SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Seminar za treću godinu
Kolegij: Programiranje 2
Tema:

Generator koda za oblikovanje prozora u Windows okruženju

Mentor: Prof.dr.sc.Slavko Vidović
Asistent: dr.sc.Danijel Radošević

Autor: Davor Sauer
33898/03-R

U Varaždinu, 01.09.2006.

Sadržaj:

1. API Funkcije

1.1. Uvod u API

API je skraćenica od riječi Application programming interface. Prevedeno na hrvatski, sučelje za aplikativno programiranje. To je skup funkcija koje nam olakšavaju rad pri pisanju programa.

Windows sustav nam pruža mnoštvo API funkcija koje nam pripomažu u razvijanju novih aplikacija, pomoću upravljanja sustavom, izgrađivanja bolje vizualnijih aplikacija i što je najvažnije aplikacije postaju funkcionalnije i lakše upotrebljive. Svaka aplikacija, ne samo u Windows okruženju, mora u barem malom dijelu upravljati sustavom kako bi si npr. „rezervirala“ prostor u memoriji ili upotrijebila neki sistemski resurs za normalno izvršavanje. Sve funkcije imaju svoji smisao i bitno olakšavaju rad programeru pri pisanju programa. Za mnoge funkcije ni ne znamo da postoje, ali one su tamo da ih upotrijebimo i olakšamo si pisanje iste takve funkcije. Njih može koristiti bilo koja Windows aplikacija u Windows operativnom sistemu. Dakle, stoje na raspolaganju svima. To uključuje i programske jezike, naravno. Bilo koji programski jezik ima opciju da iskoristi funkciju pri pisanju programa u njemu. Jedina razlika između jezika je sintaksa, ali izlaz je uvijek isti.

Aplikativne funkcije za programiranje su dio svakog operacijskog sustava koje pripomažu pri pisanju programa i služe u različite svrhe kao što su upravljanje sistemom, upravljanje mrežama, vizualizacijom, multimediji i dr.

API je jednostavno skup funkcija na raspolaganju programerima. Funkcije DOS sustava tehnički možemo smatrati DOS-ovim API sučeljem. Ako pišete programe za baze podataka u dBase, dBase funkcije koje koristite također mogu biti smatrane za dBase API.

Ovaj termin najčešće opisuje skup funkcija koje su dio jedne aplikacije, ali im pristupa neka druga. Dakle, Windows API označava skup funkcija koje su dio Windowsa, a može ih koristiti bilo koja Windows aplikacija.

API funkcije su sastavni dio Windows okruženja i sadržane su u posebnim dll-ovima u sistemskom direktoriju od Windows operacijskog sustava. Funkcije se spremaju u posebne Windows izvršne programe pod ekstenzijom dll. Na taj način je omogućeno njihovo lakše pozivanje i olakšano je pisanje funkcija na koje bi smo trošili naše vrijeme i otežavali pisanje programa.

Pozivanje API funkcija je zapravo isto kao i pozivanje bilo koje funkcije u programskom jeziku C. Dakle upiše se ime funkcije i proslijede se argumenti, npr. ShowWindow(argument1, argumet2);

1.2. DOS na Windows

Windows okruženje je bitno zamršenije od DOS okruženja – jednostavan postupak iscrtavanja linije na ekranu uključuje upotrebu objekata poznatih kao Windows identifikator, sadržaj uređaja i pero. Proces povezivanja vanjskih funkcija je različit – Windows aplikacije najčešće koriste tehniku poznatu kao dinamičko povezivanje, za razliku od statičkog koje je uobičajeno u DOS-u.

Niti jedan programski jezik ne sadrži svaku naredbu i funkciju koja bi mogla biti potrebna bilo kojem programeru. U DOS okruženju taj je problem riješen kreiranjem i upotrebom biblioteka funkcija koje proširuju mogućnosti programskog jezika i koje prema potrebi mogu biti povezane u program.

1.3. Razlika između statičkog i dinamičkog povezivanja u datoteku

Pojam povezivanje zapravo znači uključivanje vanjskih funkcija u program. Većina jezika nudi mogućnost pristupa ugrađenim funkcijama operativnog sustava i većina jezika omogućuje kreiranje biblioteka funkcija koje mogu biti povezane u program. Te funkcije programer kasnije doživljava kao da su ugrađene u sam programski jezik.

Programski moduli, u kojima su funkcije sadržane, kompilirani su u objektne datoteke. Takve datoteke su često grupirane u bibliotečne datoteke korištenjem odgovarajućih programa.

1.3.1. Statičko povezivanje

Kada je vrijeme za stvaranje konačne izvršne verzije programa, program za povezivanje pretražuje objektne datoteke projekta nalazeći pozive funkcija koje nisu definirane. Takve funkcije program za povezivanje zatim pronalazi u pridruženim bibliotekama te ih izdvaja i prepisuje u novu izvršnu datoteku, povezujući ih s programom. Ovakav proces se naziva statičko povezivanje. Problem ovog povezivanja je u bespotrebnom trošenju resursa računala. Zamislimo da se funkcija ShowMessage() pozove 5 puta u programu. Tada bi se ta ista funkcija zapisala na 5 mjesta i bespotrebno utrošila 5 puta više resursa nego što je potrebno.

1.3.2. Dinamičko povezivanje

Radi ovih nedostataka u Windows okruženju su takvi moduli, odnosno funkcije, povezani u posebnom obliku Windows izvršne datoteke poznate kao dinamički poveziva datoteka (DLL). Pri izvođenju programa, svaki puta kada se ukaže potreba za funkcijom koja nije sadržana u njegovoj izvršnoj datoteci, Windowsi unose u memoriju dinamički povezivu biblioteku tako da se sve funkcije koje sadrži postaju dostupne vašoj aplikaciji. Pri tome se adresa svake od funkcija dinamički povezuje s vašom aplikacijom.

