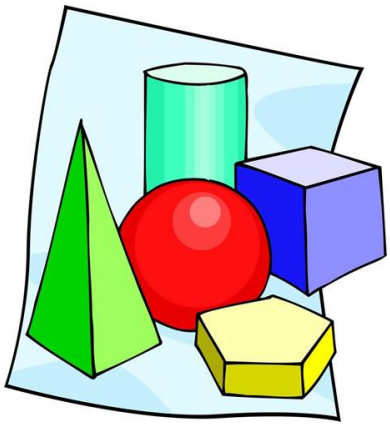
Mestna občina Celje

Komisija Mladi za Celje



Program za učenje geometrije

(RAZISKOVALNA NALOGA)

AVTOR

Urban Lavbič

MENTOR

mag. Boštjan Resinovič

Celje, Marec 2013

ŠOLSKI CENTER CELJE

Šola za kemijo, elektrotehniko in kemijo

Program za učenje geometrije

(Raziskovalna naloga)

Mentor:

Mag. Boštjan Resinovič

Avtor:

Urban Lavbič, E-4.e

Mestna Občina Celje, Mladi za Celje

Celje, 2012/2013

IZJAVA*

Mentor , mag. Boštjan Resinovič, v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi naslovom Program za učenje geometrije, katere avtor je Urban Lavbič :

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo (-ičino) dovoljenje in je hranjeno v šolskem arhivu;
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na spletnih portalih z navedbo, da je nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 18.3.2013 žig šole Šola

Podpis mentorja(-ice)

Podpis odgovorne osebe

* Pojasnilo

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja(-ice) in odgovorne osebe šole uvezati v izvod za knjižnico, dovoljenje za objavo avtorja(-ice) fotografskega gradiva, katerega ni avtor(-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

1. KAZALO

1.1 KAZALO SLIK

1.2 KAZALO TABEL

2. POVZETEK

V tej raziskovalni nalogi sem izdelal program za učenje geometrije. Osredotočil sem se predvsem na like, ki smo se jih učili v šoli (paralelogrami, trikotniki, krogi). Program omogoča uporabniku, da v vsako polje za podatke v uporabniškem vmesniku vnese pravilen matematični izraz z osnovnimi aritmetičnimi operatorji. Pri vpisovanju podatkov usmerja uporabnika k vpisovanju podatkov, ki so še potrebni za izračun vsega, kar je mogoče za ta lik. Vsakič, ko uporabnik klikne katerega od gumbov za izračun, se mu izpiše celotna izpeljava od osnovne enačbe do končnega rezultata. S tem je izdelek pridobil izobraževalno vrednost, saj pokaže uporabniku, kako se pride do rešitve.

2.1 KLJUČNE BESEDE

- Uporabniški vmesnik (UV)
- Selektivni vnos
- Leksikalni razčlenjevalnik
- C#
- Geometrija

3. UVOD

Poznamo veliko aplikacij, ki nam omogočajo izris grafov, izračunavanje enačb s spremenljivkami, toda nimamo aplikacije, ki bi nas znala naučiti, katere enačbe lahko uporabljamo v likih, postopek računanja in nas usmerjala glede na to, kakšne podatke imamo za izračunavanje lika.

Če bi želel takšen projekt narediti sam, bi moral začeti že v 1. letniku mojega šolanja, ampak takrat nisem imel niti potrebnega znanja matematike, še manj pa potrebnega znanja programiranja. Zato se nisem osredotočil toliko na izdelovanje enačb ter likov, ampak na izdelavo osnove programa, ki ponuja vse kar bi tisti, ki bi nadaljevali, potrebovali za izdelavo likov ter kasneje trikotnikov, krogov in teles.

3.1 CILJI IN HIPOTEZE

Cilj je ustvariti program, ki bo pomagal uporabnikom pri učenju geometrije.

Omogočati mora :

- Vnos matematičnih izrazov
- Selektivni vnos, ki vodi uporabnika, da vpisuje pravilne podatke za izračun
- Primerjanje primerov likov, da lahko uporabnik shrani ter primerja like

Moja hipoteza je s pomočjo uporabniškega vmesnika, prilagojenega za določen namen, ter razčlenjevanjem uporabniškega vnosa napisati program, ki omogoča hitrejše ter lažje delo z liki, pri tem pa pokaže uporabniku, kako lahko preko enačb pride do enakega rezultata kakor program.

