Izrada znanstvenog kalkulatora u programskom jeziku Visual Basic

Žentil, Berislav

SVEUČILIŠTE U RIJECI
Studij politehnike

Preddiplomski sveučilišni studij politehnike

Berislav Žentil

Izrada znanstvenog kalkulatora u programskom jeziku Visual Basic

Završni rad

Mentor: Izv. Prof. Dr. Sc. Marko Maliković

Rijeka, 2022

UNIVERSITY OF RIJEKA
School of Polytechnics

Undergraduate study of Polytechnics

Berislav Žentil

Creating a scientific calculator in the Visual Basic programing language

Bachelor thesis

Supervisor: Izv. Prof. Dr. Sc. Marko Maliković

Rijeka, 2022

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svojim kolegama i prijateljima s fakulteta koji su bili uz mene kroz preddiplomski studij i koji su mi pomogli kada mi je bilo potrebno, također se zahvaljujem svim profesorima koji su mi predavali kroz studij i koji su me naučili znanjima iz politehnike koja posjedujem.

Berislav Žentil

SVEUČILIŠTE U RIJECI
Studij politehnike
Rijeka, 15. veljače, 2022.

Zadatak za završni rad

Pristupnik: **Berislav Žentil**

Naziv završnog rada: **Izrada znanstvenog kalkulatora u programskom jeziku Visual Basic**

Naziv završnog rada na eng. jeziku: **Creating a scientific calculator in the Visual Basic programming language**

Sadržaj zadatka: Opišite što je to znanstveni kalkulator i koje funkcije za uporabu u znanosti, matematici i inženjerstvu je poželjno da ima. Opišite što je to programiranje upravljano događajima. Opišite svojstva programskog jezika Visual Basic. Opišite razvojnu okolinu koju koristite za izradu programa. Opišite tipove podataka i konstanti koje postoje u programskom jeziku Visual Basic. Opišite načine deklariranja, inicijaliziranja i korištenja varijabli u programskom jeziku Visual Basic. Opišite aritmetičke operatore koji se koriste u programskom jeziku Visual Basic. Opišite naredbe za kontrolu programa i matematičke funkcije koje koristite u svom programu. Opišite forme i kontrole koje koristite u vašem programu (prozore, gumbe, natpise, okvire s tekstom, kontrolne kućice, …). Opišite mogućnosti unaprjeđenja programa s obzirom na završnu verziju vašega programa. Priložite cjelokupni programski kôd u programskom jeziku Visual Basic.

Mentor: Izv. prof. dr. sc. Marko Maliković

Malković Marko

(potpis mentora)

Voditelj za završne radove

Zadatak preuzet: 14.3.2022.

Be ŽS

(potpis pristupnika)

Sadržaj

Sadržaj I
POPIS SLIKA II
POPIS TABLICA III
SAŽETAK IV
SUMMARY V
1. UVOD 1
2. Znanstveni kalkulator 2
2.1 Opis znanstvenog kalkulatora 2
2.2. Funkcije poželjne za znanstvene kalkulatore 3
3. Programiranje 5
3.1. Programske paradigme 5
3.2. Programiranje upravljano događajima 6
4. Programski jezik Visual Basic 9
4.1 Značajke Visual Basica 9
4.2 Razvojna okolina Visual Studio 9
4.3 Varijable i konstante u programskom jeziku Visual Basic 14
4.5 Operacije u Visual Basic-u 17
4.6 Znanstveni kalkulator izrađen u programskom jeziku Visual Basic 18
5. ZAKLJUČAK 26
LITERATURA 27

POPIS SLIKA

Slika 1. Citizen SR-135N kalkulator....2
Slika 2. Dijagram toka programa upravljanog događajima....6
Slika 3. Dijagram toka imperativnog programa....8
Slika 4. Grafičko sučelje u programskom jeziku Visual Basic....10
Slika 5. Alatna kutija (toolbox) u grafičkom sučelju Visual Basic-a....11
Slika 6. Forma s gumbom....11
Slika 7. Razlika između gumbova....12
Slika 8. Forma za program znanstvenog kalkulatora....12
Slika 9. Tekstualno sučelje programskog jezika Visual Basic....13
Slika 10. Prikaz forme za znanstveni kalkulator s prikazanim elementima....19

POPIS TABLICA

Tablica 1. Tipovi podataka u Visual Basicu 15
Tablica 2. Aritmetički operatori u programskom jeziku Visual Basic.....18

SAŽETAK

Ukratko sam opisao znanstvene kalkulatore, njihove mogućnosti i primjene. Naveo sam matematičke funkcije koje on može imati te opisao izgled i izvedbu znanstvenog kalkulatora ovisno o primjeni te sam spomenu kako znanstveni kalkulatori nemaju sve od mogućih funkcija koje mogu imati. Potom sam prikazao različite programske paradigme koje mogu biti korištene u programiranju i detaljnije sam opisao programiranje upravljano događajima jer je programski jezik Visual Basic jezik u kojem se upravo tako programira i za kraj je opisan sam jezik Visual Basic, njegova sučelja načini programiranja i matematičke funkcije koje on sadrži te je opisan program za znanstveni kalkulator i prikazan programski kod za isti.

Ključne riječi: Kalkulator, Matematičke funkcije, Programiranje, Visual Basic

SUMMARY

I explained scientific calculators, their abilities and applications, brought up the mathematical functions one may have, described their looks and implementation of scientific calculators based on application and I noted that they may not have all of the possible functions they can have. After that i described different programming paradigms that can be used in programming and gave a more detailed description of event driven programming because it is the programming paradigm that the programming language Visual Basic uses and in the end I described the programming language Visual Basic its user interfaces, mentioned the mathematical functions it uses and described the program for a scientific calculator with the code for it.

Keywords: Calculator, Mathematical functions, Programming, Visual Basic

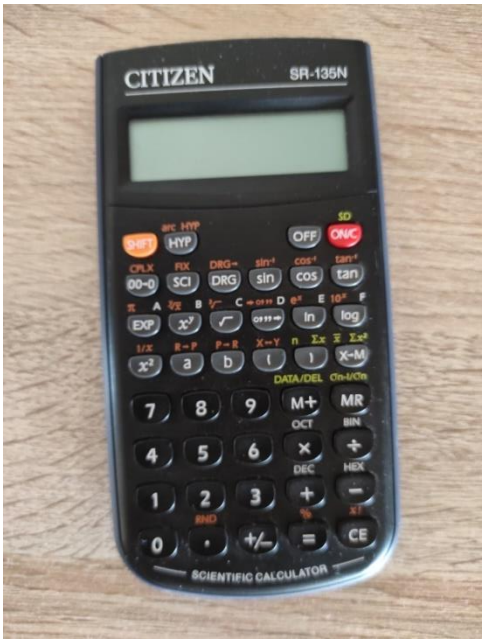
1. UVOD

U ovome radu, istražio sam znanstvene kalkulatore i matematičke funkcije koje koriste i koje su poželjne kod znanstvenih kalkulatora. Potom sam u programskom jeziku visual basic izradio program koji služi kao znanstveni kalkulator. Izrađeni kalkulator koristi neke ali ne i sve poželjne funkcije za znanstveni kalkulator. Opisan je programski jezik Visual Basic u kojem je izrađen sam kalkulator. Visual Basic je dosta moćan jezik s mogućnostima za brzu izradu programa i brzo učenje rada u njemu i programiranja općenito te je pogonjen događajima što znači da će se program odvijati prema određenim pravilima i ovisiti o onome što korisnik radi s programom i o onome što se događa u samome programu.

2. Znanstveni kalkulator

2.1 Opis znanstvenog kalkulatora

Znanstveni kalkulatori su elektronički uređaji čija je osnovna svrha vršenje izračuna kompleksnih matematičkih operacija. Koriste se u znanstvene i edukacijske svrhe, a ovisno o potrebama mogu imati različite funkcije i biti različitih dimenzija, od dovoljno malih da stanu u džep pa do vrlo velikih kalkulatora s vrlo velikim grafičkim displejem za iscrtavanje grafova (grafički kalkulatori). Najčešća uporaba znanstvenih kalkulatora je kod obavljanja vrlo kompleksnih matematičkih funkcija, ali koriste se i za računanje vrlo velikih te vrlo malih brojeva nrp u astronomiji, fizici ili kemiji i naravno mogu se koristiti za računanje kao i obični kalkulatori. Znanstveni kalkulatori izvedeni su pomoću integriranih krugova odnosno čipova. Ovisno o tome kakav čip se koristi kalkulatori mogu biti vrlo jednostavni ili vrlo složeni i imati različite mogućnosti i funkcije. Kalkulatori kao i svaki drugi elektronički uređaj koriste binarni sustav za zapisivanje brojeva i za odabir naredbi odnosno matematičkih funkcija koje se će koristiti u izračunu. Ovakva izvedba moguća je radi tranzistora koji se nalaze unutar integriranih krugova, koristeći tranzistore binarne znamenke 0 i 1 lako se mogu ostvariti tako da tranzistori provode (1) ili ne provode (0) električnu struju. Na sljedećoj slici prikazan je znanstveni kalkulator Citizen SR-135N.