1.4. O API funkcijama

API funkcije se pozivaju različito u svakom programskom jeziku. Zavisi o pozivanju funkcija u jeziku u kojem pišemo. Jedina ista značajka je da se naziv funkcije ne mijenja, niti se mijenjaju argumenti koji se prosljeđuju.

Windows 32-bitno sučelje za aplikativno programiranje daje široki set funkcija struktura za programski jezik C s kojima radimo Windows aplikacije. API pozivi često izvršavaju različite usluge za Windows aplikacije kao što je davanje običnog prozora za otvaranje i spašavanje datoteka ili za slanje datoteka na ispis, ali njihova moć se očituje u funkcijama za upravljanje sistemom.

Win 32 API – funkcijska područja	
UPRAVLJANJE PROZORIMA	KONTROLE I DIALOZI PROZORA
UPRAVLJANJE SISTEMOM	SHELL OSOBINE
GRAFIČKO SUČELJE	INTERNACIONALNE OSOBINE
MULTIMEDIJA	MREŽNE USLUGE

Postoji preko 10000 API funkcija koje su dostupne u sadašnjim verzijama Windowsa, a sadržane su unutar nekoliko desetaka dll-ova. Te dll datoteke se mogu naći na WINOWS / SYSTEM direktoriju. Neke od njih su:

kernel32.dll - sadrži kernel biblioteku
User32.dll - biblioteka za korisnika
Gdi32.dll - biblioteka za grafičko sučelje
Mmsystem.dll - biblioteka za multimediju
Shell32.dll - shell biblioteka
Wininet.dll - biblioteka za Internet
Winsock.dll i wsock32.dll - Windows sockets

Mnoštvo API funkcija su standardizirane u alatima za programiranje koji su objektno orijentirani, kao što su „Visual Basic“ (gledamo na programski alat) ili Borland C++. Ti alati nam daju gotove funkcije koje samo trebamo implementirati u program preko VCL-a (vizualnih biblioteka).

1.5. Upravljanje prozorima

Windows aplikacije su kreirane i održavane pomoću većine funkcija koje su sadržane u biblioteci za korisnike (user32.dll). Sučelje te biblioteke sadrži mnoge funkcije i rutine za prozore i održavanje menija, dijaloga, različitih poruka, pristupa mišu i tipkovnici i drugih različitih kontrola.

Pošto se ovo područje bavi održavanjem prozora, moramo shvatiti što je

zapravo prozor. Prozor je sučelje između korisnika i aplikacije. Kada se stvara program, prvo se stvara glavni prozor koji prima naredbe i daje informacije korisniku. Aplikacija može i sama stvarati prozore s kojima upravlja preko glavnog prozora.

Najčešće funkcije za održavanje prozora	**Opis funkcije**
CascadeWindows()	Cascade određeni prozor ili „dijete prozor“ od određenog „roditelja prozora“
CloseWindow()	Minimizira, ali ne gasi određeni prozor
CreateWindow()	Kreira prozor
DestroyWindow()	Gasi određeni prozor, šalje WM_DESTROY poruku određenom prozoru
EnableWindow()	Dozvoljava mišu ili tipkovnici pristup prozoru
EnumWindows()	Obilježava prozore sa brojevima
FindWindow()	Stavlja prozor kojeg tražimo u prvi plan
GetWindowRect()	Vraća koordinate određenog prozora
GetWindowText()	Vraća naziv prozora
MessageBeep()	Stvara predefinirani zvuk
MessageBox()	Stvara mali prozor sa porukom i nazivom određenim u aplikaciji
MoveWindow()	Miče lokaciju i veličinu prozora
PostMessage()	Upućuje poruku određenom prozoru
SetWindowText()	Mijenja naziv određenog prozora
SendMessage()	Šalje poruku drugom prozoru i čeka dok se poruka obradi
ShowWindow()	Stavlja status POKAŽI određenom prozoru (prozor može biti minimiziran, maksimiziran, sakriven...)

1.6. Grafičko sučelje

Ono nam omogućava da prozor ima mogućnost crtanja ili ispisa. To uključuje crtanje linija, teksta, mijenjanja fontova i promjena boja.

Najvažniji element grafičkog sučelja je device context (DC). DC predstavlja strukturu podataka koja definira grafičke objekte, atribute i izlazne stanja. DC je kreiran sa funkcijama CreateDC() i GetDC(). Imamo 7 tipova objekata grafičkog sučelja koje mogu biti selektirane u DC.

7 tipova objekata	Opis objekata
Bitmap	Kopiranje i micanje dijelova ekrana
Brush	Bojanje i ispunjavanje interijera poligona, elipsa i dr.
Font	Identificira tip, veličinu i stil fonta
Palette	Definira dostupne boje
Path	Bojanje i crtanje operacija
Pen	Crtanje linija
Region	Titranje i druge operacije

Grafičko sučelje je jako korisno kod 2D aplikacija kao što renderiranje i vizualizacija za poslovne aplikacije. Snaga i brzina tog iscrtavanja i vizualizacije je zavisna od računala do računala. Dakle koliko je ono snažno. Za neke veće pothvate, odnosno pravljenje 3D aplikacije, potrebna je podrška OpenGL-a i Direct3D naredbi. No, ovdje je to grafičko sučelje orijentirano na 2D aplikacije kojima je dovoljna podrška DirectDraw-a.

2. Generiranje koda

2.1. Uvod u generiranje koda

Generatori koda se dijele u dvije kategorije: aktivnu i pasivnu. U pasivnoj kategoriji, generator koda generira (izgrađuje) zadani kod, koji se kasnije može slobodno promijeniti.

Pasivni generator ne zadržavaju vjerodostojnost, bez obzira na dužinu koda, jer se on može naknadno mijenjati. 'Wizardi' integrirani u razvojnim sučeljima (IDE) su najčešće pasivni generatori koda.