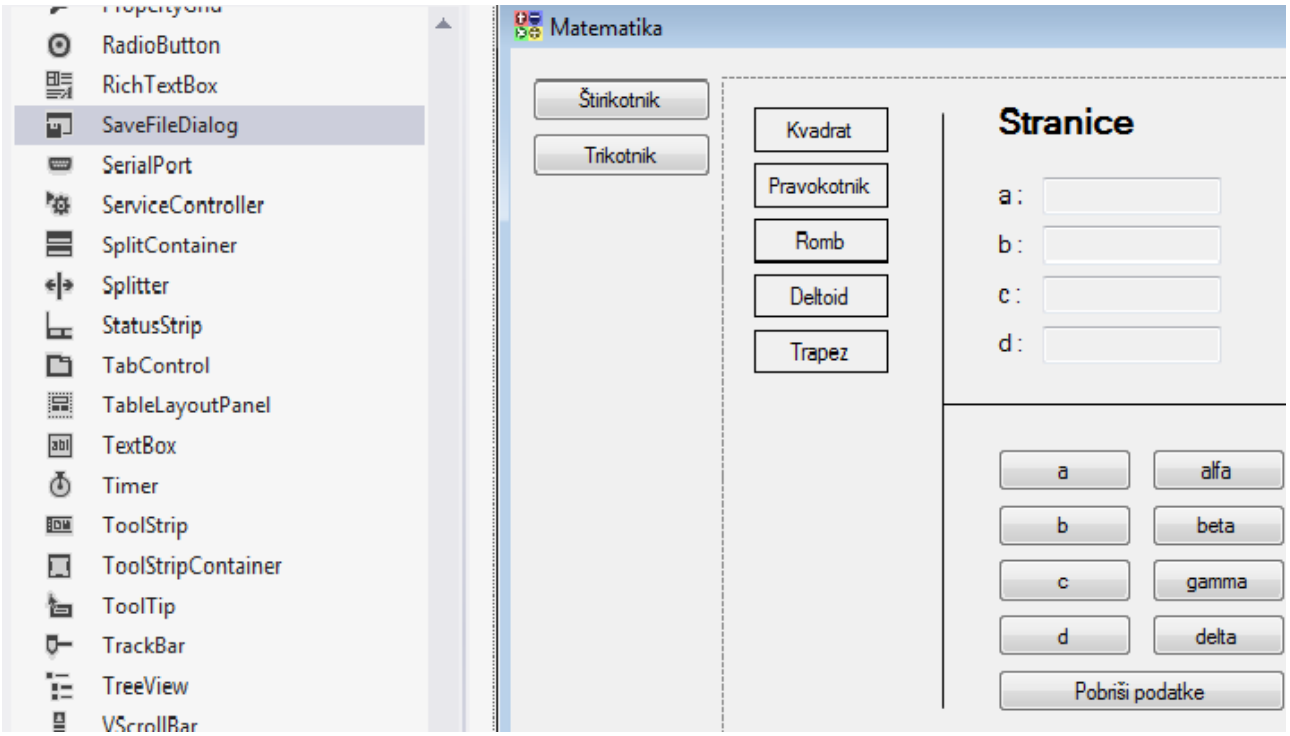
3.2 OPIS RAZISKOVANJA

Začel sem z raziskovanjem enačb, ki sem jih uporabil v programu. Moral sem poiskati vse možne uporabe in variante ter nato izločiti tiste, ki so nepotrebne. Pomagal sem si z zapiski iz matematičnega zvezka ter z učbenikom za matematiko, kjer sem dobil osnovne enačbe. Nato sem se lotil odkrivanja novih, ki mogoče obstajajo. Sledilo je ustvarjanje knjižnice za program, v kateri sem moral narediti drevesno strukturo, ki je najbolje zajela vse like in s tem hkrati zmanjšala število ponavljajoče se kode. Zaradi tega sem moral raziskati, kateri liki si med seboj delijo največ skupnih lastnosti in iz katerega bi lahko izpeljal drugega. Najtežji del je bil leksikalni razčlenjevalnik za matematične izraze. Na internetu sem našel zelo dober opis rešitve, s katero sem si pomagal napisati osnovo za lastni razčlenjevalnik. Moral sem narediti analizo sintakse ter ugotoviti, kako pravilno sortirati operatorje, da bodo na koncu, ko razčlenjevalnik prevede iz infiksne notacije v postfiksno notacijo, operandi ter operatorji v pravilnem vrstnem redu.

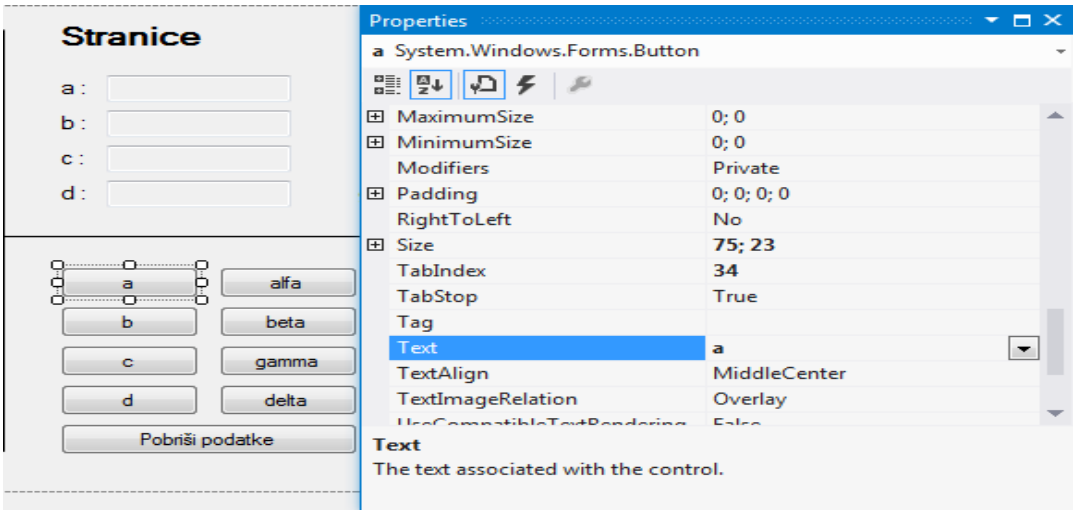
4. APLIKACIJA

4.1 ORODJA ZA IZDELAVO APLIKACIJE

Za izdelavo te aplikacije sem uporabljal Visual Studio 2012(VS). To je integrirano razvojno okolje s strani Microsoft-a, ki podpira več različnih jezikov. VS za razliko od drugih orodij ponuja veliko obširnejšo pomoč pri izdelavi, saj ima veliko več dodatnih možnosti, kar se še posebej pozna pri izdelavi uporabniškega vmesnika, kjer se uporablja sistem „drag&drop“ oz. primi ter spusti. To pomeni, da neki element, kot je recimo gumb ali vnosno polje, primemo z miško, ga povlečemo na delovno površino in ga preprosto spustimo nanjo.