Slika 1. Citizen SR-135N kalkulator

Na slici 1 vide se funkcije koje ovaj kalkulator posjeduje. Citizen SR-135N je ispodprosječan u odnosu na druge znanstvene kalkulatore. Ovaj kalkulator koristi 8 segmentni displej za prikaz znamenki i decimalne točke no znanstveni kalkulatori obično imaju grafičke displeje kako bi lakše prikazivali razlomke i znakove kao što su plus, minus, puta, podijeljeno, logaritam itd. Iz toga vidi se jednostavnost tog kalkulatora no on se i dalje može koristiti u edukacijske svrhe ili u nekim jednostavnijim područjima gdje nema neke velike potrebe za složenim matematičkim funkcijama ili izrazima.

2.2. Funkcije poželjne za znanstvene kalkulatore

Elektronički kalkulatori počeli su s nekoliko samo par funkcija odnosno zbrajanje, oduzimanje, množenje i dijeljenje no kako su se razvijali integrirani krugovi tako su moguće funkcije koje su izvedive na kalkulatorima postajale sve brojnije i brojnije. Funkcije koje su izvedive na današnjim kalkulatorima su:

- Osnovne matematičke operacije (zbrajanje, oduzimanje, množenje, dijeljenje)
- Znanstveni zapis velikih i malih brojeva
- Decimalna aritmetika (računanje s decimalnim brojevima)
- Logaritamske funkcije (računanje logaritama s različitim bazama)
- Trigonometrijske funkcije (sinus, kosinus, tangens, kotangens...)
- Eksponencijalne funkcije i korijeni
- Memorirane konstante (Pi, e i druge)
- Memoriranje i izmjene izračuna koji su uneseni u kalkulator
- Računanje s drugim brojevnim sustavima (heksadekadski, binarni, oktalni)
- Osnovna Booleova algebra
- Kompleksni brojevi
- Izračuni statistika i vjerojatnosti
- Rješavanje jednadžbi
- Rješavanje matrica
- Rješavanje derivacija i integrala
- Računanje vektora
- Pretvaranje jedinica

Osim navedenoga kalkulatori također mogu biti programabilni što daje mogućnost samostalnog unosa vlastitih funkcija. Osim toga mogu imati mogućnost iscrtavanja grafova. Naravno kalkulatori biti će prilagođeni njihovoj uporaba. Ako se kalkulator koristi za

računanje u fizici funkcije koje će imati mogu biti trigonometrijske, derivacije, integrali, kompleksni brojevi, vektori i slično ali nema potrebe da sadrži Booleovu algebru ili druge brojevne sustave. Prema tome radi jednostavnosti izvedbe i korištenja kalkulatori se prilagođavaju njihovoj namjeni i svrsi.

3. Programiranje

3.1. Programske paradigme

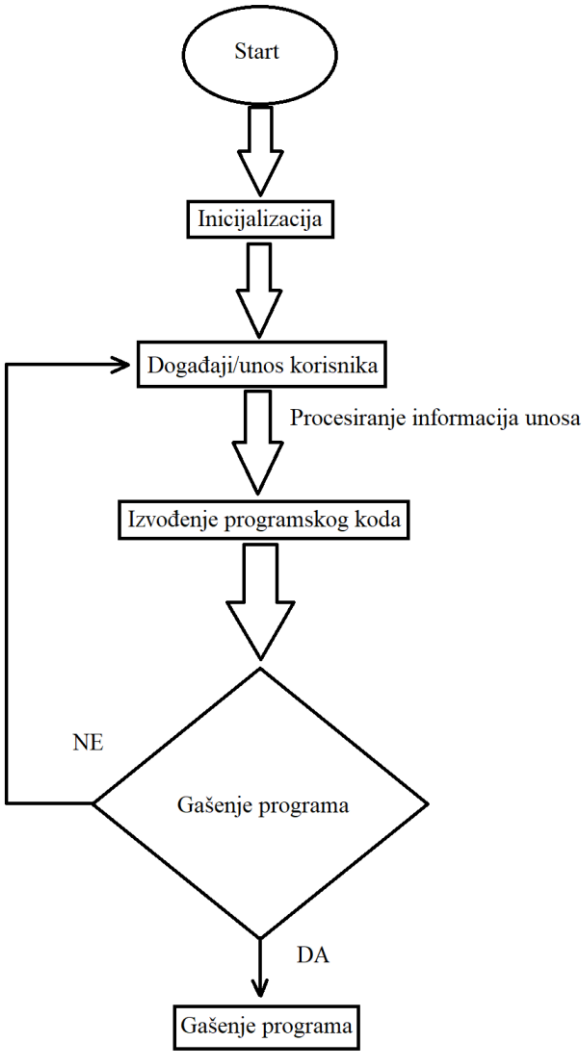
Programske paradigme zapravo su podjela programiranja prema načinu izvođenja programa. Programska paradigma određenog program ovisi o njegovoj namjeni i o programskom jeziku koji se koristi. Danas većina programskih jezika zapravo spada u više paradigmi. Najčešće programske paradigme koje poznajemo su:

- Imperativno programiranje
- Objektno-orijentirano programiranje
- Programiranje upravljano događajima
- Logičko programiranje

Imperativno programiranje je način programiranja gdje se naredbe izvršavaju određenim redoslijedom kako su napisane sve dok program ne dođe do kraja kada se izvršavanje programa završava. Objektno orijentirano programiranje je način gdje se program koristi različitim objektima, klasama i slično što omogućava bolju organizaciju podataka unutar programa. Time se rad s bazama podataka ili velikim količinama podataka puno pojednostavljuje. Objektno-orijentirano programiranje također je moguće primijeniti uz ostale paradigme i najčešće se kod imperativnog programiranja koristi i objektno-orijentirano programiranje. Programiranje upravljano događajima je takav način gdje se program izvršava tek nakon što korisnik napravi nekakav unos odnosno program čeka korisnika da napravi neku radnju prije nego što se bilo što dogodi. Program se tako izvršava ovisno o unosima korisnika dok korisnik ne ugasi program ili dok program ne naiđe na nekakvu grešku. Ovakav način programiranja može se kombinirati s objektno-orijentiranim programiranjem ukoliko je to potrebno. Ovaj način također je moguće imitirati u imperativnoj programskoj paradigmi s određenim petljama i naredbama, ali za neke komplicirane programe ovakav način izvođenja jako je loš i nepregledan i zahtjeva mnogo više truda, znanja i vremena nego ako bismo koristili programski jezik koji je namijenjen za izradu programa upravljanih događajima. Logičko programiranje je način gdje se logika koristi za tijek programa. Ovakav način koristi se za dokazivanje matematičkih teorema pomoću računala s automatskim deduktivnim sustavima.

