Uvod u programiranje kroz jezik Visual Basic

Saša Fajković

2015

Sadržaj :

1) Što je to programiranje?

Opće je poznato kako je programerski posao zapravo posao snova kojeg karakterizira velika plaća, rad od kuće, fantastični radni uvjeti i da su takvi ljudi izuzetno inteligentni. Za takav posao ljudi se školuju desetljećima a oni istinski programeri nemaju socijalni život kao niti razvijene socijalne vještine, ali zato imaju pogrbljena leđa, lice puno akni, masnu kosu i nemaju djevojku, ženu ili zaručnicu. Od svih navedenih pretpostavki niti jedna nije točna osim (na apstraktnoj razini) možda one o školovanju. Sagledamo li pojam školovanja kao edukaciju na nekoj visokoobrazovnoj ustanovi onda je teško naći nekoga tko je proveo desetljeće studirajući isti studij, a da istovremeno spada u vrsne programere. Sagledamo li ipak pojam učenja malo šire, onda se zasigurno može reći kako se programeri educiraju desetljećima, no to znači suočavanje s novim problemima, tehnologijama i načinima poslovanja. Slična je situacija i u svakom drugom zanimanju. Promjeni li se računovodstvena aplikacija u nekom poduzeću koje se bavi vođenjem knjiga za druga poduzeća, zaposlenici će se morati naučiti koristit novim softverom (*engl. software*) što je također učenje odnosno edukacija. Ipak, za programere možemo reći kako je količina i učestalost usvajanja novih znanja češća jer se radi o poslu koji nerijetko zahtijeva jedinstvena rješenja (iako možda slična, ali rijetko kada potpuno identična) a time je i svaki takav zadatak novo učenje.

Važno je napomenuti kako programiranje može biti izuzetno zabavan posao no način usvajanja gradiva je „*bottom-up*„ proces i čini se kao da ništa nema smisla i da nikada nećete stići do cilja da postanete programer. Za buduće programere to znači da će se trebati prvo naučiti određena količina znanja i tek na kraju će se sva ta znanja spojiti u veliku cjelinu i sve će imati smisla. Programiranje ipak, suprotno vjerovanju mnogih, **NIJE** bazirano na jednostavnom grafičkom sučelju. Povlačenje gotovih trodimenzionalnih modela koji nalikuju na ljude, civilne aute, cestu, zgrade i policijske aute nije način kako nastaju igre. Jako puno „dosadnih“ linija teksta i još teksta i još teksta leži iza svega onoga što krajnjim korisnicima omogućuje da pritiskom na tipku „A“ ili „lijevu strelicu“ auto zaista skrene u lijevo i spektakularnim manevrom izbjegnete policijski auto koji Vas je pratio posljednjih dvadeset minuta.

Summa summarum, programiranje je jako puno pisanja kôda (*engl. code*) i smišljanja algoritama, ispijanja kave u velikim količina (stereotip koji čak i je često istinit), traženja i velikih, ali i jako malih grešaka te frustriranja svaki put kada klijent zatraži promjenu u programu jer nije znao opisati što točno želi i sada misli kako je dodavanje njegove „male želje“ još samo par minuta posla. Programiranje nije „*drag-n-drop*“ sistem povlačenja grafičkih modela i pritisak tipke „Start“ nakon čega sve nekako magično proradi samo od sebe.

Programiranje se ne uči *štrebanjem*. S*int*aksu jezika je potrebno naučiti napamet i prihvatiti je „zdravo za gotovo“, ali dobar programer se postaje iskustvom. Što više kôda napišete postajete sve bolji programer.

Zapamtite, računalo radi ono što mu kažemo, ne ono što mi želimo. Jako često u programerskom svijetu, te su dvije stvari potpuno različite.

2) Razvojna sučelja i programski jezici

Razvijanje aplikacija (drugi često korišteni nazivi su programiranje, pisanje kôda, kôdiranje i sl.) moguće je čak i u najjednostavnijoj aplikaciji za uređivanje teksta. Primjer je popularni Notepad[1] koji dolazi pred instaliran u sklopu Windows operativnog sustava. Naravno, takva aplikacija nije namijenjena razvijanju aplikacija već pisanju jednostavnog teksta i takav pristup kodiranju nema smisla. Uz danas već zaista veliku paletu aplikacija namijenjenih kodiranju često možete čuti kako se koristi Notepad++, Code::Blocks, Eclipse, Microsoft Visual Studio, NetBeans i slične aplikacije. Njih zovemo **razvojna sučelja** odnosno **integrirana razvojna sučlelja** (*engl. integrated development environment*)**.** U sklopu ovakvih specijaliziranih aplikacija nalazimo veliki broj gotovih rješenja koje nam omogućuju više fokusiranja na programiranje i smišljanje algoritama a smanjuju količinu vremena potrebnu za pisanje ponavljajućeg kôda. Naravno, nisu sve aplikacije specijalizirane za sve programske jezike kojih također postoji veliki broj a svaki od njih donosi neke prednosti ali i mane naspram drugih. Uz programski jezik C++, iznimnu popularnost dijele i Java (izgovara se ili java ili đava), C (izgovara se „ce“ ili često zvan „čisti C“), C# (izgovara se „ce šarp“ ili „si šarp“), Python (izgovara se „pajton“) te mnogi drugi. Važno je napomenuti kako **HTML, CSS, JavaScript** i sl. ne spadaju u programske jezike već u opisne odnosno skriptne jezike.

a. Microsoft Visual Studio

Kôdiranje u bilo kojem razvojnom sučelju je jako slično. Sve važne stvari koje se tiču samog jezika su uvijek iste, neovisno o razvojnom okruženju. Razlika se očituje u dizajnu sučelja, dodatnim opcijama, pozicioniranjem dijelova aplikacije na ekranu no isti kôd će svuda izvršavati identičan proces. Kroz ove primjere koristiti će **Microsoft Visual Studio 2013** [2]razvojno sučelje s podrškom za programski jezik Visual Basic.