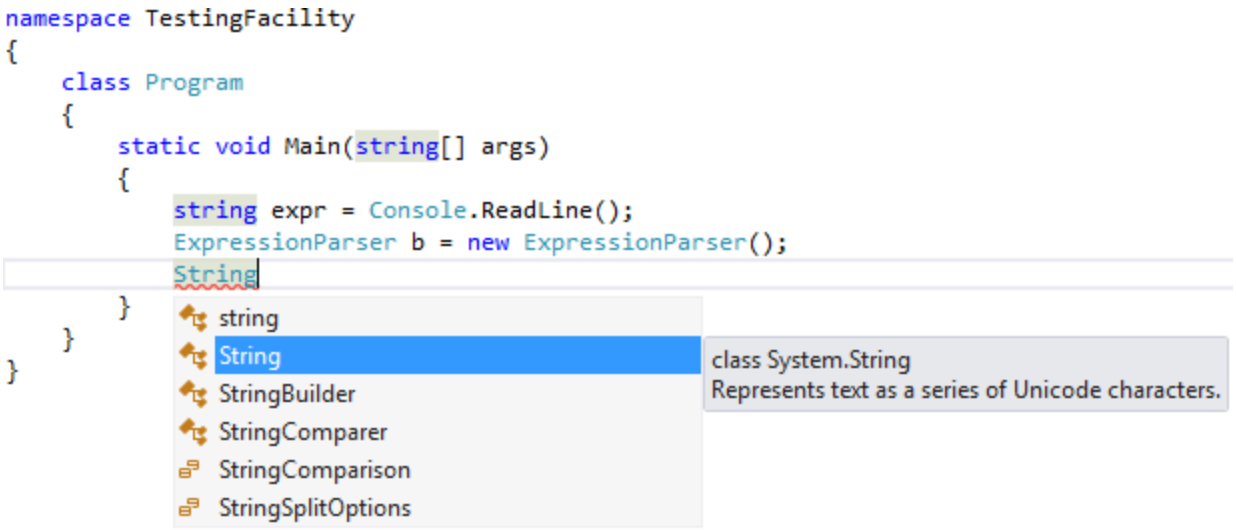


Slika 1 : Prikaz ToolBox-a in delovne površine



SLIKA 2 : PRIKAZ LASTNOSTI ELEMENTA

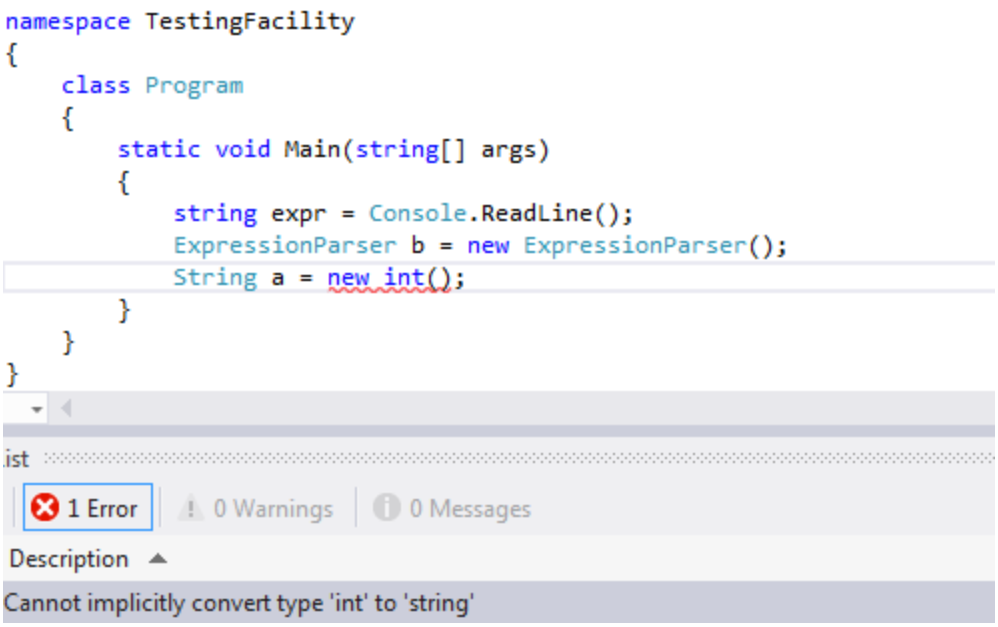
Zelo dobra funkcija VS je tudi IntelliSense, ki pomaga pri iskanju metod/funkcij pri programiranju in pomaga dokončevati dolge ukaze. Lahko prihrani veliko časa ter iskanja, zmeraj ponudi vse možnosti, kar jih je na voljo, mi pa se lahko sprehajamo gor in dol po seznamu in izberemo tisto, ki nam ustreza. Seveda pri vsaki napiše tudi veliko količino podatkov, kaj ta metoda/funkcija naredi in kaj vrne tako, da lahko lažje najdemo, kar iščemo.



SLIKA 3 : PRIKAZ KAKO INTELLISENSE PONUJA MOŽNOSTI

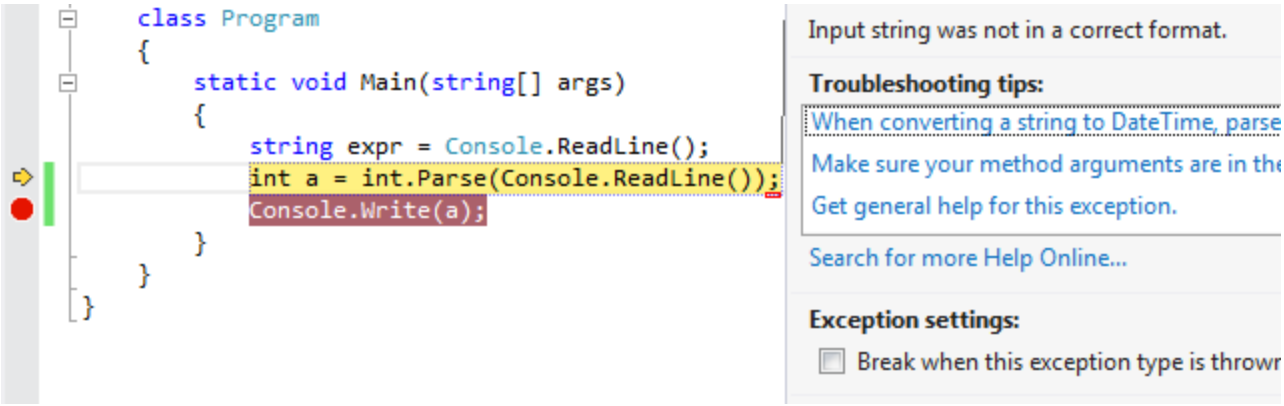
Najpomembnejša lastnost vseh razvojnih orodij je preverjanje sintakse. To pomeni, da razvojno okolje takoj, ko prenehamo pisati, preveri, ali smo pozabili zapreti kakšen oklepaj, narekovaj ali

morda pozabili podpičje na koncu stavka. Prav tako preveri, ali je koda nedostopna zaradi določenih ukazov in če smo poskusili brez pretvorbe določiti spremenljivki vrednost drugega tipa. Skratka opozori nas na vse napake, kar smo jih naredili pri pisanju. To so sintaktične napake. Na semantične napake oz. napake v „logiki“ aplikacije pa nas ne more opozoriti, saj računalniki ne razmišljajo tako kot mi.

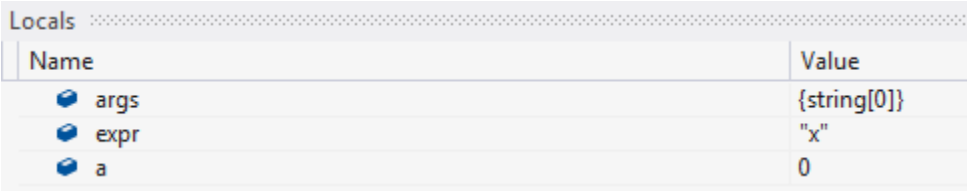


SLIKA 4 : KAKO "DEBUGGER" JAVI PRETVORBENE NAPAKE

Vendar pa nam ponujajo razvojna orodja še eno stvar, ki se ji reče „debugging“ oz. razhroščevanje, Z njeno pomočjo iščemo semantične napake, ki smo jih ustvarili in jih s pomočjo le-te popravimo. Omogoča nam, da spremljamo vrednost spremenljivk na vsakem koraku, ki mu sledi tok izvajanja programa. Zelo dobra stvar je, da lahko zamenjamo dele kode med samim izvajanjem programa, kar pomeni, da ne izgubljamo časa z nepotrebnim ugašanjem in zaganjanjem aplikacije. Tudi če za kakšno pretvorbo besedila v številko pozabimo zaščito, se program ne bo sesul in zaprl, mi pa ne bomo vedeli, kaj se je zgodilo. Program bo javil napako, njen tip in kraj, kjer se nahaja v naši kodi. Omogoča nam nemoteno izvajanje aplikacije razen na prekinitvenih točkah, kjer hočemo, da počaka. Sami lahko preverimo stanje spremenljivk in nato nadaljujemo izvajanje ročno ali pa se aplikacija nadaljuje do naslednje prekinitvene točke (angl. Break points).