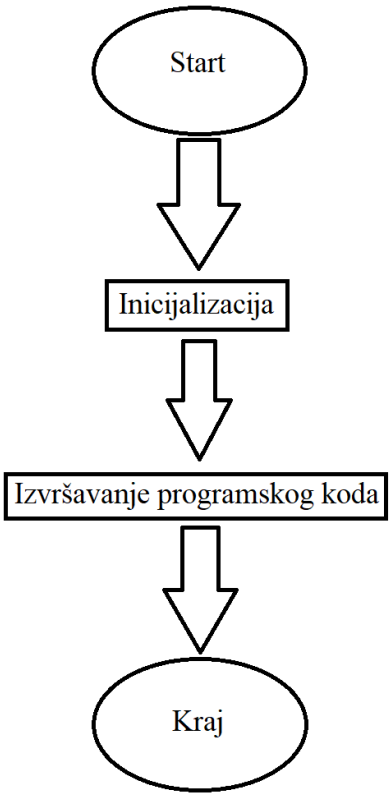
3.2. Programiranje upravljano događajima

Visual basic je programski jezik koji koristi programiranje upravljano događajima. Takvo programiranje svodi se na programe koji čekaju korisnika da nešto napravi prije nego li se išta događa. Ovisno o programu on može imati više različitih interakcija s korisnikom i mogu postojati više stvari koje se događaju nakon svake. Na primjer program prilikom pokretanja može generirati nasumičan broj od 1 do 100 te na ekranu može biti kućica u koju korisnik upisuje brojeve od 1 do 100 i program uspoređuje uneseni broj s generiranim brojem i ovisno o tome dali je veći ili manji na ekranu se može ispisati poruka. Program bi tako išao dok korisnik ne odabere broj koji je generiran i potom se program može ugasiti ili generirati novi broj ili se može dogoditi nešto sasvim drugo ovisno o tome što programer odredi u kodu samog programa. U nastavku slijedi prikaz dijagrama toka programa upravljanim događajima



Slika 2. Dijagram toka programa upravljanog događajima

Na dijagramu gore prikazan je proces rada programa upravljanog događajima, iako je dijagram jako pojednostavljen u odnosu na ono kako bi izgledao dijagram toka nekog ozbiljnijeg programa upravljanog događajima ovaj dijagram toka je dovoljan za objasnit kako programi upravljani događajima funkcioniraju. Na početku kada korisnik pokrene program (Start), program se podešava, računalo rezervira memorijske lokacije u koje će se zapisati varijable i konstante određene u programu, program zapisuje vrijednosti koje su određene u programu u za to predviđene memorijske lokacije te se prikazuje sučelje na ekranu koje može biti tekstualno (TUI – textual user interface) ili grafičko (GUI – graphical user interface) (Inicijalizacija). Potom korisnik unosi nekakvu naredbu, stišće neki gumb ili radi neku sličnu radnju koja je definirana programskim kodom (Događaji/unos korisnika), u ovoj fazi ovisno o programu može postojati vrlo veliki broj mogućnosti i nakon jednog unosa ili događaja može postojati još drugih mogućnosti za unos od strane korisnika. Kada korisnik napravi unose informacije koje je korisnik unose i koje su prethodno pohranjene u memoriju procesiraju se i na temelju toga se izvršava programski kod i u programu se izvršavaju određeni procesi koji su definirani programskim kodom (Izvođenje programskog koda). Na kraju izvođenja koda program provjerava dali je na kraju izvršavanja koda vezana uz prethodne događaja odnosno unose korisnika program gotov odnosno dali korisnik želi da se program ugasi (Gašenje programa), ako da program se gasi, ako ne program se vraća na korisnikov unos odnosno na događaje u programu. Tako na primjer unos korisnika može biti pritisak na gumb, pomicanje slidera, odabir na izborniku, unos neke vrijednosti u neku kućicu za unos ili neka druga slična radnja, a događaji mogu biti prolazak vremena, neka promjena u programu koja se desi nakon korisnikovog unosa npr promjena neke vrijednosti u programu, ispisivanje nečega na ekranu i slično. U odnosu na imperativno programiranje, programiranje upravljano događajima puno je kompleksnije i ima puno više mogućnosti. Za usporedbu s imperativnim programiranjem u nastavku prilažem dijagram toka imperativnog programa



Slika 3. Dijagram toka imperativnog programa

Na slici 3 je prikazan također jednostavan prikaz ali dovoljan da se prikaže razlika između imperativnog programiranja i programiranja upravljanog događajima. Jasno se vidi razlika da nema nikakvih dodatnih događaja ili unosa korisnika u imperativnom programiranju koji bitno utječu na program ili koji su potrebni da se program izvršava, te da se nakon izvršavanja program automatski gasi. Naravno u imperativnom programiranju moguće je omogućiti interakcije korisnika s programom usred rada samog programa s određenim naredbama i kao što sam već prethodno spomenuo moguće je s imperativnim programskim jezicima kreirati programe koji se ponašaju kao programi upravljani događajima uz pomoć petlji i određenih naredbi za unos no takvi programi bi bili prekomplicirani i koristili bi previše resursa samog računala.

4. Programski jezik Visual Basic

4.1 Značajke Visual Basica

Visual Basic je objektno orijentirani programski jezik korišten za izradu programa upravljanih događajima. Objektno orijentirani programski jezici su takvi jezici koji imaju neke određene ugrađene objekte ili klase i mogućnost izrade objekata ili klasa. Objekti i klase su posebna polja podataka koja imaju svoje osobine npr. u programskom jeziku Visual Basic postoje ugrađeni objekti kao što su: gumbovi, oznake, prozori za slike itd. Svaki od tih objekata u sebi ima zapisane neke informacije take će na primjer oznaka imati zapisan položaj oznake na ekranu, tekst oznake, dimenzije, boju i slično, dok će gumb imati u sebi zapisane druge informacije koje su potrebne za taj gumb. Osim podataka objekti mogu imati i funkcije povezane uz njih pa će tako gumb imati funkciju koja se pokreće kada se izvrši klik na taj gumb. Visual Basic je jezik koji je napravljen na bazi programskog jezika Basic koji je viši programski jezik namijenjen za brzo učenje i lako korištenje. Basic je razvijen radi tadašnjih računala koja nisu mogla bit korištena bez programiranja kako bi se korištenje računala pojednostavilo jednostavnošću samog jezika Basic s osnovnom svrhom da se korištenje računala omogući u školama i na fakultetima. Od programskog jezika Basic nastalo je mnogo drugih programskih jezika među kojima je i Visual Basic. Svi programski jezici koji su nastali na bazi Basic-a su maksimalno pojednostavljeni, laki za korištenje i namijenjeni za brzo učenje i brzu izradu programa. Kod Visual Basica programiranje je dodatno pojednostavljeno grafičkim sučeljem koje omogućuje dodatne mogućnosti kod izrade programa.

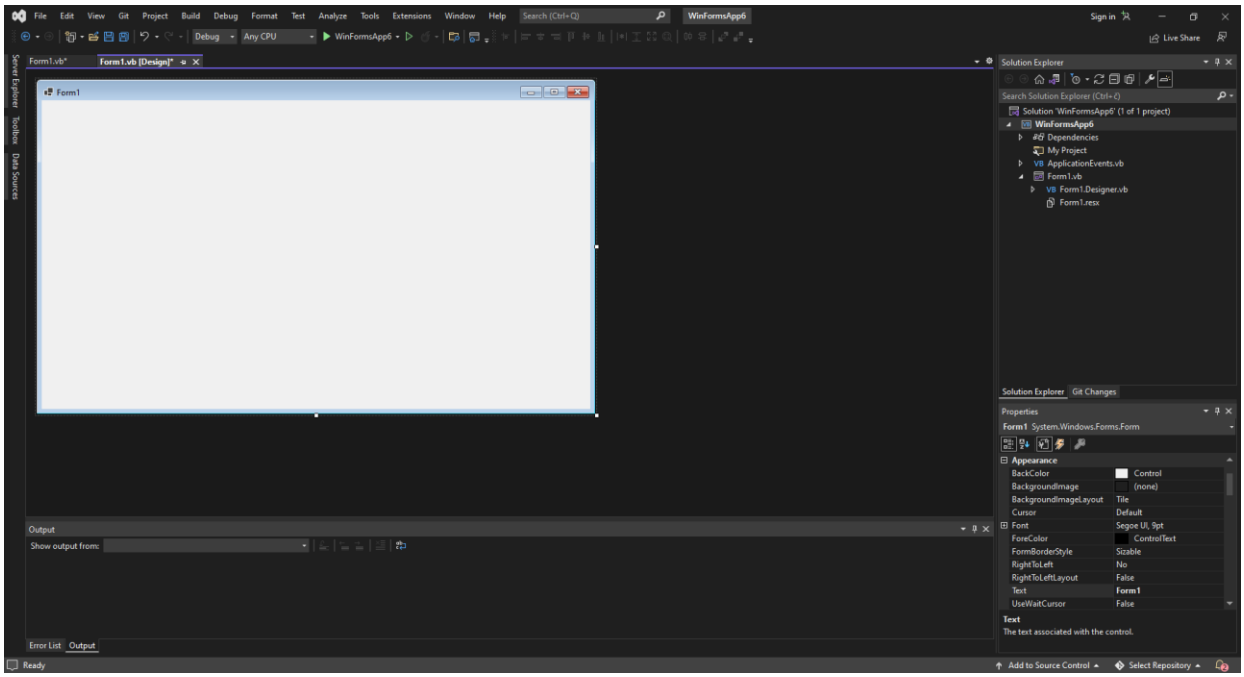
4.2 Razvojna okolina Visual Studio

Za izradu programa u programskom jeziku Visual Basic-u razvojna okolina sastoji se od 2 djela:

- Grafičkog sučelja – u grafičkom sučelju programer slaže elemente kao što su gumbi, oznake i slično u formu koja predstavlja ono što će korisnik vidjeti kada pokrene sam program. U grafičkom djelu mijenja se izgled samog programa, pozicije elemenata, njihove boje, pozadina ubacuju slike i slično no osim toga grafičko sučelje nema neku drugu funkcionalnost i funkcije samog programa ne mogu se urediti niti ostvariti u samom grafičkom sučelju.
- Tekstualno sučelje – u tekstualnom sučelju programer dodaje funkcionalnost programu, unosi funkcije, događaje i slično. Ovisno o potrebama programer

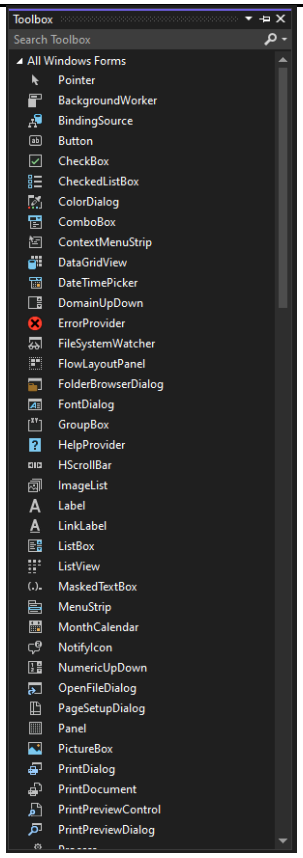
također može i kreirati nove klase ili objekte nove funkcije i može dodavati elemente koje bi inače dodavao na grafičkom sučelju ukoliko je to potrebno ili ako je lakše programeru raditi isključivo s tekstualnim sučeljem. Tekstualno sučelje je mnogo moćnije od grafičkog, ima više mogućnosti i ukoliko je programeru tako lakše cijeli program može napraviti u tekstualnom sučelju.

Iako je tekstualno sučelje mnogo moćnije od grafičkog, grafičko sučelje je bitno upravo radi same jednostavnosti programiranja u programskom jeziku Visual Basic jer je glavna svrha svih jezika baziranih na Basic-u upravo ta jednostavnost, brza izrada programa te mogućnost i jednostavnost učenja i rada s programskim jezikom i samim programiranjem. Grafičko sučelje daje upravo tu jednostavnost i mogućnost olakšanja učenja i rada s programom. U nastavku slijede prikazi grafičkog sučelja u jeziku Visual Basic.



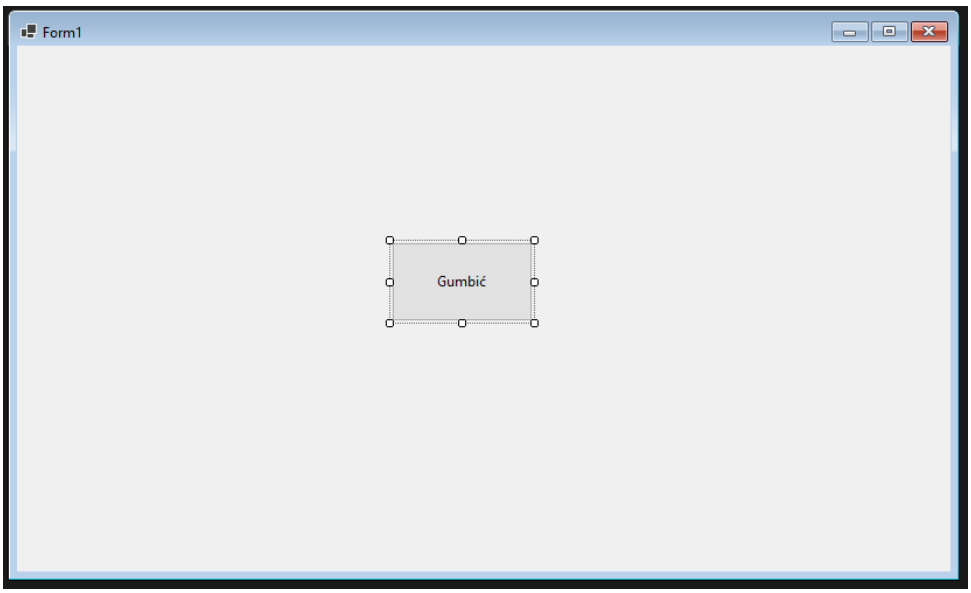
Slika 4. Grafičko sučelje u programskom jeziku Visual Basic

Na slici 4 prikazano je grafičko sučelje u Visual Basicu, na sredini sučelja nalazi se forma u koju se slažu elementi programa. Forma je ujedno i prikaz onoga što će korisnik vidjeti kada pokrene program. Na vrhu ekrana prikazane su dodatne opcije za upravljanje programom, neke od kojih su vidljive odmah, neke su skrivene unutar padajućih izbornika ili mogu biti pronađene s tražilicom. S desne strane nalaze se dodatna kontrolna ploča gdje se mogu promatrati elementi forme, programa i resursi koji su uključeni u program, a ispod toga nalaze se svojstva odabranog objekta (u ovom slučaju same forme) gdje se mogu mijenjati svojstva objekta kao što su dimenzije, boje, tekst i slično.



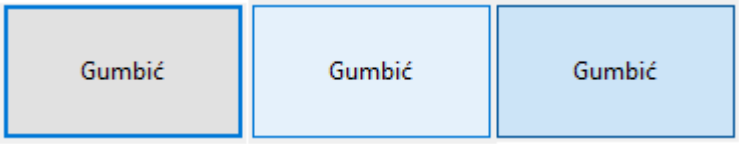
Slika 5. Alatna kutija (toolbox) u grafičkom sučelju Visual Basic-a

Na slici 5 prikazan je alatna kutija u programskom jeziku Visual Basic. U alatnoj kutiji nalaze se objekti koji su definirani u programu i koji se mogu staviti na formu. Ovdje se nalaze gumbovi i ostali elementi koje programer koristi za izradu programa i koje korisnik koristi u programu.



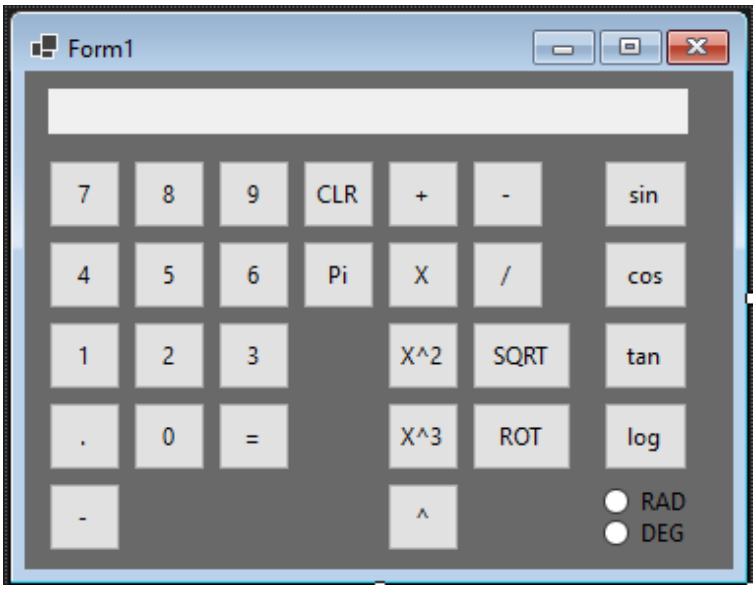
Slika 6. Forma s gumbom

Slika 6 prikazuje formu na koju je dodan gumb na koji korisnik može kliknuti u programu. Klikom na gumb dogoditi će se nešto što je određeno programskim kodom tog gumba unutar djela gdje je definiran objekt gumb kao i unutar programskog koda koji piše sam programer u tekstualnom sučelju. Osim klika na gumb korisnik može staviti pokazivač preko gumba ili držati gumb stisnut određeno vrijeme kada se također mogu odvijati određen događaji koje će programer isprogramirati ili koji su određeni unaprijed pa je tako na sljedećoj slici prikazana razlika između gumbova u različitim slučajevima.



