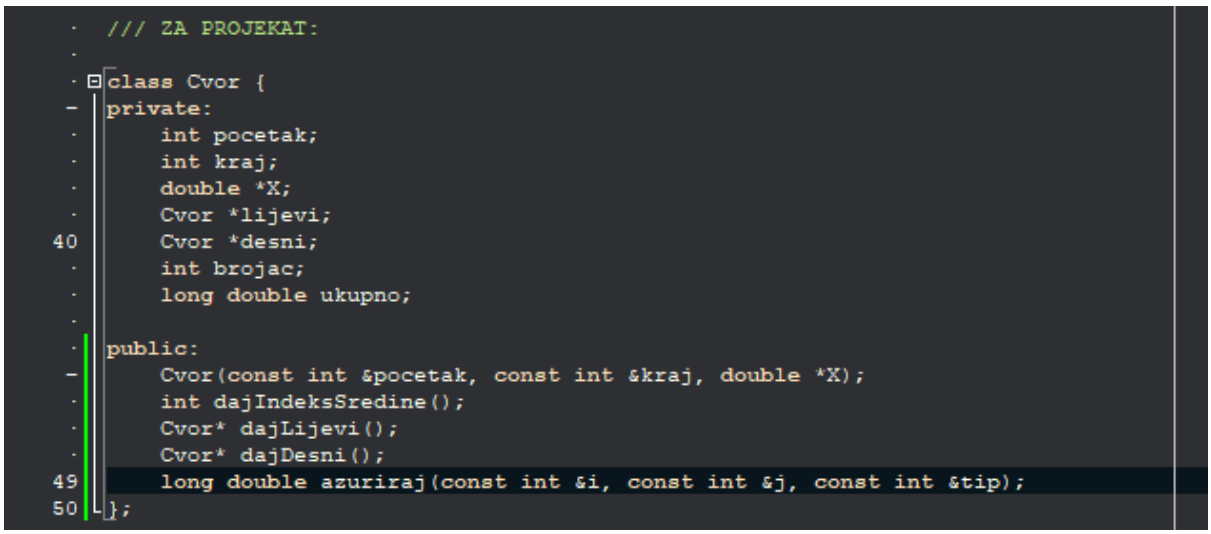
“**Forma.cpp**” - source fajl, u kojem su implementirane sve funkcije čiji su prototipovi navedeni u “*Forma.h*”.

NAPOMENA:

Projekat u kojem sam radio Temu 7 je „najnoviji“ koji sam našao u „Files“ na Teams kanalu „Uvod u kompjutersku geometriju“, kada sam započeo raditi projekat, a to je bio ovaj od 11. Maja:

Također, zbog invertovanog koordinatnog sistema, tj. invertovane y-ose zbog činjenice da su koordinate gornje lijeve tačke forme (0,0), a donje desne (širinaForme, visinaForme), u strukturu Pravougaonik ustvari dodam gornjuLijevu i donjuDesnu tačku u atribute donjaLijeva i gornjaDesna.

U fajlu „**pomocna.h**“ definisane su strukture „Cvor“ i „Pravougaonik“:



```
/// ZA PROJEKAT:

class Cvor {
private:
    int pocetak;
    int kraj;
    double *X;
    Cvor *lijevi;
    Cvor *desni;
    int brojac;
    long double ukupno;

public:
    Cvor(const int &pocetak, const int &kraj, double *X);
    int dajIndeksSredine();
    Cvor* dajLijevi();
    Cvor* dajDesni();
    long double azuriraj(const int &i, const int &j, const int &tip);
};
```

Klasa „Cvor“ ustvari predstavlja Segmentno Stablo, prvi čvor koji se napravi je korijen, a zatim preko funkcija dajLijevi() i dajDesni() prave se i lijevi i desni potomak tog čvora, a ovo se dalje radi i nad ovim potomcima...