SLIKA 5 : PRIKAZ DELOVANJA PREKINITVNE TOČKE



SLIKA 6 : PRIKAZ STANJA SPREMENLJIVK

Seveda je še veliko drugih razvojnih orodij, kot so Eclipse, NetBeans in tako dalje. Toda vsi trije so odprtokodni in zato nimajo tako hitrega in natančnega razvoja, kot ga ima VS. Je pa res, da jih razvijajo programerji za lastno uporabo, zato ta orodja ponujajo določene funkcije, ki jih VS nima.

SLIKA 9 : NETBEANS IDE

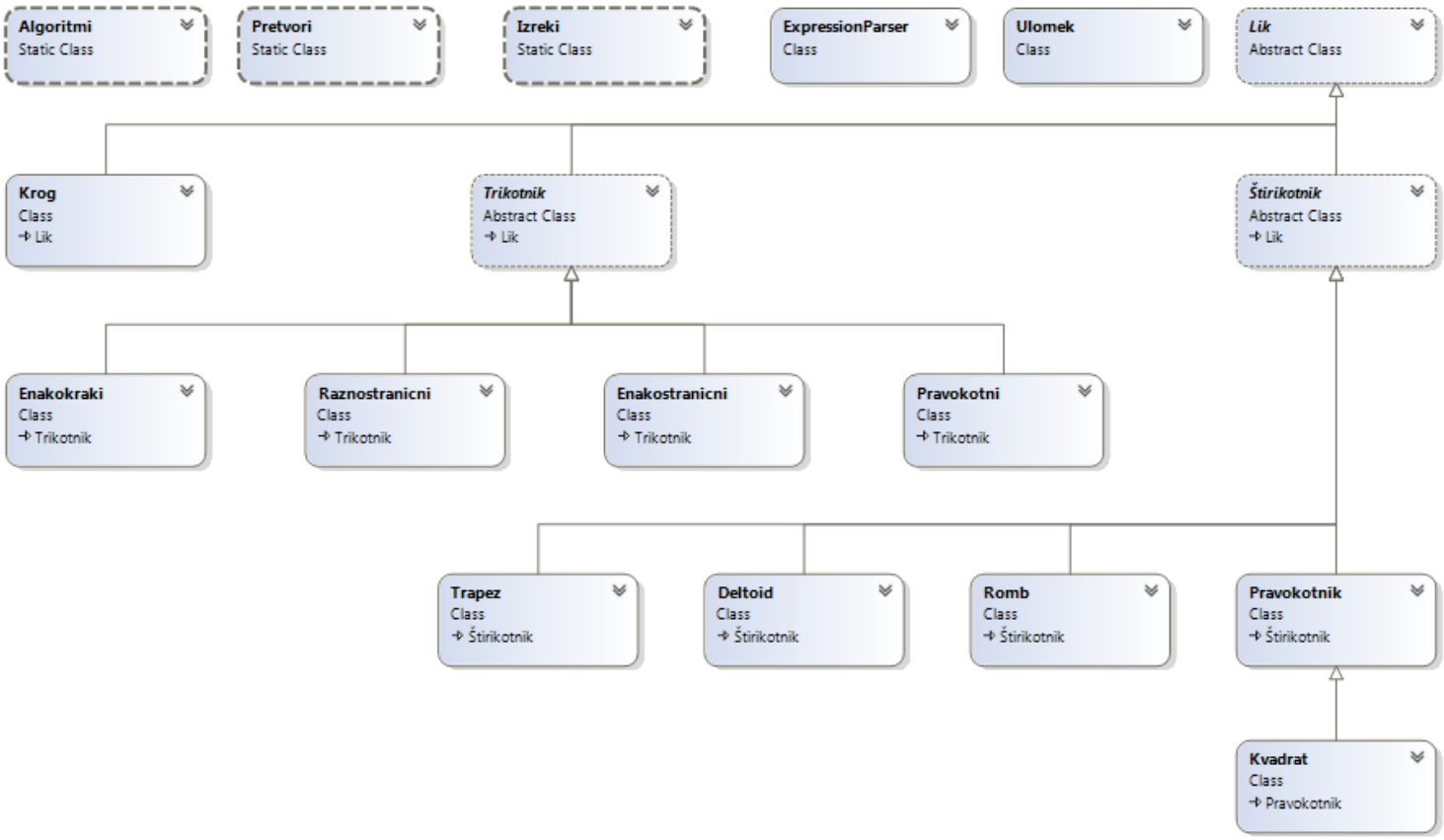
SLIKA 7 : VISUAL STUDIO

SLIKA 8 : ECLIPSE IDE

4.2 IZDELAVA KNJIŽNICE ZA APLIKACIJO

4.2.1 ZASNOVA

Celotna knjižnica temelji na razširljivosti in objektnem načinu programiranja. Vsebuje nekaj statičnih razredov, kot so Algoritmi, Izreki ter Pretvori. Ti vsebujejo metode, ki jih uporabljamo drugje v programu za izračunavanje, pretvarjanje ter iskanje določenih rezultatov ali pretvarjanje med podatkovnimi tipi. Tukaj so še liki, ki so med seboj povezani, zato je narejeno dedovanje za najmanjše možno ponavljanje kode. Iz spodnje sheme je lepo razvidno, kako se razredi med seboj dedujejo in si sledijo. Vsebuje še leksikalni razčlenjevalnik matematičnih izrazov (ExpressionParser), ki temelji na sistemu „ranžirne postaje“, in razred Ulomek, ki je v resnici poseben tip spremenljivke (angl. DTO – Data Transfer Object), ki nam omogoča delo z ulomki. Iz te celotne sheme bom podrobneje razložil razčlenjevalnik ter razred Ulomek, saj sta v primerjavi z drugimi drugačna in bolj zahtevna.



SLIKA 10 : SHEMA KNJIŽNICE

4.2.2 LEKSIKALNI RAZČLENJEVALNIK

4.2.2.1 IDEJA RAZČLENJEVALNIKA

Ranžirni algoritem (angl. Shunting Yard Algorithm) je metoda za pretvarjanje matematičnih izrazov iz infiksne notacije matematičnih izrazov (2 + 2) v postfiksno notacijo (2 2 +). Algoritem je sestavil nizozemski računalniški inženir Edsger Dijkstra.

Algoritem sprejme v osnovi matematični izraz v infiksni notaciji, kot jo zapisujemo ljudje.

Oblika matematičnega izraza: operand operator operand operator operand oz. 2 + 2 + 2

Pretvorjen izraz v postfiksni notaciji: operand operand operator operand operator oz. 2 2 + 2 +

V naslednjem koraku ga pretvori v postfiksno notacijo.