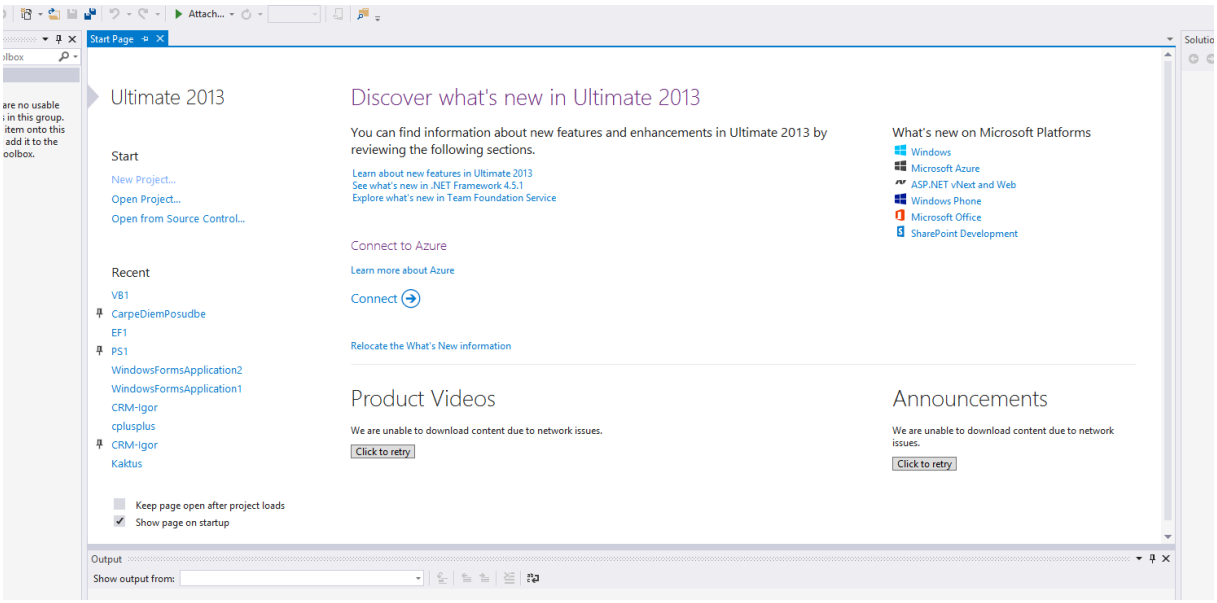
Razvojno sučelje preuzmite sa službenih Microsoft stranica. Besplatna verzija je i više nego dovoljna za naše potrebe.

[1] Notepad je program za osnovno uređivanje i stvaranje teksta a dolazi u sklopu Microsoft Windows operativnog sustava

[2] http://www.visualstudio.com/

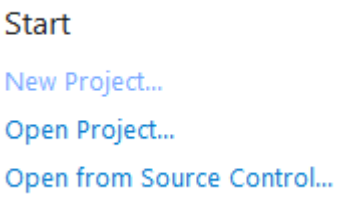
b. MS Visual Studio – pokretanje i kreiranje prvog projekta

Nakon što instalirate Microsoft Visual Studio razvojno sučelje, pokrenite ga. Dočekat će vas početni ekran sličan ovome:



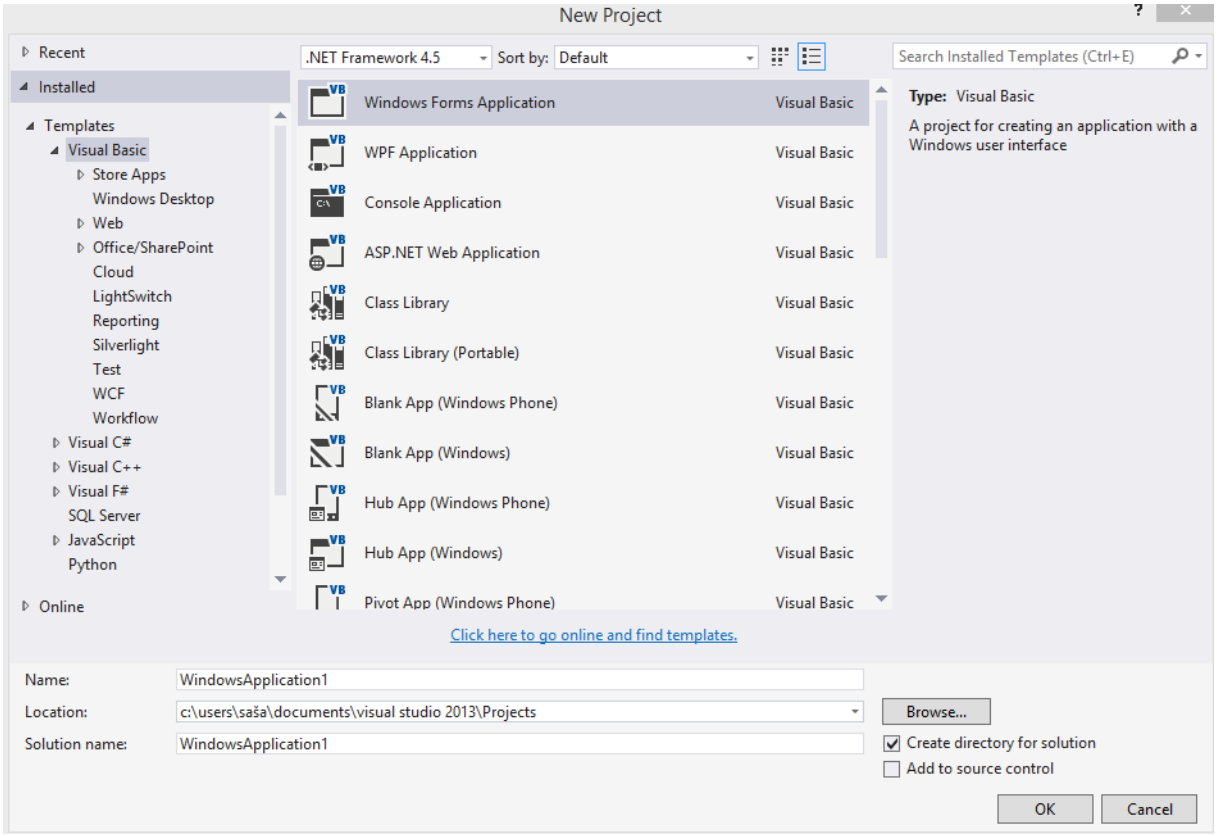
Slika 1 - Microsoft Visual Studio početna stranica