Atributi jedne instance klase „Cvor“ su:

int pocetak; - indeks početka intervala ovog čvora u nizu X

int kraj; - indeks kraja intervala ovog čvora u nizu X

double *X; - predstavlja pokazivač na niz x koordinata tačaka pravougaonika čiju površinu tražimo (sortiran u neopadajućem poretku)

Cvor *lijevi; - pokazivač na lijevi potomak ovog intervala

Cvor *desni; - pokazivač na desni potomak ovog intervala

int brojac; - broj pravougaonika koje siječe sweep linija u intervalu [X[pocetak], X[kraj]]

long double ukupno; - ukupna dužina svih intervala „ispod“ ovog u stablu, čiji je brojač pozitivan (dakle po x-u)

```
struct Pravougaonik {
    Tacka donjaLijeva, gornjaDesna;
    Pravougaonik(Tacka A, Tacka B) : donjaLijeva(A), gornjaDesna(B) {
        if(B.x < A.x){
            swap(donjaLijeva, gornjaDesna);
        }
    }
    void Crtaj(TImage*);
};
```

Struktura „Pravougaonik“ sadrži atribute „donjaLijeva“ i „gornjaDesna“, koji su tipa „Tacka“ i one jednoznačno određuju položaj pravougaonika u formi.

Konstruktor ove strukture, donjuLijevu tačku postavlja na A ili B u ovisnosti od toga koja od te dvije je manja po x-u.

Definisan je prototip funkcije Crtaj(TImage*), koji je implementiran u fajlu „**pomocna.cpp**“.

U fajlu „**pomocna.cpp**“ implementirane klasa „Cvor“ i struktura „Pravougaonik“, tj. njihove funkcije članice.

Što se tiče klase „Cvor“:

Konstruktor, funkcije dajIndeksSredine(), dajLijevi(), dajDesni() su self-explainatory.

Dok će funkcija

long double azuriraj(const int &i, const int &j, const int &tip);

biti objašnjena u glavnoj funkciji za površinu unije pravougaonika.

Funkcija za iscrtavanje pravougaonika:

Kako već imamo dvije tačke pravougaonika, „donjaLijevu“ tj. donju lijevu, te „gornjaDesna“ tj. gornju desnu, nađemo na osnovu njih donju desnu to je „donjaDesna“ i gornju lijevu to je „gornjaLijeva“ i povućemo 4 linije između odgovarajućih parova tačaka.

```
void Pravougaonik::Crtaj(TImage* slika) {
200
    Tacka donjaDesna(gornjaDesna.x, donjaLijeva.y);
    Tacka gornjaLijeva(donjaLijeva.x, gornjaDesna.y);

    slika->Canvas->MoveTo(donjaLijeva.x, donjaLijeva.y);

    slika->Canvas->LineTo(donjaDesna.x, donjaDesna.y);
    slika->Canvas->LineTo(gornjaDesna.x, gornjaDesna.y);
    slika->Canvas->LineTo(gornjaLijeva.x, gornjaLijeva.y);
    slika->Canvas->LineTo(donjaLijeva.x, donjaLijeva.y);
210
}
```

U fajlu „**Forma.h**“, u kojem je definisana klasa TForm4 koja nasljeđuje klasu TForm, se nalaze njeni sljedeće IDE-managed komponente neophodne za ovaj projekat:

```
30
    /// ZA PROJEKAT:

    TButton *povrsinaUnijeDugme;
    TRadioButton *DodavanjePravougaonika;
```

Dva dugmeta:

1) Prvo dugme „povrsinaUnijeDugme“ je obično dugme forme sa tekstom „Površina unije“, a čijim se klikom pokreće funkcija ispod, a njena implementacija je izvršena u fajlu „Forma.cpp“.

```
    void __fastcall povrsinaUnijeDugmeClick(TObject *Sender);
```

   Dakle ovo iznad je prototip **glavne** funkcije projekta koja je implementirana u drugom fajlu.