Algoritem temelji na uporabi skladov, v katere nalagamo operatorje in operande.

Algoritem se izvaja tako, da iz izraza, ki ga dobi, vzame en znak in potem prepozna, ali je operand ali operator, in ga premakne na pravilni sklad. Ta algoritem pozna prednost operatorjev, kar pomeni, da imajo oklepaji prednost pred vsemi ostalimi operatorji in si nato sledijo v naslednjem vrstnem redu:

- množenje,
- deljenje,
- seštevanje,
- odštevanje.

Zaradi tega ta algoritem v računalništvu spada v kategorijo »Razčlenjevanje s prednostjo operatorjev«.

Razčlenjevalnik deluje po sistemu, ki postavlja operande in operatorje na dva različna sklada. Kot vnosni parameter potrebuje matematični izraz tipa „string“, ki ga predeluje znak za znakom. Ko predela vse znake v izrazu, mora po pravilnosti predelati še vse operatorje, ki jih med zlaganjem operandov še ni prenesel na sklad operandov. Zaradi tega, ker ta algoritem pozna oklepaje, je potrebno po nekem zaporedju pravilnosti te operatorje med seboj prestaviti tako, da pridejo na pravilno mesto. Medtem ko smo prestavljali operatorje na pravilna mesta, smo za vsakega

pripravili še dva operanda. Skupen sklad operandov ter operatorjev je zdaj v takšni obliki – operand, operand, operator, operand, operator itd. Iz tega sklada beremo operande in jih shranjujemo na poseben sklad, dokler ne naletimo na operator. Nato iz sklada vzamemo zadnja dva operanda, izračunamo in vrnemo rešitev na sklad z operandi. Tako se ta algoritem za računanje ponavlja, dokler ne ostane samo en operand in nobenega operatorja. Zadnji operand se vrne kot rešitev enačbe. Vrne tudi zapis v postfiksni notaciji, ki ga sam računalnik ne bi znal več prebrati, kajti v osnovi je bil ta algoritem mišljen samo za cela števila od 0 do 9. Ko pa se pojavijo večmestna števila, je notacija, ki jo bere računalnik, berljiva le, dokler ostane na skladu. Beremo jo iz sklada, kajti po izpisu notacije se meje med operandi zabrišejo, razen če bi spredaj izpisovali operator, ki določa pozitivnost in negativnost operanda, kar pa bi spet začelo povzročati težavo pri ločevanju operatorja za operacijo in operatorja, ki pove, ali je operand pozitiven ali negativen.

4.2.2.2 PRIMER DELOVANJA RAZČLENJEVALNIKA V PROGRAMU

Vnosni matematični izraz: 2 * 2 / 5 - 2

Matematični izraz	Operandi	Operatorji	Računalniška notacija	Operacija algoritma
2 * 2 / 5 - 2				Prebere 2 in prestavi na sklad operandov.
* 2 / 5 - 2	2		2	Prebere * in prestavi na sklad operatorjev.
2 / 5 – 2	2	*	2	Prebere 2 in prestavi na sklad operandov.
/ 5 - 2	22	*	22	Prebere / in prestavi na sklad operatorjev. Ker sta 2 operatorja, preveri, kateri ima prednost in ga prestavi na sklad operandov.
5 – 2	22*	/	22*	Prebere 5 in prestavi na sklad operandov.
-2	22*5	/	22*5	Prebere - in prestavi na sklad operatorjev. Ker sta 2 operatorja, preveri, kateri ima prednost in ga prestavi na sklad operandov.
2	22*5/	-	22*5/	Prebere 2 in prestavi na sklad operandov.

	22*5/2	-	22*5/2	Ker ni več znakov v matematičnem izrazu, prestavi še ostale operatorje na sklad operandov.
	22*5/2-		22*5/2-	Matematični izraz v infiksni notaciji pretvorjen v obliko lažje berljivo računalniku 22/5*2- .

Tabela 1: Natančen prikaz poteka razčlenjevalnika

To je prvi del algoritma, ki pretvori enačbo v postfiksni zapis, zdaj pa poglejmo, kako računalnik to bere in izračunava.

Postfiksni izraz	Operandi	Operacija algoritma
22*5/2-		Prebere 2 in ga prestavi na sklad operandov.
2*5/2-	2	Prebere 2 in ga prestavi na sklad operandov.
*5/2-	22	Prebere * in ker sta 2 operanda, ju vzame iz sklada ter izvede operacijo, rezultat pa vrne nazaj na sklad.
5/2-	4	Prebere 5 in ga prestavi na sklad operandov.
/2-	45	Prebere / in ker sta 2 operanda, ju vzame iz sklada ter izvede operacijo, rezultat pa vrne nazaj na sklad.

2-	0,8	Prebere 2 in ga prestavi na sklad operandov.
-	0,82	Prebere – in ker sta 2 operanda, ju vzame iz sklada ter izvede operacijo, rezultat pa vrne nazaj na sklad.
	-1,2	Ker ni več znakov v izrazu, vrne rezultat 3.

Tabela 2: Natančen prikaz izračunavanja enačbe v postfiksni notaciji

Torej iz tega sledi, da je: 2 * 2 / 5 - 2 = -1,2

Algoritem gre skozi matematični izraz v infiksni notaciji rekurzivno, dokler se izraz ne konča. Nato do konca sestavi zapis v postfiksni notaciji, gre zopet rekurzivno skozi celoten izraz in ga izračunava, dokler ne ostane samo 1 operand, kar označuje, da je konec. Zadnji operand je rešitev te enačbe.

Osnovo za algoritem sem napisal iz opisa algoritma na WikiPedii, ki je prikazoval, kako program obravnava matematični izraz v infiksni notaciji.

Ta algoritem sem razširil tako, da uporablja realna števila in mu dodal še možnost določanja pozitivnosti ali negativnosti člena ter možnost korenjenja na katerokoli stopnjo. Ta funkcionalnost zahteva dosti pogojev in lažje bi bila rešljiva z razčlenjevalnikom, ki uporablja lastni slovar ter tako rečeno slovnico (angl. grammar). Za to pa bi bila potrebna uporaba naprednejših struktur, kot so slovar (angl. dictionary) ter regularni izrazi (angl. regular expressions).