Slika 7. Razlika između gumbova

Na slici 7 prikazana su 3 gumba, prvi gumb je običan gumb bez ikakve interakcije od strane korisnika, drugi gumb je gumb preko kojega je stavljen pokazivač, a treći gumb je gumb na kojega je korisnik kliknuo ili kojeg drži kliknutog. Ove interakcije s gumbom definirane su u samome programskom jeziku no programer može to promijeniti odnosno nadodati na to u samome programu kojega izrađuje. Osim gumbova također mogu postojati i drugi elementi koju su već prethodno navedeni. Naravno u grafičkom sučelju programer će napraviti formu koja je kompleksnija nego forma prikazana gore tako da u nastavku slijedi prikaz kompleksnije form.



Slika 8. Forma za program znanstvenog kalkulatora

Na slici 8 prikazana je forma koja se sastoji od nekoliko gumbova, radijskih gumbova i jedne oznake. Oznaka služi kao displej kalkulator, što je omogućeno mijenjanjem teksta oznake pomoću programskog koda. Na slici se također vide i gumbi različitih dimenzija, naziva i s

različitim tekstom na njima što korisniku daje doznanja što koji gumb radi, npr. sqrt gumb je korijen, gumbovi s brojevima služe za unos brojeva u oznaku (displej) clr briše sve iz displeja i tako dalje. Bez tekstualnog sučelja program ne bi bilo moguće izraditi pa je ono i bitnije od grafičkog sučelja. Kao što je već ranije rečeno program je moguće u potpunosti napraviti u tekstualnom sučelju iako će na taj način programski kod biti mnogo zatrpaniji, nepregledniji, kompliciraniji i potrebno je puno više znanja i vremena za izradu programa. Programi izrađeni Samo u tekstualnom sučelju mogu dati puno bolje rezultate, nadalje ponekad je potrebno raditi stvari koje se mogu napraviti u grafičkom sučelju uz pomoć tekstualnog radi funkcionalnosti programa. Na primjer ako je potrebno napraviti gumb na koji kad korisnik klikne se pojave 3 nova gumba od kojih ako se klikne na bilo koji će se pojaviti još 3 nova ista takva gumba. Prvi gumb bilo bi logično napraviti u grafičkom sučelju, no programski kod od funkcije klika na taj gumb morao bi kreirati 3 nova gumba s istom tom funkcijom bez ikakvog drugog mijenjanja forme u grafičkom sučelju. Na slici u nastavku prikazano je tekstualno sučelje programskog jezika Visual Basic zajedno se isječkom koda za znanstveni kalkulator.

```
171         If RAD.Checked Then
172             Broj1 = Val(displej.Text)
173             Rjesenje = Math.Tan(Broj1)
174             displej.Text = Rjesenje
175         ElseIf DEG.Checked Then
176             Broj1 = Val(displej.Text) * pi / 180
177             Rjesenje = Math.Tan(Broj1)
178             displej.Text = Rjesenje
179         End If
180     End Sub
181     Private Sub logaritam_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles logaritam.Click
182         Broj1 = Val(displej.Text)
183         Rjesenje = Math.Log10(Broj1)
184         displej.Text = Rjesenje
185     End Sub
186     Private Sub jednako_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles jednako.Click
187         Broj2 = Val(displej.Text)
188         Select Case Operatori
189             Case "+"
190                 Rjesenje = Broj1 + Broj2
191             Case "-"
192                 Rjesenje = Broj1 - Broj2
193             Case "/"
194                 Rjesenje = Broj1 / Broj2
195             Case "*"
196                 Rjesenje = Broj1 * Broj2
197             Case "^"
198                 Rjesenje = Broj1 ^ Broj2
199             Case "ROT"
200                 Rjesenje = Broj1 ^ (1 / Broj2)
201
202         End Select
```

Slika 9. Tekstualno sučelje programskog jezika Visual Basic

Na slici 9 prikazan je isječak koda zajedno s brojem redaka i odjeljcima za svaku klasu i petlju. Svaka petlja i klasa mogu se zatvoriti radi lakše preglednosti koda. radi preglednosti različite riječi u kodu su označene različitim bojama tako su na primjer deklaracije varijabli i pozivanje funkcija vezanih uz objekte pobojane plavo, nazivi funkcija, žuto, tekst pod navodnim znakovima koji program prepoznaje kao znakovni niz narančasto, određene naredbe i dijelovi koda vezani uz njih ljubičasto itd. Osim toga ukoliko je negdje u programskom kodu greška vezana za sintaksu ili je neka riječ krivo napisana odnosno nije prepoznata od strane programa i ako program prilikom pokretanja odnosno kompajliranja

programskog koda pronađe grešku u kodu tada će se na mjestu gdje je greška pojaviti crvena oznaka koja programeru daje doznanja da nešto nije uredu. Kompajler i debugger su jako bitni dijelovi razvojne okoline. Kompajler je dio programa Visual Studio koji pretvara programerov programski kod koji je zapisan u jeziku Visual Basic, zajedno s grafičkim izgledom i rasporedom forme u strojni jezik odnosno jezik koji računalo razumije. Strojni jezik je sam po sebi binaran odnosno naredbe su nizovi nula i jedinica koje računalo može čitati i na temelju kojih može vršiti računske i logičke operacije te prikazivati njihov rezultat na ekranu u obliku programa odnosno prozora u kojem se on odvija.

4.3 Varijable i konstante u programskom jeziku Visual Basic

U programskom jeziku Visual Basic postoje mnoge vrste varijabli i konstanti. Varijable i konstante su podatci koji su ugrađeni u program i imaju određeni tip. Varijable i konstante funkcioniraju na način da program odredi adresu u koju će biti zapisane prilikom pokretanja programa, tada se ovisno o programu rezerviraju potrebne memorijske lokacije u koje će se spremiti određene vrijednosti. Razlika između varijable i konstante je jedino ta da kada se one deklariraju u programu kao određeni tip konstante moraju imati vrijednost prilikom deklaracije i ona se ne može mijenjati u programu, dok varijable ne moraju imati vrijednost prilikom deklaracije i njihova vrijednost se može mijenjati kroz rad programa. Tako na primjer može postojati konstanta π koja će imati vrijednost koju ima π u matematici i bilo gdje gdje bi se u programu trebao koristiti π koristila bi se ta varijabla radi lakšeg pisanja programa ili ukoliko bi korisniku bilo dopušteno da koristi π kao unos vrijednosti umjesto da sam upisuje vrijednost broja π. U tom slučaju programer bi sam deklarirao konstantu i u deklaraciji bi toj konstanti pridružio vrijednost koju ta konstanta predstavlja npr. za π ta vrijednost bi bila 3.14159265358979. Što se varijabli tiče one moraju biti deklarirane u programskom kodu isto kao što moraju biti deklarirane i konstante no kao što je već prethodno rečeno ta varijabla ne mora, ali može imati pridruženu vrijednost prilikom deklaracije. Ta vrijednost se za razliku od konstante može mijenjati ovisno o programskom kodu i unosima korisnika, ili korisnik čak može unositi svoje vrijednosti u neke varijable ako je to dopušteno programskim kodom odnosno ako je ta opcija potrebna. U programskom jeziku Vizual Basicu postoje mnogi tipovi varijabli i konstanti koji mogu biti određeni. Svaki tip može imati zapisane različite podatke u svojoj memorijskoj lokaciji i zauzima različit prostor u memoriji. Iako su sve vrijednosti zapisane kao nizovi nula i jedinica ovisno o tome koji je tip određen za varijablu računalo će tretirati te nizove na različite načine. U tablici koja slijedi u nastavku prikazani su svi tipovi varijabli i konstanti koji postoje u Visual Basic-u.