2) Drugo dugme „DodavanjePravougaonika“ je radio dugme, koje kad je čekovano (eng. checked) omogućeno je dodavanje pravougaonika klikanjem na formu, to dodavanje je objašnjeno ispod:

   Atributi klase TForm4 potrebni za ovaj projekat su:

```
    /// ZA PROJEKAT:

    vector<Pravougaonik> pravougaonici;
    bool dodajemDonjuLijevu = true;
    Tacka donjaLijeva;
    Tacka gornjaDesna;
```

   U vektoru pravougaonika „pravougaonici“ čuvam pravougaonike nad kojim će se pokrenuti algoritam za računanje površine, koristeći ova tri atributa ispod (dodajemDonjuLijevu, donjaLijeva i gornjaDesna) sam ustvari omogućio klikanjem dva puta na panel ubaciti novi pravougaonik u vektor „pravougaonici“. Prvi klik mora biti donja lijeva tačka pravougaonika, dok drugi klik gornja desna tačka pravougaonika. Omogućeno je i obratno tj. ukoliko je drugi klik „lijevo ispod“ od prvog

klika i dalje će se u vektor „pravougaonici“ ubaciti odgovarajuće tačke za donjuLijevu i gornjuDesnu u atribute klase Pravougaonik.

Opisano iznad radim na dnu prve funkcije fajla „**Forma.cpp**“:

*void __fastcall SlikaMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift,*
int X, int Y);

```
    else if(DodavanjePravougaonika->Checked) {
        if(dodajemDonjuLijevu){
            donjaLijeva = Tacka(X,Y);
            donjaLijeva.Crtaj(Slika);
            dodajemDonjuLijevu = false;
        }
        else{
            gornjaDesna = Tacka(X,Y);
            gornjaDesna.Crtaj(Slika);

            Tacka donjaDesna(gornjaDesna.x, donjaLijeva.y);
            Tacka gornjaLijeva(donjaLijeva.x, gornjaDesna.y);

            Pravougaonik pravougaonik(gornjaLijeva, donjaDesna);
            pravougaonici.push_back(pravougaonik);
            pravougaonik.Crtaj(Slika);

            dodajemDonjuLijevu = true;
        }
    }
```

Osim navedenog u fajlu „Forma.cpp“ se nalazi **glavna** funkcija ove teme, a to je:

void __fastcall povrsinaUnijeDugmeClick(TObject *Sender);

- Ova funkcija računa površinu unije pravougaonika koji se nalaze u vektoru „pravougaonici“ i to u vremenu O(nlogn)

Opis funkcije:

vector<vector<double>> dogadjaji – vektor sa 2n redova, gdje je n broj pravougaonika u vektoru „pravougaonici“, pri čemu svaki red ima 4 kolone

Jedan događaj je jedna tačka strukture „Pravougaonik“, znači da za svaki pravougaonik imamo 2 bitna događaja, donju lijevu tačku i gornju desnu tačku.

X - Pokazivač na niz, sa 2n elemenata čiji su elementi x koordinate donjihLijevih i gornjihDesnih tačaka svih pravougaonika iz vektora „pravougaonici“ (n – broj pravougaonika iz vektora „pravougaonici“)

Prolazimo kroz sve pravougaonike vektora „pravougaonici“ i dodajemo za svaki pravougaonik dva nova reda:

Prvi red kao nultu kolonu ima y koordinatu tačke „donjaLijeva“ pravougaonika, druga kolona je setovana na 1 jer se radi o donjoj lijevoj tački pravougaonika, treća kolona čuva x koordinatu tačke „donjaLijeva“ ovog pravougaonika, a četvrta kolona čuva x koordinatu tačke „gornjaDesna“ ovog pravouganika.

Drugi red kao nultu kolonu ima y koordinatu tačke „gornjaDesna“ pravougaonika, druga kolona je setovana na -1 jer se radi o gornjoj desnoj tački pravougaonika, treća kolona čuva x koordinatu tačke „donjaLijeva“ ovog pravougaonika, a četvrta kolona čuva x koordinatu tačke „gornjaDesna“ ovog pravouganika.