Ovo je početna stranica MS Visual Studio razvojnog sučelja koja nudi prikaz posljednje korištenih projekata, mogućnost kreiranja novih, prikaz eventualnih nadogradnji, itd. Kako bi kreirali novi projekt potrebno je odabrati opciju *New Project* koja se nalazi u gornjem lijevom uglu početne stranice pod kategorijom Start odnosno File / New / Project.



Slika 2 - Start izbornik

Otvorit će se novi izbornik za kreiranje novog projekta. Imajte na umu da Visual Studio u ovom trenutku ne zna s kojim jezikom planirate raditi.



Slika 3 - Izrada novog projekta

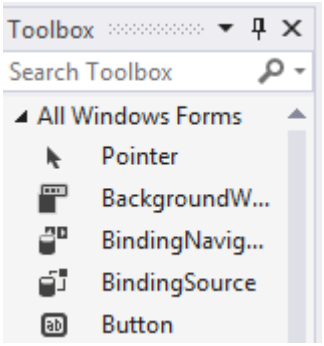
Izbornik je podijeljen tako da s lijeve strane odabiremo jezik s kojim želimo raditi. U srednjem dijelu se nalazi popis mogućnosti odnosno vrste projekata za odabrani jezik. S desne strane se prikazuje kratak opis odabrane vrste projekta.

U donjem dijelu nalazimo tri tekstualna polja. U prvo (*Name*) upisujemo ime projekta. U drugo tekstualno polje (*Location*) određujemo lokaciju gdje će biti spremljen *Solution* koji radimo. Treća opcija (*Solution*) određuje naziv „Rješenja“. Ideja je da jedan *Solution* može sadržavati više projekata kako bi se postignulo konačno rješenje problema (aplikacija koju želimo izraditi).

Pripazite da je s lijeve strane odabran dobar jezik (Visual Basic) i da je pod kategorijom Visual Basica odabrana izrada Windows Forms Application.

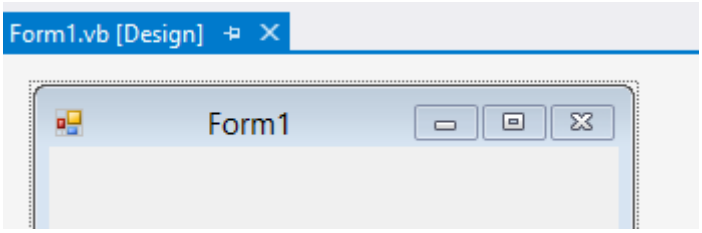
Nakon što upišemo potrebne vrijednosti, pritisnemo OK i kreirat će se naš prvi projekt. Primijetite kako je raspoređeno razvojno sučelje. S lijeve strane se vrlo vjerojatno nalazi popis kontrola. Kontrole su primjerice *Button, TextBox, Label*, itd. Kontrole se nalaze pod kategorijom *Toolbox*.

Ako **Toolbox** (ili neka druga kategorija nije vidljiva) možete ih upaliti preko **VIEW** izbornika. Kratica za upaliti ili ugasiti prikaz Toolboxa je Ctrl+Alt+X.



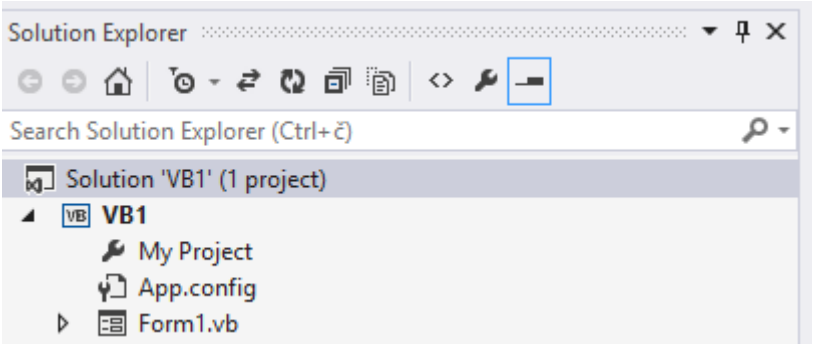
Slika 4 – Toolbox

U srednjem dijelu se nalazi **Windows Forma**. Nju zamišljajte kao prozor vaše buduće aplikacije. U formu ćemo postavljati kontrole koje će nešto raditi.