4.2.3 ULOMEK

Ta razred je dejansko samo neki podatkovni tip, s katerim lahko prenašamo vrednost ulomka, ponuja pa nam tudi določene operacije, ki so posebne oz. drugačne kot pri drugih tipih, saj ta tip objekta nima ene vrednosti, po kateri se mora primerjati, ampak dve, zatorej je treba vsa primerjanja preobložiti in spremeniti. Tu se lahko pojavi problem, ker sta dva ulomka lahko enaka, a imata različna števca ter imenovalca, za kar je treba uporabiti Evklidov algoritem, da se izračuna skupni imenovalec, in šele nato vidimo, če sta števca enaka. Takemu razredu, ki je nov podatkovni tip (angl. Data Transfer Object), je treba hkrati dodati še metode, s katerimi lahko urejamo, nastavljamo in upravljamo.

```
public static bool operator !=(Ulomek a, Ulomek b)
{
    double imenovalec = Algoritmi.Evklidov_Algoritem(a.imenovalec, b.imenovalec), stevec1 = 0, stevec2 = 0;
    if (imenovalec >= a.imenovalec)
        stevec1 = (a.imenovalec / imenovalec);
    else
        stevec1 = (imenovalec / a.imenovalec);
    if (imenovalec >= b.imenovalec)
        stevec2 = (b.imenovalec / imenovalec);
    else
        stevec2 = (imenovalec / b.imenovalec);
    if (stevec1 != stevec2)
        return true;
    else
        return false;
}
```

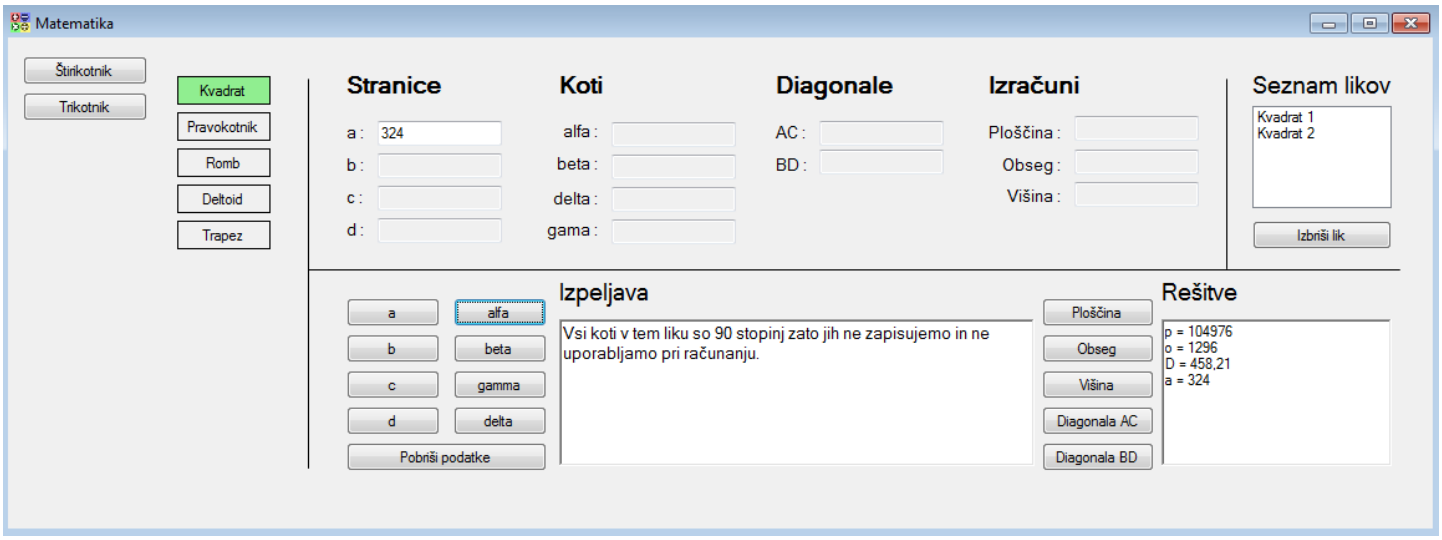
SLIKA 11 : PRIKAZ OPERATORJA V RAZREDU ULOMEK

4.2.4 NAMEN KNJIŽNICE

Namen knjižnice je, da jo lahko za menoj uporabi kdorkoli, saj deluje neodvisno od uporabniškega vmesnika. Je določen nabor funkcij, ki so vnaprej pripravljene za uporabo. Pomembna stvar pri projektu, ki vsebuje uporabniški vmesnik ter knjižnico je to, da nikoli ne pozabimo, kaj spada v uporabniški vmesnik in kaj v knjižnico. Če ju na katerikoli točki združimo, kršimo osnovna pravila in s tem knjižnica ne postane več 100-odstotno samozadostna, saj uporabniški vmesnik prevzame določene lastnosti knjižnice, ki jih ne bi smel. Strogo ločevanje teh delov nam omogoča, da lahko knjižnico uporabimo sami drugje in pa hkrati omogočimo drugim, da uporabijo naše delo v svojih projektih. Praviloma naj bi knjižnico ter uporabniški vmesnik izdelovali ločeno, še bolje pa je, če ju izdelujejo različni programerji, saj s tem zelo zmanjšamo verjetnost, da bi se katera koli metoda, funkcija in lastnost prenesla iz enega projekta v drugega.

4.3 IZDELAVA UPORABNIŠKEGA VMESNIKA

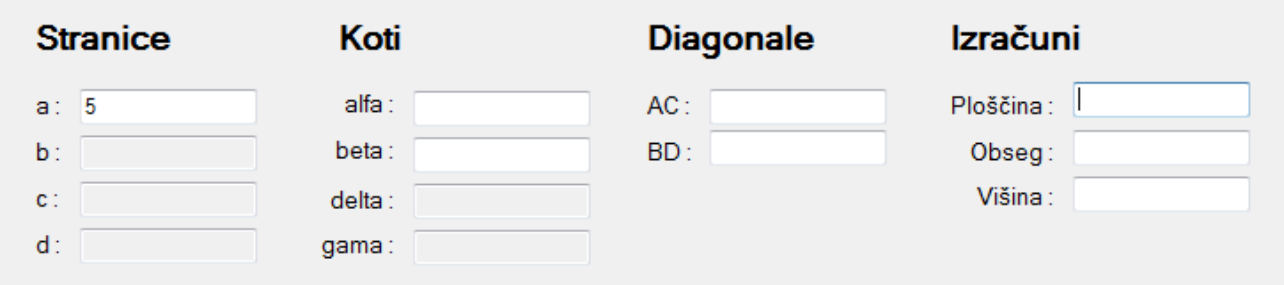
Drugi del mojega projekta je izdelava uporabniškega vmesnika, ki popolnoma izkoristi zmožnosti knjižnice, ki sem jo napisal, hkrati pa tudi doprinesel nekaj posebnega, kar bi uporabnikom pomagalo pri učenju. Zato sem poleg tega, da vmesnik izkorišča knjižnico, naredil dokaj preprost sistem, po katerem glede na to, kaj uporabnik vnese v sistem, usmerja in ponudi preostale možnosti, ki jih lahko vpisujemo, dokler ne vpišemo toliko stvari, da bi kakršen koli dodaten vnos bil nepotreben, saj bi z danimi podatki vmesnik uporabil vse enačbe, ki jih vpisuje knjižnica. To funkcijo sem poimenoval »selektivni vnos«.