Aktivni generatori zadržavaju vjerodostojnost, te omogućuju više generiranja koda za jednak izlaz. Na primjer mogu se promijeniti ulazni parametri, te ponovo pokrenuti generator kako bi se nadogradio dio koda koji ovisi o promijenjenim ulaznim parametrima.

2.2. Različite forme aktivnog generiranja koda

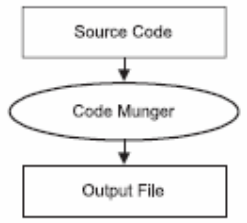
Unutar metode aktivnog generiranja koda postoje vrlo jednostavna i mala rješenja, ali isto tako i ona velika i vrlo složena rješenja. Te postoje mnogi načini raspodjele generatora koda. Osnova toj raspodjeli može biti njihova složenost, količina korištenja ili prema njihovom izlazu. Jadan od načina da ih razlikujemo je i prema njihovim ulazima i izlazim, gdje se oni dijele na šest tipova koji su opisani u daljnjem tekstu.

2.2.1. Mijenjanje koda

Mijenjanje koda ('Munging') je zapravo naziv za prepravljanje i preoblikovanje nekog koda iz jednog oblika u drugi.

Recimo da imamo neki postojeći kod, te pomoću ovog modela uzimamo samo važne dijelove tog koda koje ubacujemo u neku novu datoteku ili u neki kod da bi smo dobili novi izlazni kod korišten u neku drugu svrhu. Proces je vrlo jednostavan, što pokazuje slika 2.1.

Ova tehnika ima mnoge mogućnosti, na primjer kreiranje dokumentacije, ili čitanje konstanti ili funkcija iz datoteka.

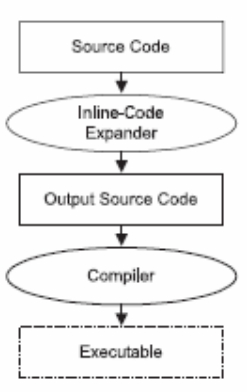


Slika: 2.1.

2.2.2. Linijsko proširivanje koda

Ova metoda uzima izvorni kod kao ulaz, te kreira izlazni kod. Ulazni kod sadrži specijalne oznake koje se proširivanjem zamjenjuju određenim kodom, koji na kraju postaje izlazni kod (slika 2.2.).

Ova vrsta generatora, pomoću proširivanja se često koristi kako bi se nadopunio SQL kod. Radi na principu da proširivač ide redom po kodu, te kad dođe do specijalne oznake, na tom mjestu implementira predložak ili naredbu za neki od jezika. Ideja je da se zadrži radni kod netaknutim, i da struktura ostaje postojana, te da se na kraju radi proširivanje.

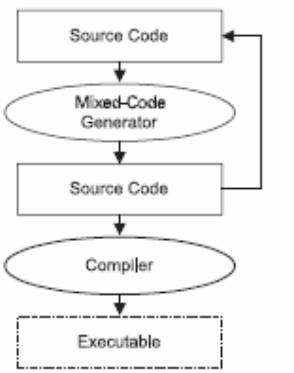


Slika: 2.2.

2.2.3. Mješoviti generator koda

Mješoviti generator koda čita izvorni kod, te ga mijenja i sprema u istu datoteku. To je osnovna razlika od linijskog generatora koda proširivanjem, što se izlazni kod sprema u ulaznu datoteku, tj. na mjestu ulaznog koda. Ova vrsta generatora radi tako da ulazni kod pretražuje za specijalnim oznakama, te ispunjuje to područje s nekim novim potrebnim kodom (slika 2.3.).

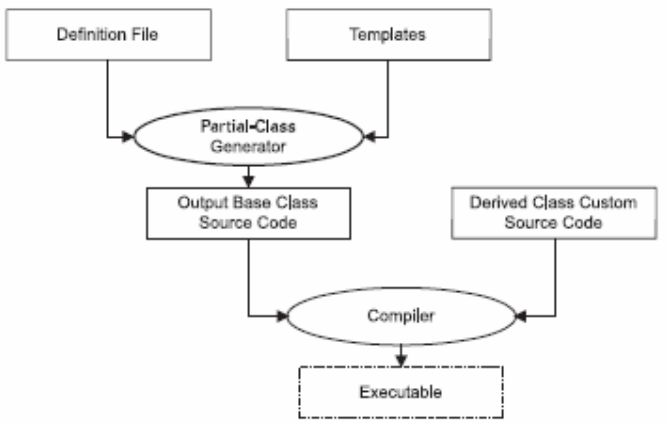
Mješoviti generatori koda imaju široku primjenu. Jedna od primjena je kretanje informacija između kontrolnih dijaloga i reprezentativnih varijabli u podatkovnoj strukturi. Komentari u kodu definiraju način mapiranja između podatkovnih elemenata i kontrola, te generator dodaje implementaciju koda će udovoljavati zadanom komentaru.



Slika: 2.3.

2.2.4. Djelomičan generator koda

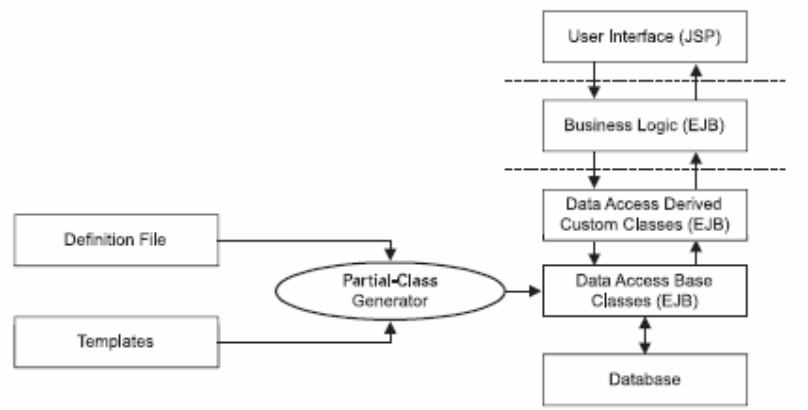
Djelomičan generator koda čita apstraktno (sažeto) definiranu datoteku, koja sadrži dovoljno informacija za izgradnju seta klasa. U sljedećem koraku se pomoću predložaka kreira izlazna bazna klasa. Te tu izlaznu baznu klasu dovršava inženjer, te nakon toga nastaje konačan set klasa.