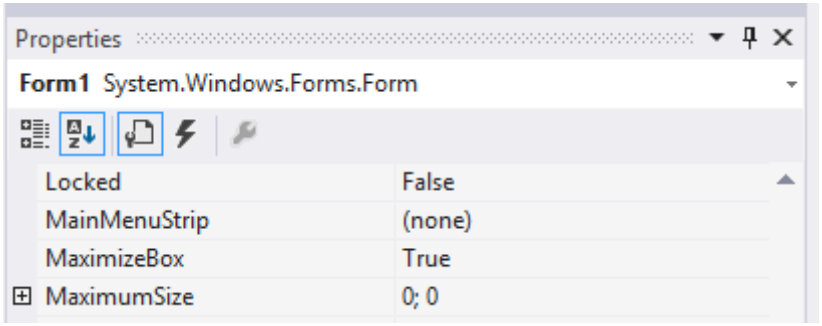
Slika 5 - Windows forma

S desne strane se vjerojatno nalazi **Solution Explorer** koji prikazuje naš projekt i datoteke u njemu. Naravno, da imamo dva projekta u ovom Solutionu, prikazala bi se oba projekta.



Slika 6 - Solution Explorer

Ispod Solution Explorera se nalazi prozor s popisom svojstava odnosno događanja dostupnih za neku kontrolu. Svojstvo kontrole je primjerice pozadinska boja, boja teksta koji je upisan u kontrolu, dimenzije kontrole, lokacija kontrole, itd. Događaji su ono što se događa na neku određenu akciju poput pritiska na Button, duplog pritiska i slično.



Slika 7 – Properties

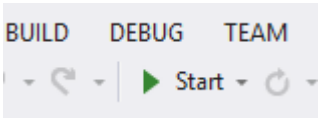
3) Pokretanje projekta

Projekt pokrećemo pritiskom na tipku F5, kombinacijom Ctrl+F5 ili korištenjem za to predviđene tipke.

F5 – pokrećemo aplikaciju/projekt s uključenim debuggerom

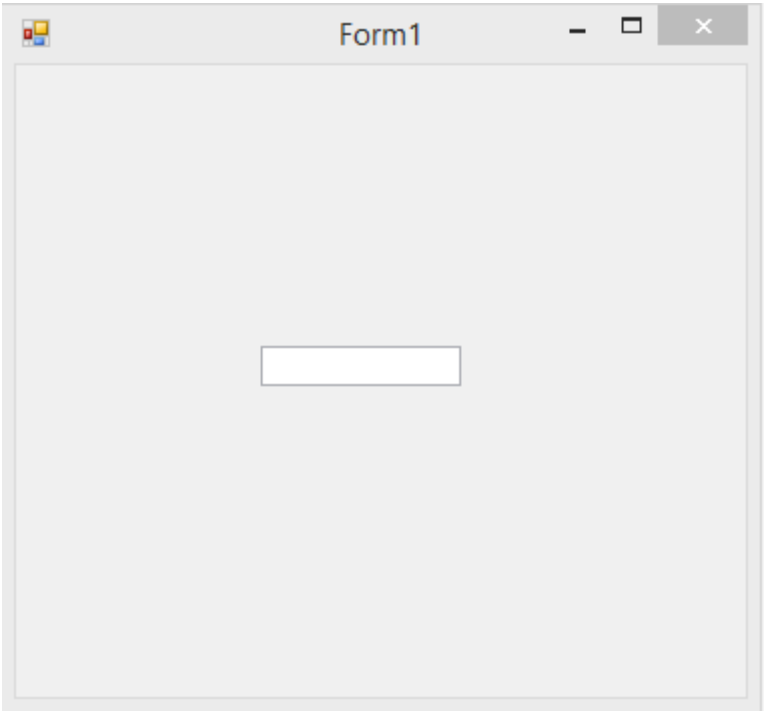
Ctrl+F5 – pokrećemo aplikaciju/projekt s isključeniim debuggerom

Start button – isto kao i F5



Slika 8 – Start – pokretanje projekta

Nakon toga će se otvoriti naša aplikacija u jednom od dva moguća načina rada (s ili bez debuggera). Preporuka je da projekte pokrećete s uključenim debuggerom koji će pomoći u bržem i lakšem otklanjanju pogrešaka.



Slika 9 - Pokretanje projekta

Primijetite kako je aplikacija pokrenuta i sada stoji u Taskbaru[3]. Naravno, prikaz prozora u Taskbaru je moguće isključiti.

Slika 10 - ikona aplikacije u Taskbaru

Aplikaciju gasimo klasično na „X“ u gornjem desnom uglu aplikacije ili na Button Stop u Visual Studiu.

[3] http://windows.microsoft.com/en-us/windows7/products/features/windows-taskbar

4) Rad s kontrolama (dodavanje, svojstva, događaji)

U ovom poglavlju ćemo upoznati rad s kontrolama. Kontrole su elementi koje dodajemo na Windows Formu [4]poput *Button*[5]*, TextBox* [6]i mnogih drugi elemenata

a. Dodavanje kontrola na formu

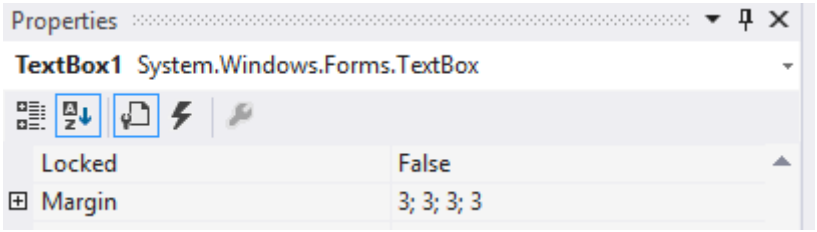
Kontrole koje se nalaze unutar Toolbox izbornika (vjerojatno na lijevoj strani Visual Studio razvojnog sučelja) možemo dodati na dva načina. Možemo napraviti dvoklik na kontrolu kako bi se ona pozicionirala unutar Forme ili možemo koristiti *Drag-n-drop* princip u kojem kontrolu povučemo na Formu.