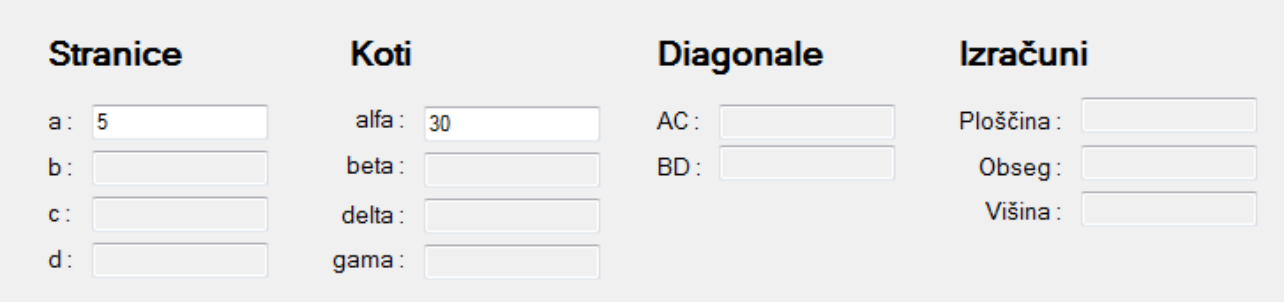
SLIKA 12 : UPORABNIŠKI VMESNIK

4.3.1 SELEKTIVNI VNOS

To je sistem, ki pomaga uporabniku pri vpisovanju podatkov za določen lik, telo itd. Je zelo enostavno strukturiran sistem odločanja glede na določene pogoje, ki so v tem primeru vpisani podatki. Ima drevesno strukturo, ki se z vsakim nivojem manjša ter ponuja manj možnosti, dokler ne pristane na dnu svoje veje. Ko pristane na dnu veje, ne moremo vnesti nobenega podatka več in lahko uporabljamo samo še gumbe za izračunavanje.



SLIKA 13 : VSE MOŽNOSTI ŠE ODPRTE



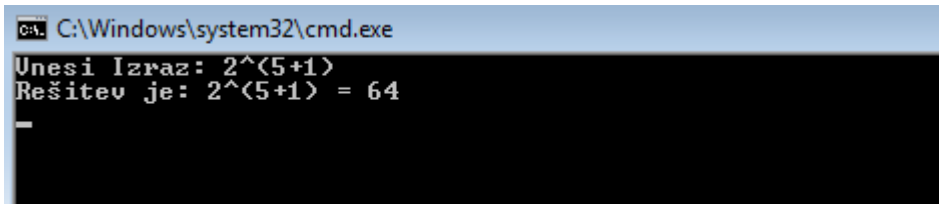
SLIKA 14 : MOŽNOSTI SE ZAPREJO KER JE DOVOLJ PODATKOV

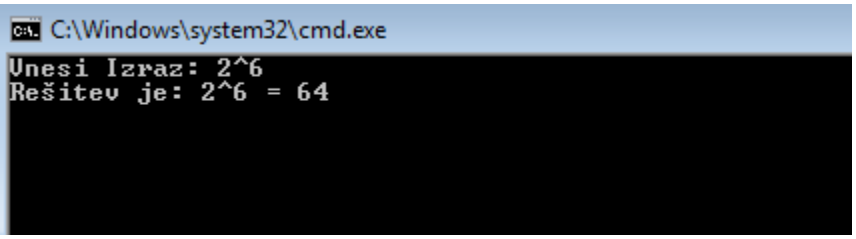
4.3.2 IZPISOVANJE POSTOPKA

Ker naj bi nas aplikacija učila, kako računati, in ker knjižnica vsebuje vse potrebne enačbe, lahko za vsako enačbo izpisujemo popolni postopek od osnovne enačbe do vstavljanja podatkov ter prikazovanjem rezultata. Takšna stvar pokaže uporabniku, kako ter na kakšen način se pride do rešitve.

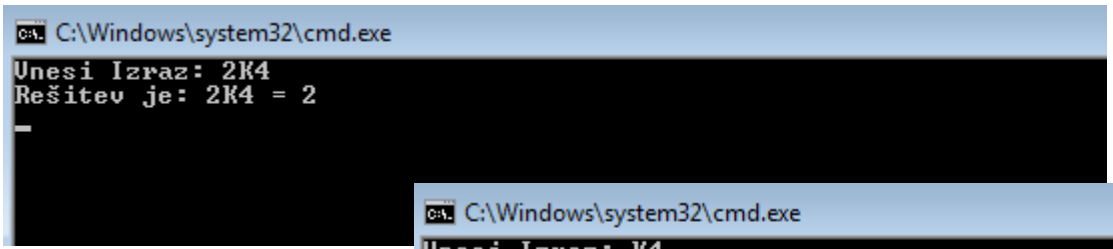
4.4 NAVODILA ZA UPORABO VMESNIKA

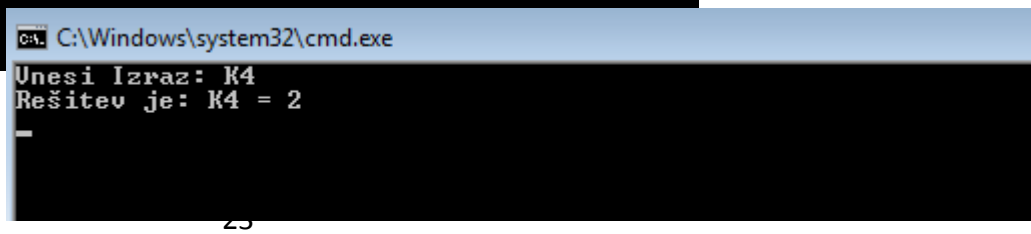
Načeloma bi moral razčlenjevalnik brati pravilne matične izraze, ampak na tipkovnici nimamo možnosti, da bi nadpisali potenco ali napisali koren. Zato uporabljamo na tipkovnici drugačne znake, kot so recimo »^«, ki se uporablja za potenco, ter »K«, ki se uporablja kot koren. Ker algoritem pozna oklepaje in glede na njih tudi pravilno sortira, lahko zapisujemo potence, ki se same po sebi izrazijo takole:

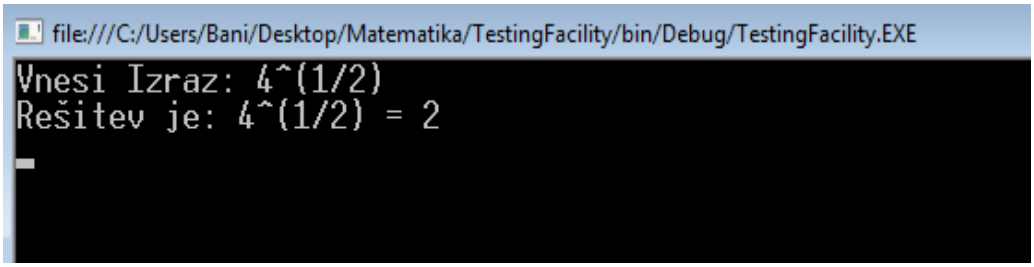




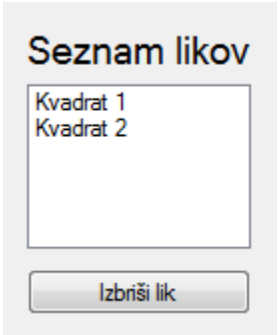
Za zapisovanje korenov imamo dve možnosti. Ker pri korenih druge stopnje te stopnje ne zapisujemo, lahko v aplikaciji, namesto da bi zapisali »2K4«, zapišemo »K4«, kar pomeni drugi koren od 4, rešitev je 2, vsak koren pa lahko zapišemo tudi s potenco »4^(1/2)«, »2« je stopnja v tem zapisu.