Slika 2.4.

Slika 2.5 ilustrira kako se izlazni kod djelomičnog generatora koda uklapa u web arhitekturu. Podatkovni sloj je sastavljen od dvije klase. Prva klasa je bazična klasa koju generira djelomični generator koda, a druga je dorađena klasa od strane inženjera.

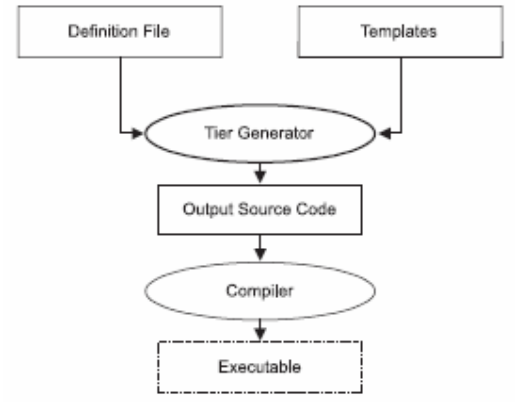
Djelomični generator koda je dobar početak za izgradnju generatora koji će biti u stanju generirati potpune i funkcionalne linije koda.



Slika 2.5.

2.2.5. Redni ili slojevni generator koda

U ovom modelu generator generira cijelu liniju koda, od N linija. Ulaz i izlaz generatora jednake je djelomičnom generatoru koda (slika 2.6.). Ovaj generator čita opisnu (apstraktnu) datoteku, te uz uporabu predložaka izrađuje izlaznu klasu na osnovu zadanih opisa u opisnoj datoteci.



Slika 2.6.

Velika razlika između rednog generatora koda i djelomičnog je ta što redni generator generira sav kod za red, a djelomični generator generira baznu klasu, koju je potrebno dovršiti, što radi inženjer.

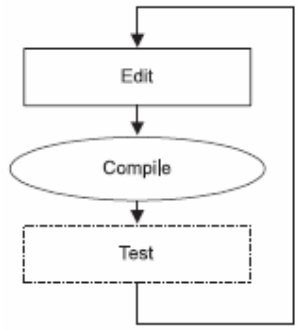
Osnovna prednost djelomičnog generatora je brzina implementiranja. Najveći problem u izgradnji rednog (slojevitog) generatora je definirati ga za specifičan problem, dok kod djelomičnog generatora možemo napraviti jednostavan generator, te djelomično generiran kod prilagoditi i definirati za specifičan problem.

2.2.6. Jezik široke primjene

Jezik široke primjene je Turingov jezik dorađen tako da omogući inženjerima reprezentaciju koncepta u nekoj domeni što lakše. Turingov jezik je široko primjenjiv računalni jezik koji podržava manipulaciju varijablama, logičke funkcije, grananje i druge mogućnosti uključene u današnje programske jezike. Prednost je što je moguće vrlo komplicirane funkcije opisa, tj. definiranja rješenja nekog problema kompajlirati u gotovo svaki viši programski jezik. Nedostatak je jedino u uvođenju novog jezika, te ga je potrebno podržati i dokumentirati.

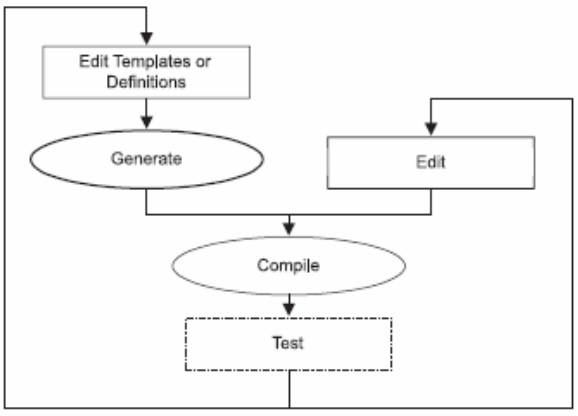
2.3. Tijek generiranja koda

Standardan tijek izrade i ispravljanja koda je editiranje, kompajliranje i testiranje (slika 2.7.), koja pokazuje ta tri stanja.



Slika 2.7.

U generiranju koda se dodaje i još nekoliko elemenata (slika 2.8.). Prvo se editira predložak i datoteka s definicijama (ili sami generator), te se tada pokreće generator koji generira izlazni kod. Izlazna datoteka je kompajlirana zajedno s promjenama, te se aplikacija testira.



Slika 2.8.

2.3.1. Editiranje generiranog koda

Generirani kod se ne bi trebao nikad direktno editirati, osim u slučaju ako se koristi generator koji linijski proširuje kod (poglavlje: 2.2.2. Linijsko proširivanje koda), u kojem se izlazna datoteka dalje koristi u ciklusu generiranja. Zato što generator koda u potpunosti izmjenjuje izlaznu datoteku, te bi se bilo kakvim editiranjem i izmjenama izgubila prethodno originalno generirana datoteka. Zato se preporuča da se na početku svake datoteke stavi komentar, koji upozorava korisnika da ne mijenja sadržaj dotične datoteke.

Postoji samo jedna iznimka, a to je u slučaju ispravljanja pogrešaka, jer je lakše utvrditi problem u izlaznoj datoteci, te rješenje tog problema ili greške promijeniti u predlošku prema kojem se generirala izlazna datoteka, koju u osnovi nismo promijenili, nego je samo poslužila za utvrđivanje pogreške ili dorade.