Za sada ćemo dodati kontrolu TextBox.

Slika 11 - TextBox kontrola

b. Svojstva kontrola

Za svaku kontrolu možemo vidjeti njena svojstva u prozoru „Properties“ koji se vjerojatno nalazi s desne strane. Pazite da je odabrana ikonica „papira“ umjesto „munje“.



Slika 12 - Properties prozor

Svaka kontrola ima svoja određena svojstva koja mogu i ne moraju biti slična svojstvima druge kontrole. U našem slučaju, kontrola TextBox ima svojstva poput imena, pozadinske boje, vrste obruba i mnogih drugih.

Svaka kontrola ima i događaje koji se nad njom mogu dogoditi. Primjerice, jedan od mnogih događaja za TextBox kontrolu je kada kliknemo u nju. Drugi može biti kada izađemo iz te kontrole.

Za početak, uredimo svojstvo tako da početni tekst u ovoj kontroli bude „Hello Word“ umjesto dosadašnjeg stanja gdje nije bio upisan nikakav početni tekst.

[4] https://msdn.microsoft.com/en-us/library/dd30h2yb(v=vs.110).aspx
[5] https://msdn.microsoft.com/en-us/library/66817acc(v=vs.90).aspx
[6] https://msdn.microsoft.com/en-us/library/19z8k5by(v=vs.90).aspx

Potrebno je pronaći svojstvo „Text“ i upisati željeni tekst (u našem slučaju Hello World). Naravno, kako bi vidjeli svojstva neke kontrole, ona mora biti odabrana/selektirana.

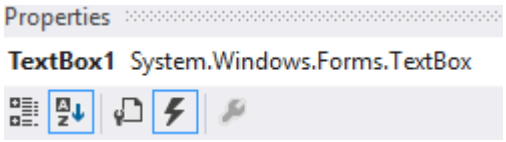
Slika 13 - Text svojstvo TextBox kontrole

Pogledamo li sada kontrolu, vidimo da je u nju upisan tekst „Hello World“.

Slika 14 - TextBox kontrola s upisanim tekstom u svojstvu Text

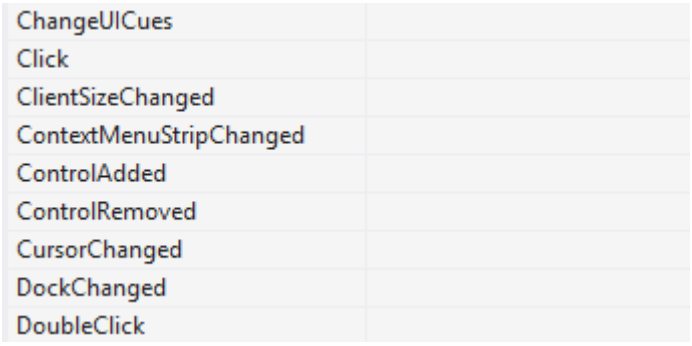
Isti tekst će biti upisan u kontrolu čim pokrenemo aplikaciju.

c. Događaji nad kontrolama



Slika 15 - Doagađji nad TextBox kontrolom

Broj događaja koji mogu biti povezani s TextBox kontrolom je zaista velik.



Slika 16 - Događaji kontrole TextBox

Želimo li napisati što će se dogoditi ako korisnik napravi klik na kontrolu onda trebamo gledati događaj „Click“. Napravimo dvoklik desno u desnom stupcu (desno od riječi „Click“) kako bi se generirao osnovni kod za taj događaj.

```
2 references
Public Class Form1

    0 references
    Private Sub TextBox1_Click(sender As Object, e As EventArgs) Handles TextBox1.Click

    End Sub
End Class
```

Slika 17 - Automatski geneirarani kod za događaj "Click" nad TextBox kontrolom

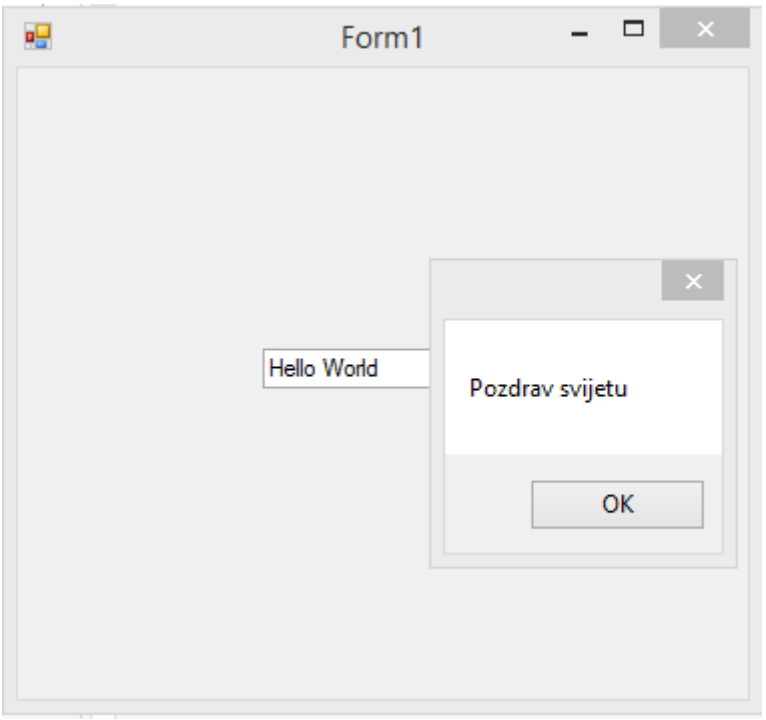
Sve što napišemo između „Private Sub“ i „End Sub“ dijela će se izvršiti kada kliknemo na TextBox.