U niz X dodajemo x koordinatu donjeLijeve tačke.

U niz X dodajemo x koordinatu gornjeDesne tačke.

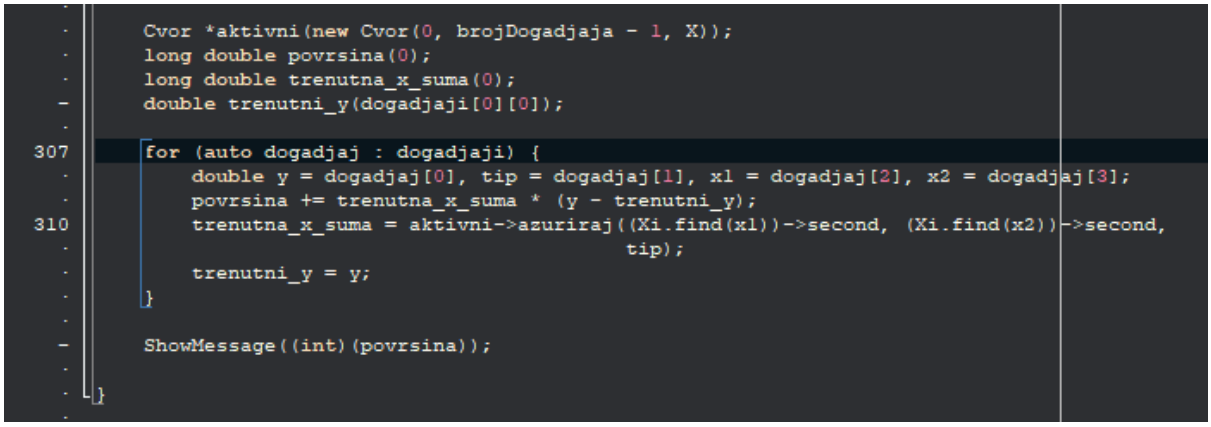
Zatim algoritam nastavlja tako što se sortira vektor vektora „dogadjaji“ po nultoj koloni u neopadajućem poretku, tj. sortiramo dogadjaje po y-koordinatama tih događaja.

Sortiramo i niz X u neopadajućem poretku.

Zatim ćemo svaki element iz niza X mapirati tj.

unordered_map<double, int> Xi - Heš mapa bez duplikata

U ovoj heš mapi Xi, čuvamo za svaku x koordinatu iz niza X-eva, njen indeks u nizu X. Dakle ključ je x-koordinata, a vrijednost je indeks ove koordinate u nizu X. Ovo je potrebno kako bi adekvatno mogli ažurirati strukturu Čvor, tako da nam ne uzima dodatno vremena ova operacija. U ovom slučaju ona je konstantna.



Zatim površinu unije pravougaonika setujemo na 0, kao i trenutnu_x_sumu.

Trenutni_y uzima na početku vrijednost polja nulte kolone prvog događaja, tj. to je y koordinata tačke najmanje tačke (opet po y-onu).

U Segmentnom stablu „aktivni“ čuvamo u korijenima elementarne x intervale.

Algoritam dalje pokreće horizontalni Sweep Line tj. Sweep liniju odozdo ka gore po y-koordinatama, u našem slučaju, kako je y-osa invertovana, odozgo ka dole ta zamišljena sweep linija ide. Jedan događaj je ustvari y koordinata gornje desne ili donje lijeve tačke pravouganika. (a prethodno smo ih sortirali u neopadajućem poretku po y koordinati). Druga kolona je tip događaja, a ustvari tip je jednak 1 ako tek nailazimo na taj pravougaonik (ako naletimo na donjuLijevu tačku), u suprotnom ako „napuštamo“ taj pravougaonik onda je tip jednak -1 (ako naletimo na gornjuDesnu tačku). Ovo je vrlo bitno za funkciju ažuriraj klase „Cvor“ naše instance „aktivni“.