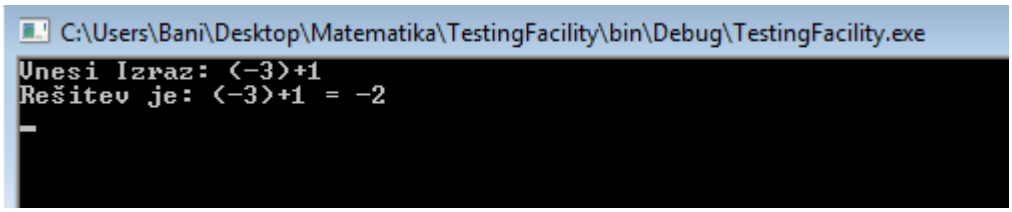


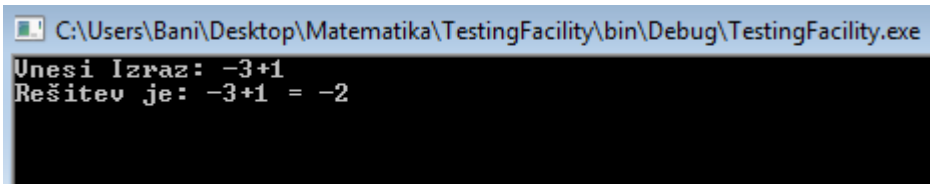
Vsakič, ko kliknemo na katerikoli lik, se ustvari nov primerek tega lika, ki nima nobenih izračunov ter rešitev ali podatkov. Vidimo ga lahko v okenčku, ki je na desni zgornji strani. Zraven je prejšnji primerek lika, če smo ga že imeli. Če kliknemo na prejšnji primerek, se vam bodo v vsa polja samodejno vnesle vrednosti prejšnjega lika, vsi izračuni in zadnje stanje izpeljave. Zato lahko like pri majhnem spreminjanju podatkov med seboj primerjamo.



Biti moramo previdni, kajti vsakič, ko spremenimo kakšen podatek v liku, se rešitve za ta lik izbrišejo, ker postanejo neveljavne, kar preprečuje, da bi uporabnik imel kakršnokoli napačno rešitev za vnesene podatke.

Na voljo so gumbi za izračunavanje vseh 4 stranic pri štirikotniku, ampak ker so v kvadratu vse 4 enake, se, namesto, da bi pokazalo »b = a«, izpiše nasvet/nauk, ki je že vnesen v knjižnico.





5. MOŽNOSTI RAZŠIRITVE, IZBOLJŠANJA APLIKACIJE

Knjižnico je možno izboljšati z vnosom ter izdelavo še več likov, vnosom še kakšnega algoritma ali izreka, največja možnost izboljšave pa bi bil leksikalni razčlenjevalnik matematičnih izrazov z lastno slovnico. Takšen razčlenjevalnik bi lahko podpiral vse funkcije na veliko lažji način ter z veliko manj dela pri dodajanju ali odstranjevanju funkcij.

6. ZAKLJUČEK

V tem delu sem si zastavil cilj izdelati program, ki bo pomagal uporabniku pri učenju geometrije, omogočal uporabniku vnos matematičnih izrazov, selektivni vnos in shranjevanje ter primerjanje likov. Dosegel sem vse cilje, saj program omogoča, da lahko uporabnik vnese matematični izraz, ki vsebuje osnovne aritmetične operatorje. Ko uporabnik vpiše podatek, mu program zapre vse možnosti za vnos, ki niso več možne, s tem pa ga usmerja k vpisovanju podatkov, da program izračuna vse, kar je mogoče. Uporabnik lahko s pomočjo seznama likov, ki se shranjujejo samodejno na vsaki spremembi, primerja prejšnjega in sedanjega ter tako vidi razliko med rešitvami.

Moja hipoteza je bila, da lahko s pomočjo uporabniškega vmesnika pomagamo uporabniku tako, da lažje dela z geometrijskimi liki in se pri tem hkrati uči iz rešitev, ki mu jih program vrača. To hipotezo sem dokazal, saj program izpisuje celoten potek računanja od osnovne enačbe, kakšni so podatki, vstavljeni v to enačbo, do končne izpeljave rešitve, rezultat te izpeljave pa mu shrani posebej in mu omogoča tudi primerjanje likov.

7. VIRI IN LITERATURA

- *Shunting Yard Algorithm (online). Wikipedia: The freee encyclopedia (Nazadanje preverjeno 14.3.2013). Dostopno na spletnem naslovu http://en.wikipedia.org/wiki/Shunting-yard_algorithm*

- *C# .NET(online). Srečo Uranič (Nazadnje preverjeno 14.3.2013). Dostopno na spletnem naslovu http://uranic.tsckr.si/C%23/C%23.pdf*

- *Kozak J. Numerična in računalniška matematika. Algoritmi v nalogah numerične aproksimacije. Ljubljana : Inštitut za matematiko, fiziko in mehaniko 1983. (Matematika : raziskovalna naloga)*

- *Alfa 3 : zbirka nalog za matematiko v 3. letniku srednjega tehniškega oz. strokovnega izobraževanja. 1. Izd 8.natis. Ljubljana : Ataja ,2009 (Matematika - Vaje za srednje šole)*

8. ZAHVALE

Zahvaljujem se vsem profesorjem za spodbudo, usmerjanje ter pomoč skozi vsa štiri leta, v katerih sem pridobival znanje, ki ga imam, ter ga zdaj uporabil pri izdelavi te aplikacije. Še posebej se zahvaljujem profesoricama, ge. Heleni Klepej Viher ter ge. Nataši Besednjak, za pomoč pri enačbah za geometrijske like ter mag. Boštjanu Resinoviču, ki me je kot mentor usmerjal po pravilni poti ter mi pomagal pri premagovanju težav, ki sem jih imel pri izdelavi aplikacije.