Primjerice:

```
Public Class Form1

    Private Sub TextBox1_Click(sender As Object, e As EventArgs) Handles
TextBox1.Click
        MessageBox.Show("Pozdrav svijetu")
    End Sub
End Class
```

Ovaj kod će omogućiti iskočnu poruku (MessageBox u ovom slučaju) u kojem će pisati tekst „Pozdrav svijetu“ kada kliknemo na TextBox kontrolu.



Slika 18 - Prikaz MessageBoxa na Click događaj nad TextBox kontrolom

5) Varijable

Varijable koristimo za spremanje nekih vrijednosti. Primjerice, ako napravimo varijablu X onda u nju možemo spremiti neki broj koji korisnik unese.

a. Kreiranje varijabli

Kako bi kreirali varijablu koristimo ključnu riječ „Dim“, zatim navodimo ime varijable. Potom slijedi ključna riječ „As“ nakon čega se navodi tip varijable (tip podatka).

```
Public Class Form1
    Private Sub TextBox1_Click(sender As Object, e As EventArgs) Handles
TextBox1.Click
        Dim ime As String
        ime = "Saša"
        MessageBox.Show(ime)
    End Sub
End Class
```

Ovaj kod će stvoriti varijablu tipa String koja se zove „ime“. U drugoj liniji se toj varijabli dodjeljuje ime „Saša“. U trećoj varijablu se u MessageBox ispisuje vrijednost pridružena varijabli „ime“ odnosno riječ „Saša“.

b. Korištenje varijabli

Varijablama možemo dodijeliti neku vrijednost korištenjem operatora pridruživanja (=). Korištenjem istog operatora možemo i prebrisati trenutno dodijeljenu vrijednost i upisati novu. Ako ipak samo želimo nadodati još neku vrijednost na već postojeću, napisali bi ovakav kod:

```
        Dim ime As String
        ime = "Saša"
        ime = ime + " Fajković"
```

Ovaj kod će kreirati varijablu „ime“ koja je po tipu String. U drugom koraku će joj dodijeliti vrijednost „Saša“, a u trećem će na već postojeću vrijednost dodati još „ Fajković“.

Kraći način da zapišemo ovakvo pridruživanje je:

```
        Dim ime As String
        ime = "Saša"
        ime += " Fajković"
```

6) Tipovi podataka

a. Osnovni tipovi podataka

Osnovni tipovi podataka su najosnovniji brojčani i tekstualni tipovi podataka. Primjerice, cijeli brojevi su osnovni tipovi podataka. Kompleksni brojevi bi bili izvedeni tipovi podataka jer se oni temelje na Izvedenim tipovima podataka. „Character“ tip podatka koji omogućava spremanje jednog znaka (broj, slovo ili simbol) je osnovni tip dok je „String“ (spremanje više znakova odnosno riječi) je izveden jer se temelji na „Character“ tipu podatka.

b. Izvedeni tipovi podataka

Izvedeni tipovi podataka su oni koji su kreirani na osnovi osnovnih tipova podataka poput već spomenutih kompleksnih brojeva koji mogu biti kreirani na osnovi cijelih brojeva ili String tipa podatka koji je kreiran na osnovi Character tipa podatka.

Izvedeni tipovi podataka mogu biti već gotovi koji dolaze u sklopu programskog jezika (poput String tipa podataka) ili možemo kreirati svoje tipove podataka.

c. Integer

Integer tip podatka predstavlja cjelobrojnu vrijednost. To su vrijednosti poput 4,5,6,7,999, itd. Broj poput 3.14 nije Integer jer nije cijeli broj (ima decimalni dio).

U Visual Basicu će Inteer zauzeti 32 bita (4 bytea) i može prikazati sadržavati vrijednosti od -2,147,483,648 do 2,147,483,647.

```
Dim cijeliBroj As Integer
cijeliBroj = 7
```

Ovaj kod će stvoriti varijablu imena „cijeliBroj“ koja će po tipu biti Integer. U drugom koraku ćemo toj varijabli dodijeliti vrijednost 7.

Detaljnije: https://msdn.microsoft.com/en-us/library/06bkb8w2.aspx

d. Single

Single tip podatka su brojevi s decimalnim dijelom poput 3.142. Raspon ovog tipa podataka je od -3.4028235E+38 do 3.4028235E+38. Dakle od -3.4 na 38 do 3.4 na 38.

```
Dim decimalniBroj As Single
 decimalniBroj = 3.14
```

Ovaj kod će kreirati varijablu „decimalniBroj“ koja je po tipu Single, a u drugom koraku će se toj varijabli pridijeliti vrijednost 3.14

Detaljnije: https://msdn.microsoft.com/en-us/library/xay7978z.aspx

e. Double

Double tip podataka omogućava rad s decimalnim brojevima kao i Single tip podataka, samo što omogućava znatno veći raspon brojeva. Raspon ovog tipa podatka je od -1,79769313486232E+308 do 1,79769313486232E+308

```
Dim decimalniBroj As Double
decimalniBroj = 3.14
```

Ovaj kod će kreirati varijablu imena „decimalniBroj“ koja će po tipu biti Double. U drugom koraku se u nju sprema vrijednost 3.14.