- Tek kada prvi put naiđemo na tačku površina ne treba da se poveća, jer tek ulazimo u prvi pravougaonik zato izraz:
  povrsina += trenutna_x_suma * (y - trenutni_y);
  u prvoj iteraciji daje 0.

- Izraz:
  trenutna_x_suma = aktivni->azuriraj((Xi.find(x1))->second, (Xi.find(x2))->second, tip);
  vrši ažuriranje intervala [x1, x2] na način da će čvoru koji odgovara ovom intervalu uraditi sljedeće:

```
long double Cvor::azuriraj(const int &i, const int &j, const int &tip) {
    if (i >= j)
        return 0;
    if (pocetak == i && kraj == j)
        brojac += tip;
    else {
        dajLijevi()->azuriraj(i, min(dajIndeksSredine(), j), tip);
        dajDesni()->azuriraj(max(dajIndeksSredine(), i), j, tip);
    }

    if (brojac > 0)
        ukupno = X[kraj] - X[pocetak];
    else
        ukupno = dajLijevi()->ukupno + dajDesni()->ukupno;

    return ukupno;
}
```

Prisjetimo se:
int brojac; - broj pravougaonika koje siječe sweep linija u intervalu [X[pocetak], X[kraj]]
long double ukupno; - ukupna dužina svih intervala „ispod“ ovog u stablu, čiji je brojač pozitivan (dakle po x-u)

Ako je i>=j ne radimo ništa (jer ne možemo imati takav interval [x[i],x[j]], gdje je x[i]>x[j].

U slučaju da se indeks od x1 (tj. i) poklapa sa početkom posmatranog čvora i indeks od x2 (tj. j) poklapa sa krajem posmatranog čvora segmentnog stabla, brojac ćemo povećati za tip. U slučaju da je tip 1, povećavamo brojac, u slučaju da je tip -1, i dalje povećavamo ali za -1 tj. smanjujemo brojac. Logično je ako napuštamo pravougaonik, da treba u odgovarajućem intervalu smanjiti broj takvih pravougaonika za 1, u suprotnom povećati za 1. Dok istovremeno, ukoliko tek izvršavanjem ove fje povećavamo brojac za 1 ili ako smo u čvoru (intervalu) u kojem je brojac veći od 0, ukupno setujemo na X[kraj]-X[pocetak], inače ako je brojac 0, ukupno će biti zbir ukupnih dužina x intervala njegovih potomaka.
Na ovaj način u svakom trenutku znamo dužinu (u širinu, jer je po x-u) pravougaonika koje trenutno siječe sweep linija.

Sada da bismo saznali površinu unije do sada pređenih pravougaonika sweep linijom, moramo odrediti ukupnu dužinu svih intervala čiji je brojač pozitivan.

Za svaki čvor tj. interval to čuvamo u atributu „ukupno“

Na kraju iteracije stavimo trenutni_y na y.

```
    for (auto dogadjaj : dogadjaji) {
        double y = dogadjaj[0], tip = dogadjaj[1], x1 = dogadjaj[2], x2 = dogadjaj[3];
        povrsina += trenutna_x_suma * (y - trenutni_y);
310     trenutna_x_suma = aktivni->azuriraj((Xi.find(x1))->second, (Xi.find(x2))->second,
                                            tip);
312     trenutni_y = y;
    }

    ShowMessage((int)(povrsina));
```

Dakle u svakoj iteraciji površina se povečava za vrijednost izraza

Trenutna_x_suma * (y-trenutni_y), gdje je y-trenutni_y razlika između prethodne posjećene tačke i trenutne tačke na kojoj je sweep linija, a trenutna_x_suma je dužina po x-u pravougaonika koje presjeca pomenuti horizontalni line sweep.

Pri čemu trnenutni_y ažuriramo na y.

Na kraju računa prikažemo Poruku na ekranu koja pokazuje izračunatu površinu u cijelom broju. (Double nije htio kako treba zbog nekih problema sa specijalnim vrstama stringova)