2.4. Vještine generiranja koda

2.4.1. Korištenje tekstualnih predložaka

Generiranje koda programski znači izgradnja datoteke sa složenom strukturom. Da bi proces bio što jednostavniji uvodi se korištenje tekstualnih predložaka u HTML-u, JSP-u, ASP-u i drugim jezicima. Korištenje tekstualnih predložaka znači da možemo odvojiti strukturu i način oblikovanja koda od problematike i rješenje problema. Na taj način je apstraktno odvojena logička razina definiranja koda i logička razina formatiranja koda.

2.4.2. Pisanje standardnih izraza

Pisanje standardnih izraza se radi pomoću najjednostavnijih alata zamjene, tako da se traži neki kontekst te se on zamjenjuje s nekim određenim kodom ili se traži informacija koju sadrži pronađeni izraz. Čitanje konfiguracijskih datoteka ili skeniranje izvornog koda, znatno je olakšano pomoću standardnih izraza. Standardni izrazi omogućuju da definiramo neki oblik teksta te uz pomoć njega tražimo tekst koji se poklapa s njima, tj. pretražujemo neku datoteku za takvim tekstom, te na taj način dolazimo do informacije u tom tekstu. Tada s tim informacijama možemo manipulirati na bilo koji način.

2.5. Izbor jezika za generiranje koda

Jedna od prednosti razdvajanja samog pisanja koda od samog konačnog koda, je ta što se ta dva koda ne moraju pisati istim jezikom. Zapravo je i dobra ideja koristiti dva različita jezika.

Generatori koda čitaju mnoge datoteke (konfiguracijske, datoteke s predlošcima) koje koriste za stvaranje konačnog izlaznog generiranog koda, koji je rješenje problema.

Karakteristike jezika za stvaranje generatora koda:

- Laka čitljivost,
- Tekstualni predlošci su bitni, te moraju biti jednostavni za uporabu, te mora postojat neki bolji alat za rad s predlošcima
- Jednostavno održavanje i pretraživanje datoteka, što je zapravo posao generatora koda, da traži i čita ulaznu datoteku i manipulira izlaznom datotekom sa izlaznim generiranim kodom.

U osnovi se traži jezik koji je djelotvoran za razvoj, jednostavan za održavanje, te podržava programske modele od jednostavnih skripti do objektno orijentiranih rješenja.

2.5.1. Usporedba jezika za generiranje koda

Programski jezik	Prednosti	Nedostatci
C, C++	- Ako je generator poslan korisniku, tada korisnik neće biti u mogućnosti čitati ili mijenjati izvorni kod.	- Jezik je pogodan za tekstualnu analizu zadatka - Stroga klasifikacija nije idealna za tekstualno obrađene aplikacije - Brzina izvođenja programa pisanih u C-u nije toliko bitna - Ulazno izlazni direktorij nije pokretan - XML i regularni alati sa izrazima su teški za uporabu
Java	- Ako je generator poslan korisniku, tada korisnik neće biti u mogućnosti čitati ili mijenjati izvorni kod. - Kod je prenosiv na razne operativne sisteme - postoji alat za rad s XML-om - postoji alat za rad s tekstualnim predlošcima	- Java nije idealna za tekstualnu analizu - Stroga klasifikacija nije idealna za tekstualno obrađene aplikacije - Implementacija je veća za manje generatore
Perl, Ruby, Python	- Kod je prenosiv na razne operativne sustave	- Ostali inženjeri moraju naučiti kako kreirati i mijenjati kod u tim

	- jezici su namijenjeni za mele i velike generatore - analiza je jednostavna - moguće je koristiti unaprijed ugrađene standardne izraze - jednostavna je uporaba XML API-ja - postoje alati za rad s tekstualnim predlošcima	jezicima

3. O programu

Program je namijenjen ponajprije programerima ili dizajnerima sučelja koji žele oblikovati programsku formu (izgled prozora) prema vlastitoj viziji. Program služi samo za osnovno oblikovanje prozora, znači vanjski oblik prozora, bilo jednog ili više koji će se koristiti u projektu na kojem neki tim radi. Način rada s programom, opisan je u poglavlju "Primjer korištenja". U osnovi program radit tako da mu zadate neku željenu sliku prozora, osnovni oblik, tj. oblik prozora kakav se želi dobiti, te uz pomoću programa izrežete željeni prozor, tj. uklonite rubove koji vam smetaju. Te nakon toga program generira programski kod s API funkcijama i potrebnim koordinatama, koji programer stavlja u program, te uz pomoć tog koda prikazani prozor će biti oblikovan na željeni način.

Način korištenja programskog koda ovisi o programeru, tako da forma može imati željeni oblik prilikom svakog pokretanja forme ili nakon pritiska neke tipke, pokretanja funkcije i slično.

Generirani programski kod namijenjen je za rad u Visual Basicu okruženju.

3.1. Korišteni alati i realizacija

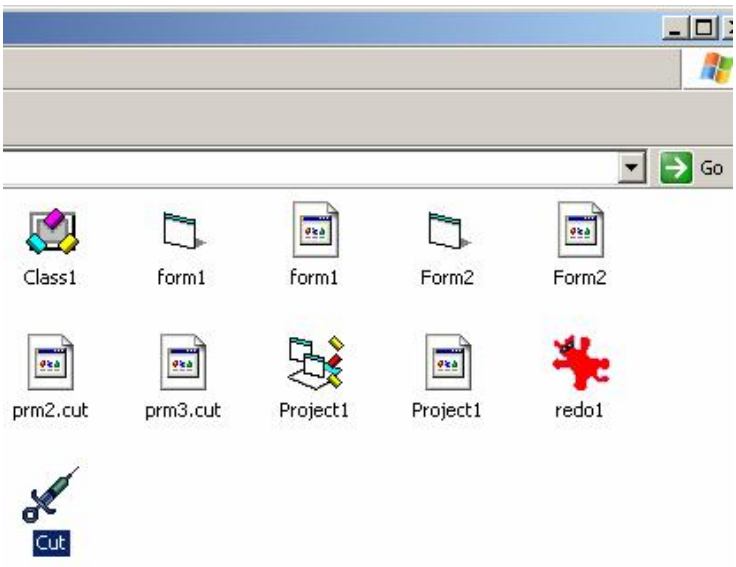
Program je napisan u Visual Basic-u (verzija 6.0), te je sve potrebno napravljeno u njegovom okruženju. Za rad s API funkcijama sam koristio MSDN *library* na Internetu. Za korištenje dijaloških prozora 'save' i 'open' koji su standardni dio Windowsa, koristio sam implementirani modul s potrebnim API funkcijama za pozivanje dijaloga, iako je taj dijalog moguće pozvati preko komponenata za projekt (Project->Components ; Microsoft Common Dialog Control), ali ja sam koristio ovaj način zato da program bude ovisan samo standardnim run-time komponentama.