Detaljnije: https://msdn.microsoft.com/en-us/library/x99xtshc.aspx

f. Char

Znakovi su brojevi, slova i simboli. Character tip podatka omogućava spremanje samo jednog znaka.

```
Dim znak As Char
znak = "k"
```

Ovaj kod će kreirati varijablu imena „znak“ koja će po tipu biti Char u koju ćemo zatim pohraniti vrijednost „k“.

Detaljnije: https://msdn.microsoft.com/en-us/library/7sx7t66b.aspx

g. Boolean

Boolean tipovi podataka su specifični jer imaju samo dvije moguće vrijednosti, a to su True i False.

```
Dim vrijednost As Boolean
vrijednost = True
```

Ovaj kod će kreirati varijablu imena „vrijednost“ koja je po tipu Boolean te će joj se dodijeliti vrijednost True.

Detaljnije: https://msdn.microsoft.com/en-us/library/wts33hb3.aspx

h. String

String je izvedeni tip podatka u koji možemo pohraniti više znakova, riječi ili rečenica.

```
        Dim rijec As String
       rijec = "Učim se programirati"
```

U ovom kodu će kreirati varijabla imena „rijec“ koja je po tipu String. U drugom koraku joj pridjeljujemo vrijednost „Učim se programirati“.

String vrijednosti se uvijek pišu unutar dvostrukih navodnika.

Detaljnije: https://msdn.microsoft.com/en-us/library/hzcd8ze0.aspx

7) Matematički operatori i operacije

a. Binarni operatori

U programiranju često ćete se susresti s binarnim operatorima. Većinu od njih znate. Zbrajanje, množenje, dijeljenje i oduzimanje sigurno već znate.

```
Dim a As Integer
Dim b As Integer
Dim rezultat As Integer

a = 10
b = 2

rezultat = a + b
rezultat = a - b
rezultat = a * b
rezultat = a / b
```

Ovaj kod će prvo kreirati tri Integer varijable. U drugom dijelu varijablama „a“ i „b“ dodjeljujemo vrijednosti 10 i 2. U trećem koraku isprobavamo osnovne matematičke operacije.

Postoji još jedan operator koji se relativno često koristi u programiranju, a to je operator **Modulo**. Ovaj operator kao rezultat daje ostatak pri cjelobrojnom dijeljenju.

```
a = 7
b = 3
rezultat = a Mod b
```

Rezultat će biti 1. To je ostatak pri cjelobrojnom dijeljenju brojeva 7 i 3. Broj 3 stane točno 2 cijela puta u broj 7. Dakle, 2x3=6. Razlika između 6 i 7 je jedan što nam daje rezultat pri cjelobrojnom dijeljenju.

Detaljnije: https://msdn.microsoft.com/en-us/library/wz3k228a.aspx

b. Skraćeni izrazi

Često (posebice kod petlji koje ćemo kasnije obraditi) ćete imati potrebu za povećavanjem ili smanjivanjem nekog cijelog broja za jedan. To je toliko učestalo u programiranju da ima i svoju kraticu.

Postupak povećanja za jedan se zove **inkrementacija.**

Postupak smanjivanja za jedan se zove **dekrementacija.**

```
a = 7
a = a + 1
a += 1
```

Prvo smo varijabli „a“ dodijelili vrijednost 7. Zatim smo tu vrijednost uvećali za 1 i sada „a“ iznosi 8. Skraćeni naziv tog zapisa je „a += 1“. Pošto je u drugom koraku varijabla „a“ poprimila vrijednost 8, nakon trećeg koraka će imati vrijednost 9.

Na isti način možemo i izvršiti dekrementaciju.

```
a = 7
a = a - 1
a -= 1
```

c. Matematičke operacije i prednost izvršavanja

Kao i što smo naviknuli, važnost operatora se poštuje i kod programiranja. Tako operatori za množenje i dijeljenje imaju prednost nad operatorima zbrajanja i oduzimanje.

```
a = 10
b = 2

rezultat = a + b * a – b
```

Rezultat će biti 28 jer se prvo množi 10*2 nakon čega se izvršavaju ostale operacije. Dobijemo dakle 10+10*2-2 što je jednako 10+20-2 što je 30-2 i to daje 28.

Naravno, ako želimo bolje kontrolirati tok izvođenja možemo koristiti formule.

```
a = 10
b = 2

rezultat = (a + b) * (a - b)
```

Ovaj kod će prvo zbrojiti vrijednosti u prvoj zagradi (10+2), zatim će oduzeti vrijednosti u drugoj zagradi (10-2) te konačno će se vrijednosti iz zagrada pomnožiti 12*8 što daje rezultat 96.

d. Logički operatori

U programiranju ćemo često koristiti logičke operatore „i“ i „ili“. Logički operatori daju kao konačnu vrijednost jednu od Boolean vrijednosti (True ili False).

And - Logički operator i zahtijeva da svi parametri/uvjeti budu ispunjeni kako bi on bio True. Ako barem jedan od uvjeta nije ispunjen, logički operator i će vratiti False.

```
Dim a As Integer
Dim b As Integer

a = 11
b = 15

If a > 10 And b < 20 Then
    MessageBox.Show("Između 10 i 20")
End If
```

Unutar If dijela (s If naredbom ćemo se upoznati kasnije) uspoređujemo dva uvjeta. Prvi uvjet je da je vrijednost varijable „a“ manja veća od 10. Drugi uvjet je da je vrijednost varijable „b“ manja od 20. Ako su oba dva uvjeta istovremeno zadovoljena, operator And (logičko i) će vratiti True. Ako nisu svi uvjeti ispunjeni, vratit će False.