Tip podatka u Visual Basic-u	Mjesto u memoriji	Opis i moguće vrijednosti
Boolean	Ovisi o platformi	Logička vrijednost korištena za logičke operacije i petlje, vrijednosti true ili false (istina (1) ili laž (0))
Byte	1 bajt	Vrijednosti od 0 do 255 bez predznaka
Char	2 bajta	Vrijednosti od 0 do 65535 bez predznaka, ukoliko program ispisuje varijablu ili konstantu tipa char ispisat će se znak koji odgovara vrijednosti varijable char
Date	8 bajta	Datumi i vrijeme od ponoći prvog siječnja 0-te godine do 11:59:59, 31. Prosinca 9999. godine
Decimal	16 bajta	Vrijednosti od 0 do +/-7.9x10^28 s mogućnošću postavljanja decimalne točke
Double	8 bajta	Sadrži vrijednosti IEEE 64 bitnog zapisa broja
Integer	4 bajta	Sadrži brojeve od -2'147'483'648 do 2'147'483'647
Long	8 bajta	Sadrži brojeve od -9,223'372'036'854'775'808 do 9'223'372'036'854'775'807
Object	4 ili 8 bajta ovisno o platformi	Objekt može sadržavati bilo koji tip podataka i više varijabli različitih tipova
SByte	1 bajt	Vrijednosti od -128 do 127
Short	2 bajta	Vrijednosti od -32'768 od 32'767
Single	4 bajta	Sadrži vrijednosti IEEE 32 bitnog zapisa broja
String	Ovisi o platformi	Od 0 do 2 milijarde unikodnih znakova (char)
UInteger	4 bajta	Vrijednosti od 0 do 4'294'967'295 bez predznaka
ULong	8 bajta	Vrijednosti od 0 do 18'446'744'073'709'551'615 bez predznaka
User defined (struktura)	Ovisno o platformi	Sadrži različite varijable različitih tipova podataka
UShort	2 bajta	Vrijednosti od 0 do 65'535 bez predznaka

Tablica 1 Tipovi podataka u Visual Basicu

U tablici prikazane su sve vrste podataka odnosno varijabli i konstanti koje mogu postojati. Sve vrste podataka su definirane u programu osim objekta (Object) i strukture (User defined) koje programer sam kreira u programskom kodu ili koje već postoje u samom programskom jeziku Visual Basic. Razlika između njih i ostalih varijabli je u tome što se objekti i strukture kreiraju u program kao posebne nove varijable koje mogu postojati. Prema tome programer može kreirati strukturu auto i u tu strukturu staviti varijable koje su potrebne što se tiče informacija o autu npr. programer može kreirati strukturu tako da u njoj postoje varijable za broj tablice, vrstu goriva koje auto koristi, dali je prednji, zadnji ili pogon na sva 4 kotača itd. što omogućuje bolju organizaciju podataka koji se koriste u programu. Osim struktura postoje i objekti koji su slični u tome da imaju više različitih varijabli spremljenih u pojedinu varijablu ali se bitno razlikuju od struktura iz više razloga. Najbitnija razlika između strukture i objekta je ta što objekt ima javne (public) i privatne (private) podatke kojim se može ili ne može pristupiti ovisno o tome na koji način su isprogramirani, osim toga za objekte se mogu isprogramirati različite funkcije koje su vezane uz taj objekt. U Visual Basic-u variable se deklariraju s naredbom „Dim“. Pri deklaraciji varijabli potrebno je specificirati ime varijable i tip podatka koji je spremljen u nju, a moguće je odmah odrediti vrijednost koja će biti spremljena u varijablu. Primjer naredbe za deklaraciju varijable:

Dim varijabla as Integer – ovom naredbom deklarira se varijabla koja se zove „varijabla“ i koja je tipa integer. Ukoliko programer želi odmah spremiti neku vrijednost u varijablu tada će naredba izgledati ovako:

Dim varijabla as Integer = 100

Nakon izvršavanja te naredbe varijabla koja ima naziv „varijabla“ biti će tipa integer i u nju će automatski biti unesena vrijednost 100. Za pristup vrijednosti spremljenoj u varijabli u programu potrebno je staviti njezino ime u tom slučaju ako bi ova varijabla bila korištena za neku naredbu bilo bi potrebno staviti njezino ime u tu naredbu. Dakle za deklaraciju varijable općenito naredba glasi:

Dim [ime varijable] as [tip podatka] i na kraju može stajati „= vrijednost“ ukoliko je potrebno. Kod deklariranja konstanti funkcionira na sličan način, ali se umjesto „Dim“ koristi se „const“, također kod deklaracije konstante potrebno je konstanti dodijeliti vrijednost prilikom deklaracije zato što se vrijednosti konstanti ne mogu mijenjati u programu. Ovisno o postavkama programa ponekad za konstante nije potrebno definirati koji tip podatka je sama konstanta koja se deklarira već samo vrijednost koja se sprema u konstantu. Tako da naredbe za deklaraciju konstante mogu biti:

Const pi = 3.14159265358979 ili Const pi as double = 3.14159265358979.

Oba slučaja deklarirat će istu stvar, a to je konstanta imena pi koja sadrži vrijednost 3.14159265358979. Varijable se koriste u programskom jeziku Visual Basic kao promjenjivi spremnici podataka i mogu se iz njih podatci čitati ili spremati u njih ovisno o naredbama u programu. One su potrebne kako bi se moglo manipulirati vrijednostima i pamtiti ih te kako bi se mogle koristiti vrijednosti određene od strane programera za druge operacije nevezane uz same varijable. Kao i varijable konstante se koriste na isti način osim što su konstante konstantne vrijednosti odnosno nisu promjenjive i ne može se s njima manipulirati no može ih se koristiti za operacije kao i varijable. Konstante se koriste uglavnom radi jednostavnosti kako programer ne bi morao stalno pisati neku vrijednost koja se često ponavlja i osim toga nema nekog određenog razloga zašto bi se koristile konstante umjesto varijabli. Čak i inače za česte vrijednosti mogu se koristiti varijable ukoliko programer ne stavi neku naredbu koja bi promijenila vrijednost te varijable tada ne bi bilo mogućnosti da se ona promjeni što je zapravo jedina razlika između varijable i konstante. Konstante se ipak koriste u programiranju kako bi se programer lakše snašao u programu i kako bi neki drugi programer znao da programer koji je napravio program želi da ta vrijednost ostane nepromijenjena iz nekog razloga ili kako bi sam sebe podsjetio na to. Varijable i konstante se oboje mogu koristiti za aritmetičke i logičke operacije.

4.5 Operacije u Visual Basic-u

U Visual Basic-u postoje brojne aritmetičke i logičke operacije koje se mogu izvršavati s različitim operatorima koristeći različite varijable, konstante ili jednostavno vrijednosti koje programer unese. Tako je moguće izvršiti neku operaciju bez deklaracije varijable ili konstante za neku određenu vrijednost i to se tako izvršava u slučajevima kada se neka vrijednost ponavlja jako malo puta i kada nema smisla kreirati varijablu ili konstantu za to. Aritmetičke operacije su operacije kao što su zbrajanje, oduzimanje, množenje i dijeljenje dok su logičke operacije one koje koriste Boole-ovu algebru i koriste logičke vrijednosti 0 ili 1 odnosno istina ili laž i kod takvih operacija pretežito se koriste varijable tipa Boolean koje mogu imati samo vrijednost „true“ ili „false“ odnosno samo 0 ili 1 odnosno imaju samo logičke vrijednosti. Druge varijable mogu se koristiti za logičke operacije na način da ukoliko varijabla sadrži neku vrijednost koja je različita od 0 tada ju program čita kao 1 odnosno kao istinu, a ukoliko sadrži vrijednost 0 tada ju program čita kao 0 odnosno laž u logičkim operacijama. U tablici koja slijedi u nastavku prikazani su aritmetički operatori koji se koriste u programskom jeziku Visual Basic