Na taj način je izbjegnuta komplicirana instalacija svih komponenti. Tako da za pokretanje programa na računalu treba imat samo instalirane run-time komponente za visual Basic 6.0 ili kasnije verzije i izvršnu verziju programa.

3.2. Primjer korištenja

Prije pokretanja programa, na računalu je potrebno imat instaliran Visual Basic Run-time Library, od verzije 6 pa na više.

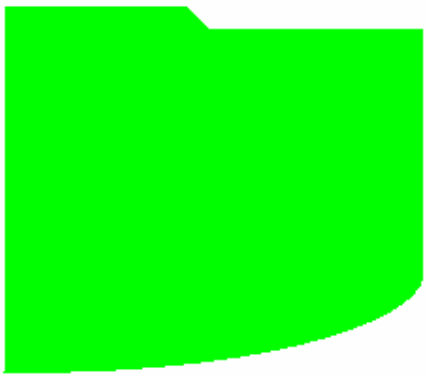
Program nije potrebno prethodno instalirati, pa je njegovo pokretanje svedeno na pokretanje izvršne datoteke (slika 3.1.).



Slika 3.1. Pokretanje

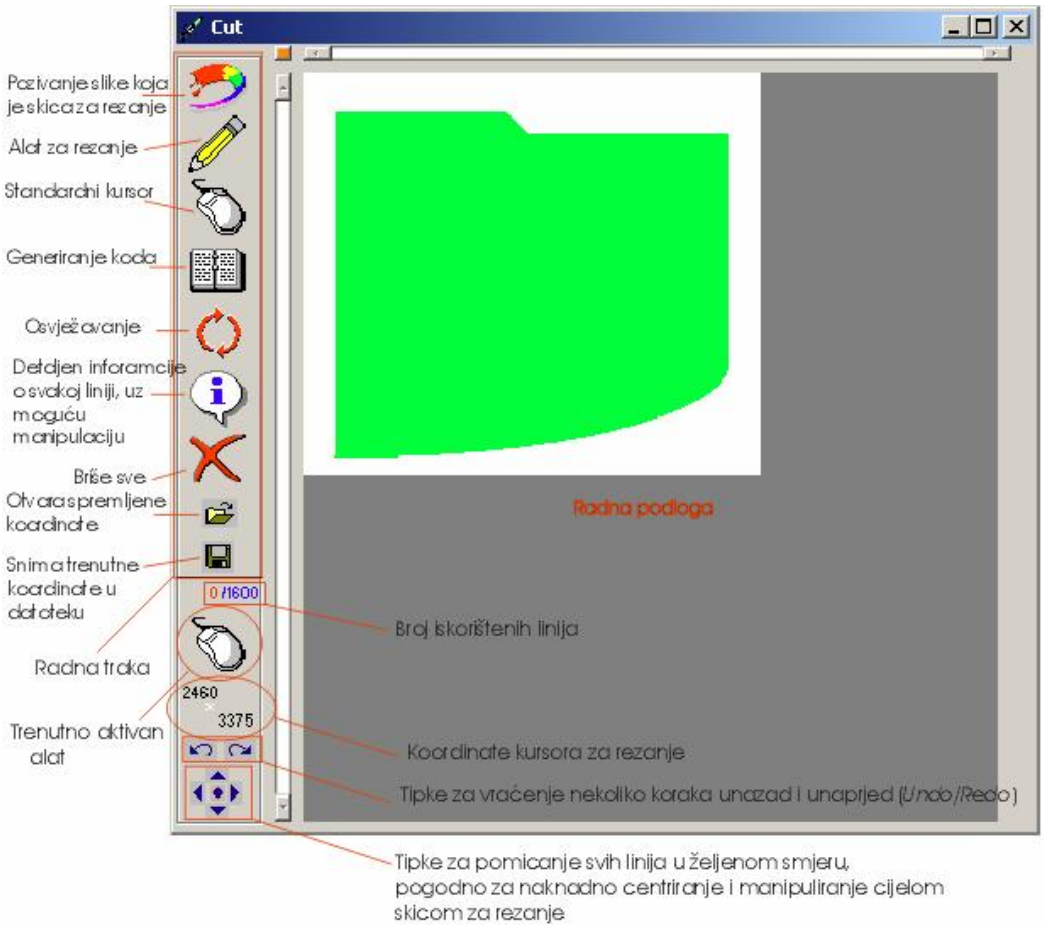
Nakon pokretanja programa, možemo imat definiran izgled prozora koji želimo oblikovati, ili možemo pozvati neki prozor i proizvoljno ga izrezati. U ovom primjeru sam proizvoljno napravio predložak kako bi konačno prozor trebao izgledati. Ta dio posla može se obaviti i u paintu.

Primjer skice prozora:



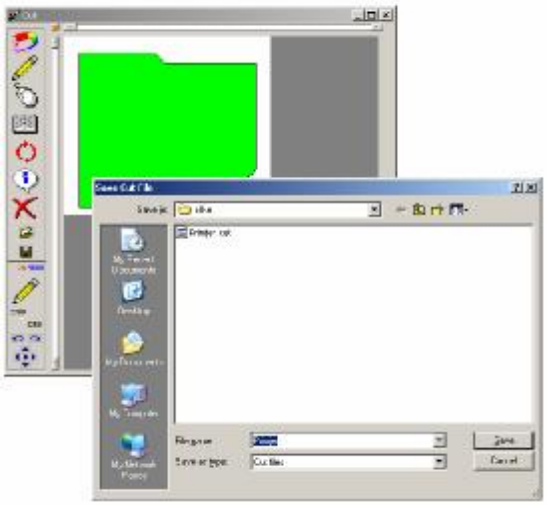
Slika 3.2. Oblik prozora koji želimo dobiti

Nakon toga u programu za rezanje forme (Cut.exe), pozivamo sliku željenog prozora.



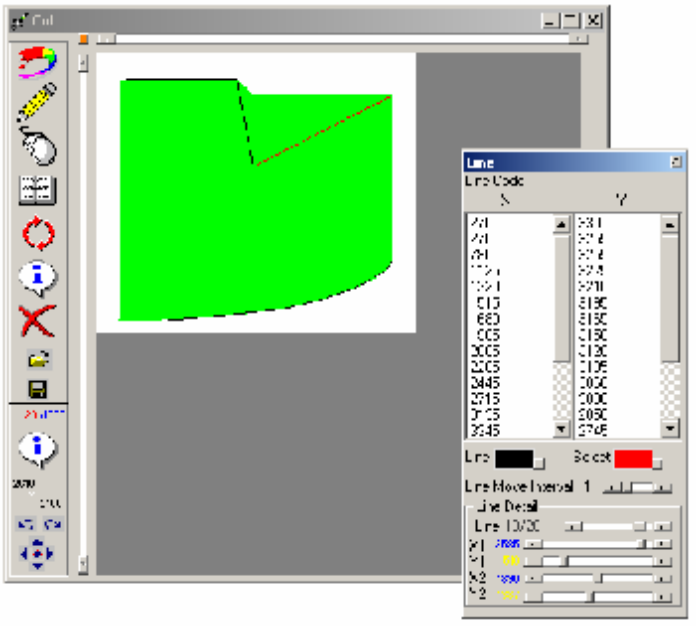
Slika 3.3. Glavno radno sučelje programa

Nakon toga oblikujemo željeni oblik prozora (Slika 3.4.). Vidljiva crna linija oko 6 Vidljiva crna linija oko zelenog objekta, te aktivni prozor, pomoću kojeg spremamo koordinate pojedinih linija u datoteku.



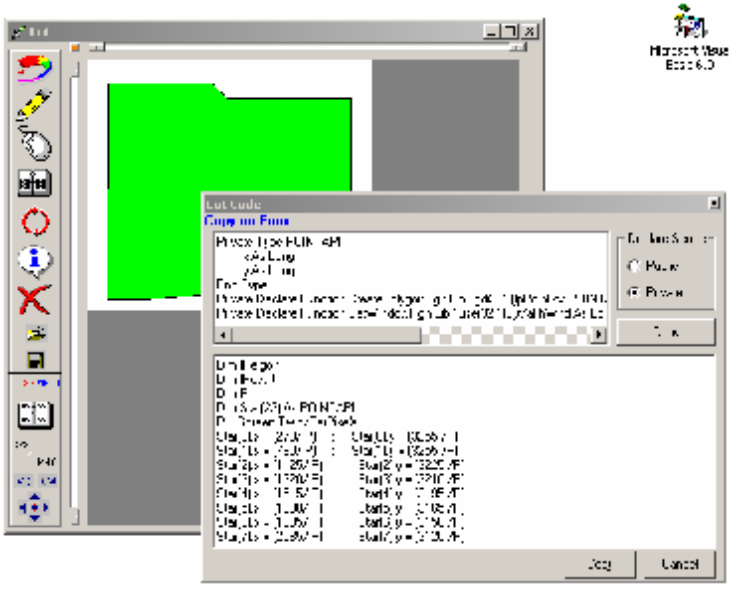
Slika 3.4.

Nakon toga, možemo naknadno korigirati pojedinu liniju. Označena linija je crvene boje, te smo u ovom slučaju njezinu lijevu stranu povukli prema dolje (Y2), s čijom se promjenom koordinate, mijenja i koordinata linije koja je nadovezana na tu točku. Boje linija, aktivnih i neaktivnih možemo naknadno mijenjati, zbog potreba raznobojnosti sheme prozora koju želimo izrezati. Jer postoji mogućnost da s crnom linijom nećemo vidjeti gdje vučemo željene linije.