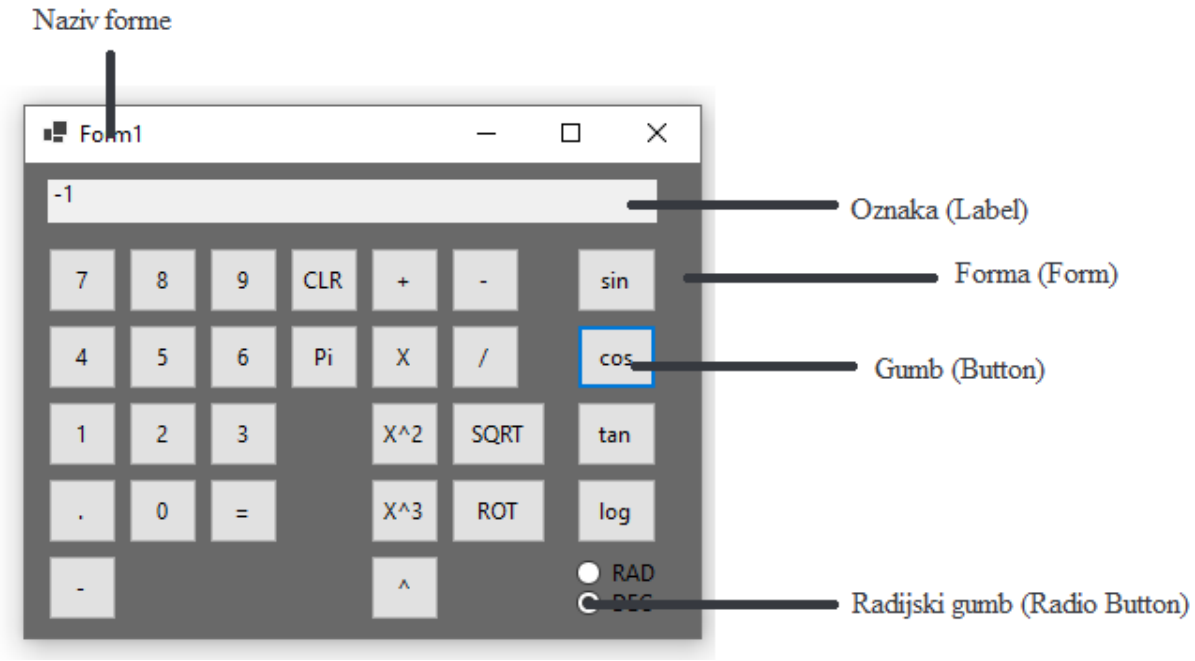
Operator	Opis operacije
+	Operator zbrajanja, zbraja dva broja
-	Operator oduzimanja, vraća razliku dva broja
*	Operator množenja, množi prvi broj, drugim brojem
/	Operator dijeljenja, dijeli prvi broj drugim brojem i vraća Float vrijednost (s decimalnom točkom)
\	Operator dijeljenja, dijeli prvi broj drugim brojem i vraća integer vrijednost (bez decimalne točke)
MOD	Operator modulacije, vraća ostatak cjelobrojnog dijeljenja
^	Operator potenciranja, vraća vrijednost prvog broja na potenciju određenu drugim brojem (također se koristi i za korjenovanje)

Tablica 2. Aritmetički operatori u programskom jeziku Visual Basic

U tablici 2 prikazani su svi aritmetički operatori koji postoje u programskom jeziku Visual Basic. Unatoč tome postoje još logički operatori, operatori usporedbe i slično. Iako su ovo jedini operatori koje koristi programski jezik Visual Basic moguće je kreirati funkcije koje mogu služiti kao druge operacije npr sinus ili kosinus nekog broja ili pretvaranje mjernih jedinica ili neka druga operacija koja se može ostvariti pomoću jedne ili više od gore navedenih aritmetičkih operacija. Osim što programer može programirati funkcije kako bi ostvario kompleksnije aritmetičke operacije od onih koje su moguće u samom programskom jeziku programer također može uključiti i razne klase koje sadrže prethodno isprogramirane funkcije koje programer može pozivati kao da ih je on sam napravio. Klase mogu biti uključene u program s kojim programer radi ili se mogu skinuti s interneta ili kreirati na vlastitom računalu ukoliko programer želi koristiti neke funkcije koje je sam isprogramirao ili koje je neko drugi napravio a nisu dostupne u sklopu programa koji koristi za programiranje. Jedna klasa koja postoji u sklopu Visual Basica je system.math koja dodaje razne dodatne matematičke funkcije koje nisu dostupne u samom Visual Basic-u, a vrlo su česte u uporabi u matematici i znanosti. Neke od funkcija koje dodaje system.math su; sinus, kosinus, tangens i drugi korijen.

4.6 Znanstveni kalkulator izrađen u programskom jeziku Visual Basic

Program znanstvenog kalkulatora koji sam ja izradio poprilično je jednostavan i moguća su mnoga unaprjeđenja za njega. U nastavku slijedi slika prozora samog programa uz objašnjenja što koji element predstavlja.



Slika 10. Prikaz forme za znanstveni kalkulator s prikazanim elementima

Na slici 10 prikazana je forma programa zajedno s ostalim elementima koji se nalaze na njoj i koji omogućuju programu da radi. Sama forma je mjesto gdje se slažu elementi kao što su gumbi, radijski gumbi, oznake, prozori za slike, prozori za unos i slično, a naziv forme određuje programer prilikom izrade programa. Osim forme vidljivi su gumbi kojih ima najviše. Njihova funkcija je takva da prilikom klika na njih oni izvršavaju neki dio programa koji je određen programskim kodom. Tako gumbi za brojeve (0-9) služe za dodavanje broja na kraj teksta koji je napisan u oznaci zapisuju broj koji se nalazi na gumbu, gumb „Pi“ na ekran ispisuje vrijednost broja pi koja je definirana s konstantom na početku samog programa, gumbi s točkicom označava decimalnu točku odnosno kreira decimalan broj, a gumb s minusom ispod brojeva služi za mijenjanje predznaka broja napisanog na ekranu. Gumbi s desne strane služe za izvršavanje aritmetičkih operacija. Operacije koje zahtijevaju 2 operanda (zbrajanje, oduzimanje, množenje dijeljenje, potenciranje na unesenu potenciju, korjenovanje) napravljeni su na način da nakon unosa prvog operanda i klika na gumb oni zapisuju tekst koji je prikazan u oznaci u varijablu u obliku brojčane vrijednosti onoga što je napisano. Potom obrišu tekst iz oznake kako bi se mogao upisati novi tekst u nju, a zatim prilikom klika na tipku „=“ izvršava se operacija koja je određena s prethodno stisnutim gumbom koristeći operand iz varijable i tekst sa ekrana pretočenu u brojčanu vrijednost. To je omogućeno pomoću select case naredbe. Select case naredba omogućuje da se izvršava ona operacija koja je odabrana prilikom klika na „=“ tako da program odradi operaciju koja je odabrana od strane korisnika. Operacije koje zahtijevaju samo jedan operand (kvadrat, kub,

drugi korijen, sinus, kosinus, tangens, logaritam na bazi 10) izvedene su na način da se koristi samo vrijednost s oznake pretvorena u brojčanu vrijednost i za njih nije potrebno koristiti select case naredbu nego se operacija izvršava prilikom klika na gumb. Oznaka se koristi kao mjesto za zapisivanje teksta no u ovom slučaju ona se koristi za dohvaćanje brojčane vrijednosti iz teksta pomoću naredbe `Val(displej.Text)` s kojom se tekstualna vrijednost koja je zapisana u oznaci „displej“ pretvara u brojčanu. Ovo je potrebno napraviti zato što program čita sve znakove u oznaci kao unikodne znakove, a ne kao brojčane vrijednosti radi mogućnosti unosa unikodnih znakova tako da je prije korištenja unesenih brojeva potrebno iste pretvoriti u brojčanu vrijednost. Ovo je slučaj radi načina na koji je objekt oznake definiran u Visual Basic-u odnosno gdje je tekst zapisan u oznaku realiziran kao podatak tipa string. Osim njih ostaju još radijski gumbi koji se koriste za trigonometrijske funkcije (sinus, kosinus i tangens) kao odabir za korištenje radijana ili stupnjeva za vrijednosti kuteva. Radijski gumbi rade na način da kada se stisne na jedan od njih on se označi s točkicom, a ukoliko je neki radijski gumb u istoj skupini prethodno bio označen tada se s njega oznaka briše što omogućuje odabir više različitih opcija, ali samo jedna od njih može biti odabrana u bilo kojem trenutku. Ukoliko bi programer htio dopustiti korisniku da odabere više opcija može postaviti potvrdni okvir (Check box) koji funkcionira isto kao i radijski gumb uz razliku da ih se može označiti ili maknuti oznaku s njih bez obzira na druge potvrdne okvire, a moguće je i napraviti potvrdne okvire koji kada su označeni mogu maknuti oznake s drugih potvrdnih okvira ili označiti druge potvrdne okvire. Kada je gumb za neku od trigonometrijskih funkcija pritisnut s If naredbom provjerava se koji radijski gumb je označen (odnosno dali je uopće ijedan od njih označen) i ovisno o tome pokreće se dio programskog koda koji odgovara označenom radijskom gumbu. Jedina razlika je ta što se prije poziva funkcije vrijednost koja je odabrana pretvara u vrijednost koju će koristiti program za izvršavanje funkcije. Ako je odabran radijski gumb „RAD“ koji označava da korisnik želi da vrijednost bude u radijanima, program neće ništa mijenjati jer se u programu vrijednosti automatski čitaju u radijanima no ako je odabran radijski gumb „DEG“ program će pretvoriti vrijednost koja je unesena iz stupnjeva koje je korisnik unesao u radijane koje će program onda koristiti za izvršavanje funkcije. Funkcije kalkulatora za zbrajanje, oduzimanje, množenje, dijeljenje, potenciranje i korjenovanje su dio samog programskog jezika Visual Basic no dodatne funkcije za logaritam po bazi 10, sinus, kosinus i tangens su dodatne funkcije iz klase System.Math koja se mora posebno uključivati u program kako bi se mogle koristiti funkcije iz nje. Kao što je vidljivo na slici gore moj program nema veliki izbor matematičkih operacija koje se mogu koristiti, nema mogućnost unosa neke duže jednadžbe

ili unosa razlomaka i nema konstantu e. Kako bi se ovaj program poboljšao postoje mnoge stvari koje bi se mogle napraviti, prva od kojih je dodavanje više aritmetičkih operacija koje se mogu izvršiti samim kalkulatorom. Neke od operacija koje bi se mogle dodati su:

- Računanje logaritma po bilo kojoj bazi – bilo bi potrebno kreirati funkciju ili ju pronaći u sklopu neke klase
- Pretvaranje stupnjeva u radijane i obratno
- Računanje s drugim brojevnim sustavima
- Mogućnost unosa razlomaka
- Mogućnost unosa dužih matematičkih izraza
- Mogućnost unosa i rješavanja jednadžbi
- Mogućnost rješavanja derivacija i integrala
- Mogućnost proračunavanja i grafičkog prikaza grafova
- Ostale mogućnosti koje su navedene u prvom poglavlju ovoga rada

Pamćenje prethodnih unosa jedna je od funkcija koje ne bi bilo teško izvesti i moglo bi ići do nekog određenog broja unosa recimo program bi pamtio 10 operacija koje se rješavaju (prvi operand, drugi operand i operacija koja se izvršila) pomoću jedne strukture i niza od 10 članova u toj strukturi gdje bi se spremale operacije onako kako se izvode i kada bi se izvršila jedanaesta operacija spremila bi se na prvo mjesto a ostalih 9 bi ostale gdje jesu i nakon toga bi se nastavile operacije spremati na svako sljedeće mjesto u nizu.

4.7 Programski kod

```
Imports System.Math
Public Class Form1
    Dim Broj1, Broj2, Rjesenje As Double
    Dim Operatori As String = "0"
    Dim obrisi As Boolean
    Const pi = 3.14159265358979

    Private Sub t0_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t0.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "0"
    End Sub
    Private Sub t1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t1.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "1"
    End Sub
```

```
    Private Sub t2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t2.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "2"
    End Sub
    Private Sub t3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t3.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "3"
    End Sub
    Private Sub t4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t4.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "4"
    End Sub
    Private Sub t5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t5.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "5"
    End Sub
    Private Sub t6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t6.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "6"
    End Sub
    Private Sub t7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t7.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "7"
    End Sub
    Private Sub t8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t8.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "8"
    End Sub
    Private Sub t9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles t9.Click
        If obrisi Then
            displej.Text = " "
        End If
        obrisi = False
        displej.Text = displej.Text + "9"
```

```
    End Sub
    Private Sub ponisti_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ponisti.Click
        displej.Text = " "
    End Sub
    Private Sub plus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles plus.Click
        Broj1 = Val(displej.Text)
        Operatori = "+"
        obrisi = True
    End Sub
    Private Sub minus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles minus.Click
        Broj1 = Val(displej.Text)
        Operatori = "-"
        obrisi = True
    End Sub
    Private Sub djeli_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles djeli.Click
        Broj1 = Val(displej.Text)
        Operatori = "/"
        obrisi = True
    End Sub
    Private Sub mnozi_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnozi.Click
        Broj1 = Val(displej.Text)
        Operatori = "*"
        obrisi = True
    End Sub

    Private Sub kvadrat_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles kvadrat.Click
        Broj1 = Val(displej.Text)
        Rjesenje = Broj1 ^ 2
        displej.Text = Rjesenje
    End Sub
    Private Sub potencija_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles potencija.Click
        Broj1 = Val(displej.Text)
        Operatori = "^"
        obrisi = True
    End Sub
    Private Sub kub_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles kub.Click
        Broj1 = Val(displej.Text)
        Rjesenje = Broj1 ^ 3
        displej.Text = Rjesenje
    End Sub
    Private Sub drugikorjen_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles drugikorjen.Click
        Broj1 = Val(displej.Text)
        Rjesenje = Broj1 ^ 0.5
        displej.Text = Rjesenje
    End Sub
    Private Sub tocka_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles tocka.Click
        If displej.Text.IndexOf(".") > 0 Then
            Exit Sub
        Else
            displej.Text = displej.Text + sender.text
        End If
    End Sub
```

```
    Private Sub sinus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles sinus.Click
        If RAD.Checked Then
            Broj1 = Val(displej.Text)
            Rjesenje = Math.Sin(Broj1)
            displej.Text = Rjesenje
        ElseIf DEG.Checked Then
            Broj1 = Val(displej.Text) * pi / 180
            Rjesenje = Math.Sin(Broj1)
            displej.Text = Rjesenje
        End If
    End Sub
    Private Sub kosinus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles kosinus.Click
        If RAD.Checked Then
            Broj1 = Val(displej.Text)
            Rjesenje = Math.Cos(Broj1)
            displej.Text = Rjesenje
        ElseIf DEG.Checked Then
            Broj1 = Val(displej.Text) * pi / 180
            Rjesenje = Math.Cos(Broj1)
            displej.Text = Rjesenje
        End If
    End Sub

    Private Sub korjen_Click(sender As Object, e As EventArgs) Handles
korjen.Click
        Broj1 = Val(displej.Text)
        Operatori = "ROT"
        obrisi = True
    End Sub

    Private Sub PI_Click(sender As Object, e As EventArgs) Handles PIgumb.Click
        displej.Text = pi
    End Sub

    Private Sub Predznak_Click(sender As Object, e As EventArgs) Handles
Predznak.Click
        If displej.Text.IndexOf("-") = 0 Then
            displej.Text = Val(displej.Text) * -1
        Else
            displej.Text = sender.text + displej.Text
        End If
    End Sub

    Private Sub tangens_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles tangens.Click
        If RAD.Checked Then
            Broj1 = Val(displej.Text)
            Rjesenje = Math.Tan(Broj1)
            displej.Text = Rjesenje
        ElseIf DEG.Checked Then
            Broj1 = Val(displej.Text) * pi / 180
            Rjesenje = Math.Tan(Broj1)
            displej.Text = Rjesenje
        End If
    End Sub
    Private Sub logaritam_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles logaritam.Click
        Broj1 = Val(displej.Text)
        Rjesenje = Math.Log10(Broj1)
        displej.Text = Rjesenje
    End Sub
```

```
    Private Sub jednako_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles jednako.Click
        Broj2 = Val(displej.Text)
        Select Case Operatori
            Case "+"
                Rjesenje = Broj1 + Broj2
            Case "-"
                Rjesenje = Broj1 - Broj2
            Case "/"
                Rjesenje = Broj1 / Broj2
            Case "*"
                Rjesenje = Broj1 * Broj2
            Case "^"
                Rjesenje = Broj1 ^ Broj2
            Case "ROT"
                Rjesenje = Broj1 ^ (1 / Broj2)

        End Select
        displej.Text = Rjesenje
        obrisi = True
    End Sub
End Class
```

5. ZAKLJUČAK

Znanstveni kalkulatori su dosta kompleksni uređaji pomoću kojih se olakšavaju izračuni u matematici, inžinjerstvu i drugim poljima u kojima je potrebno računanje te ovisno o namjeni mogu biti jednostavniji ili kompleksniji. Rađenje i učenje u programskom jeziku Visual Basicu su vrlo jednostavni za početnike i programiranje upravljano događajima iako kompleksno je jako korisno ovisno o tome što je poželjno za program da ima u smislu funkcija i mogućnosti.

LITERATURA

[1] Wikipedia: Znanstveni kalkulator
https://en.wikipedia.org/wiki/Scientific_calculator

[2] Wikipedia: Grafički kalkulator
https://en.wikipedia.org/wiki/Graphing_calculator

[3] Free code camp, Germán Cocca: Programming Paradigms – Paradigm Examples for Beginners
https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/

[4] Wikipedia: Visual basic (clasic)
https://en.wikipedia.org/wiki/Visual_Basic_(classic)

[5] Wikipedia: Basic
https://en.wikipedia.org/wiki/BASIC

[6] Microsoft: Sažetak tipova podataka (Visual Basic)
https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/data-types/

[7] Microsoft: User defined constants (Visual Basic)
https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/constants-enums/user-defined-constants

[8] Tutorials point: VB.net operatori
https://www.tutorialspoint.com/vb.net/vb.net_operators.htm