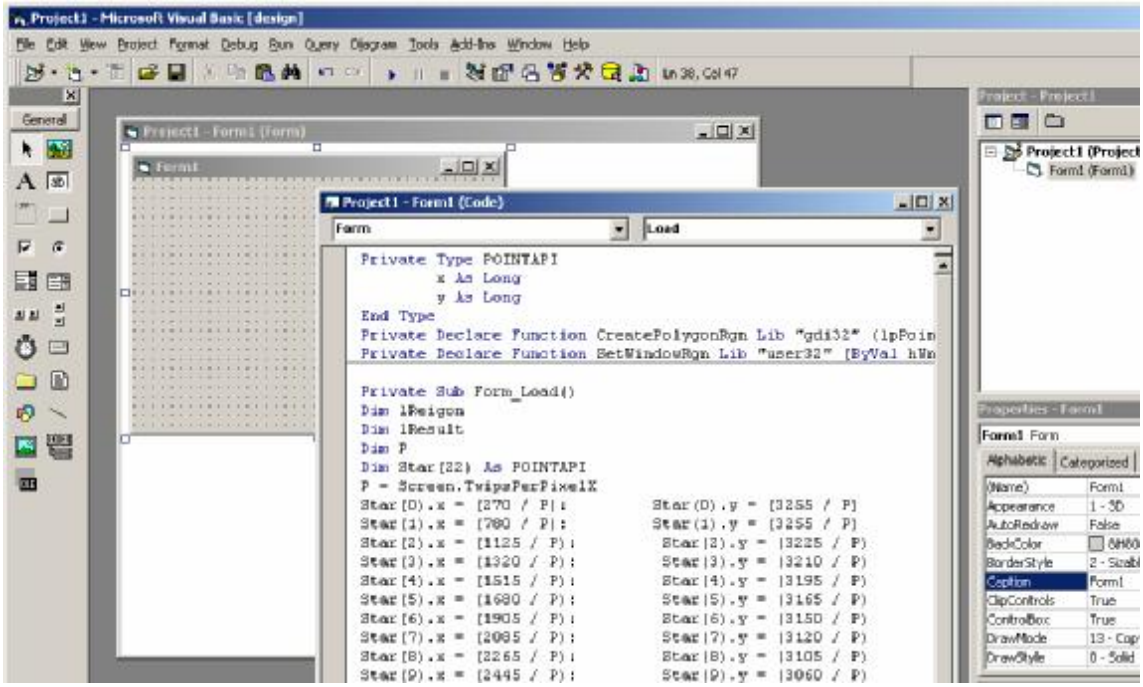
Slika 3.5. Detaljne informacije i opcije nad svakom linijom

Nakon svih korekcija i rezanja, slijedi generiranje koda, koje radi na principu da generator pokupi koordinate svih linija koje smo kreirali, te ih stavlja u predložak koji je fiksan, na početku i kraju, jedino su promjenjive vrijednosti varijabli koje su koordinate i količina koordinata, tj. broj linija koje okružuju željeni oblik prozora.



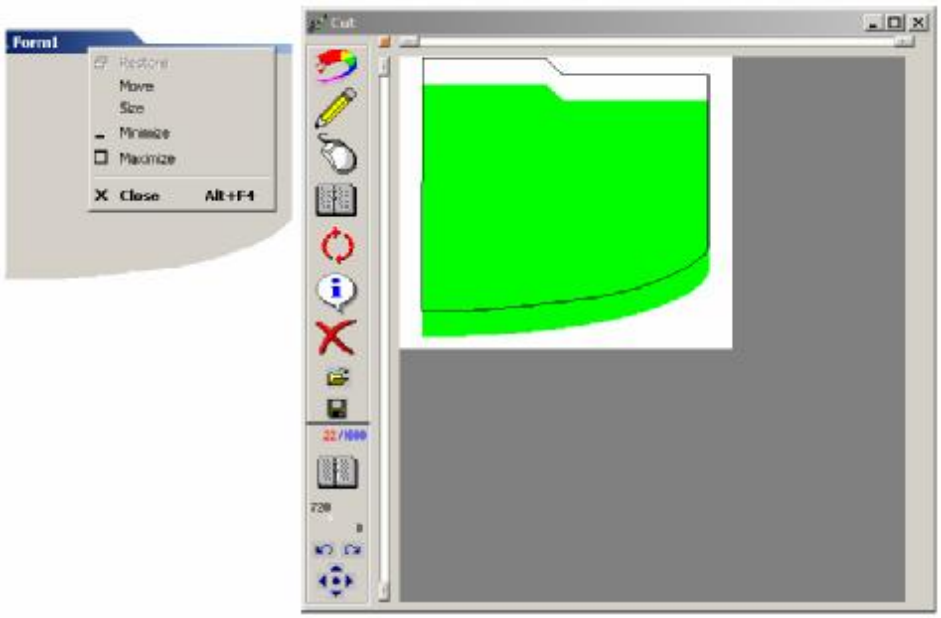
Slika 3.6. Prozor sa generiranim kodom

Generirani kod kopiramo u Visual Basic, kako je prikazano na slici.



Slika 3.7. Generirani kod u Visual Basicu

Te nakon toga slijedi pokretanje programa u Visual basicu, i kao što se vidi na slici, dobivamo željeni oblik prozora. Lijevo je dobiveni prozor u Visual Basicu, a desno je željeni izgled prozora u programu Cut, pomoću koje smo izrezali prozor i dodatno korigirali oblik, pomaknuvši ga prema gore, tako da bi prozor u Visual basicu imao i naslovnu traku. (Slika 3.8.)



Slika 3.8. Izrezani prozor (lijevo), skica prozora koji želimo dobiti (desno)

4. Zaključak

Generatori su pogodni za kreiranje neke druge forme na osnovu parametara koje zadajemo u nekom formularu, bazi podataka ili najobičnijoj datoteci, te generator koda na osnovu tih ulaznih informacija kreira zadanu formu. Zadana forma ovisi o vrsti primjene.

Generatori koda se često koriste na webu, tako da generiraju kod web stranice, te da korisnik ne mora stalno biti u doticaju s programiranjem weba, nego jednostavno dodaje informacije u bazu, a generator koda automatski generira kod za web sa novim informacijama, te ih stavlja na web stranicu. Na taj način je omogućena automatizacija i ažuriranje novosti i drugih informacija. To je samo jedan primjer korištenja generatora, drugi je recimo u ovom seminaru, gdje se generatorom služim da bi generirao kod koji izrezuje prozor u zadanom obliku.

Generatori koda dolaze u nekoliko standardnih formi, i potrebno je biti upoznat sa svakom od njih, da bi mogli radit s njima i da bi smo mogli točnije izabrati onu prikladniju formu za trenutan projekt ili problem, kao dio rješenja. Generatori nam uvelike olakšavaju posao u kojem ima dosta ponavljanja, te na taj način ubrzavaju proces izrade projekta, tj. realizacije rješenja.

5. Literatura

1. Jack Harrington, Code Generation in Action, Manning, 2003
2. MSDN Library