Or – Logički operator ili

Za razliku od And, ovaj operator zahtijeva da minimalno jedan od uvjeta bude zadovoljen kako bi on vratio True. Mogu i svi uvjeti biti zadovoljeni i opet će vratiti True. Jedini put kada će vratiti False će biti ako niti jedan od uvjeta nije zadovoljen.

```
Dim a As Integer
Dim b As Integer

a = 7
b = 15

If a > 10 Or b < 20 Then
    MessageBox.Show("Između 10 i 20")
End If
```

U ovom primjeru uspoređujemo je li vrijednost varijable „a“ veća od 10 kao prvi uvjet. Drugi uvjet je da je vrijednost varijable „b“ manja od 20. Vrijednost varijable „a“ nije veća od 10 i prvi uvjet nije zadovoljen ali je drugi uvjet zadovoljen jer je vrijednost varijable „b“ manja od 20. S obzirom da je jedan od uvjeta zadovoljen, logički operator ili će vratiti True.

Detaljnije: https://msdn.microsoft.com/en-us/library/wz3k228a.aspx

8) Pseudo kod i dijagram toka

Iako se na prvu čini kako se programiranje svodi samo na neprestano pisanje kôda, istina je ipak nešto drugačija. Prilikom izrade aplikacija, programeri najčešće znaju krajnju točku odnosno čemu aplikacija koju izrađuju treba služiti (i početak naravno a to je prazan *editor*). Kako bi si mogli olakšati ali i predočiti probleme s kojima će se susresti, programeri problem razbijaju u manje probleme a te probleme u još manje i sve do određenih dijelova koji su im znatno jednostavniji za riješiti. Po završetku jednog dijela pomiču se na drugi. Ovakav pristup se također zove **DEKOMPOZICIJA**. Papir i olovka su dobri prijatelji svakom programeru s kojima se često i druže. Dva su osnovna načina prikazivanja problema. Za prvi primjer uzet ćemo da je problem izgradnja drvene kuće a u drugom primjeru ćemo izračunavati površinu naše kuće.

a. Pseudo kôd

Iako stručan i zanimljiv izraz, pseudo kôd se zaista odnosi na tekstualno opisivanje problema. Napravimo li malu dekompoziciju problema gradnje kuće možemo doći do više manjih i jasnijih dijelova:

1) izgraditi temelje
2) izgraditi drvenu konstrukciju tijela kuće
3) izgraditi drvenu konstrukciju krova
4) Dodati daske za tijelo kuće
5) Dodati krov kuće

Svaki od tih problema možemo rastaviti na još manje cjeline pa tako izgradnju temelja možemo podijeliti prvo na obradu zemlje a zatim na betoniranje. Svaki od tih problema možemo rastaviti na još manje problemske jedinice i tako više-manje u nedogled. **Napomena** – treba znati granicu kada prestati razdvajati probleme u manje cjeline kako se ne bi previše vremena utrošilo na pisanje pseudo kôda za koji postoji dobra šansa da na kraju neće biti ni uporabljiv.

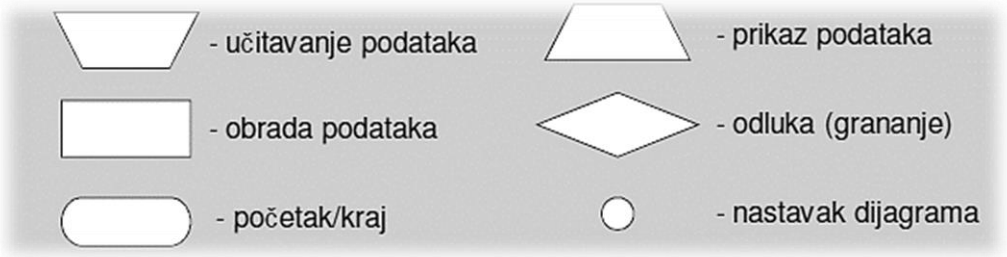
Naš pseudo kod bi za prvih pet koraka izgledao :

```
izgradi temelje
napravi konstrukciju
napravi krov
dodaj daske
dodaj krov
```

Kao što vidite, opis problema kuće je sada razumljiviji i smisleniji a nismo napisali niti jednu s*int*aksno ispravnu rečenicu odnosno liniju kôda.

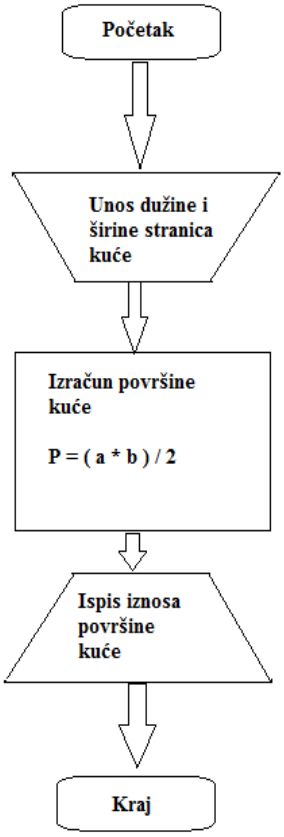
b. Dijagram toka

Pristupiti dijagramu toka možemo kao i pseudo kôdu. Osnovni način razmišljanje je isti a razlika je samo u prikazu. Za razliku od pseudo kôda, dijagram toka karakterizira grafičko skiciranje problema. Za potrebe dijagrama toka postoji šest najčešće korištenih simbola čijih se značenja kao niti simbola nije potrebno pridržavati ali radi konvencije je poželjno.



Slika 19 - Simboli koji se koriste u dijagramu toka [7]

- Radi lakšeg pojašnjavanja, zamislite da je naša kuća pravokutnog oblika (ravan krov, jednaka lijeva i desna strana kuće te prednja i stražnja strana)

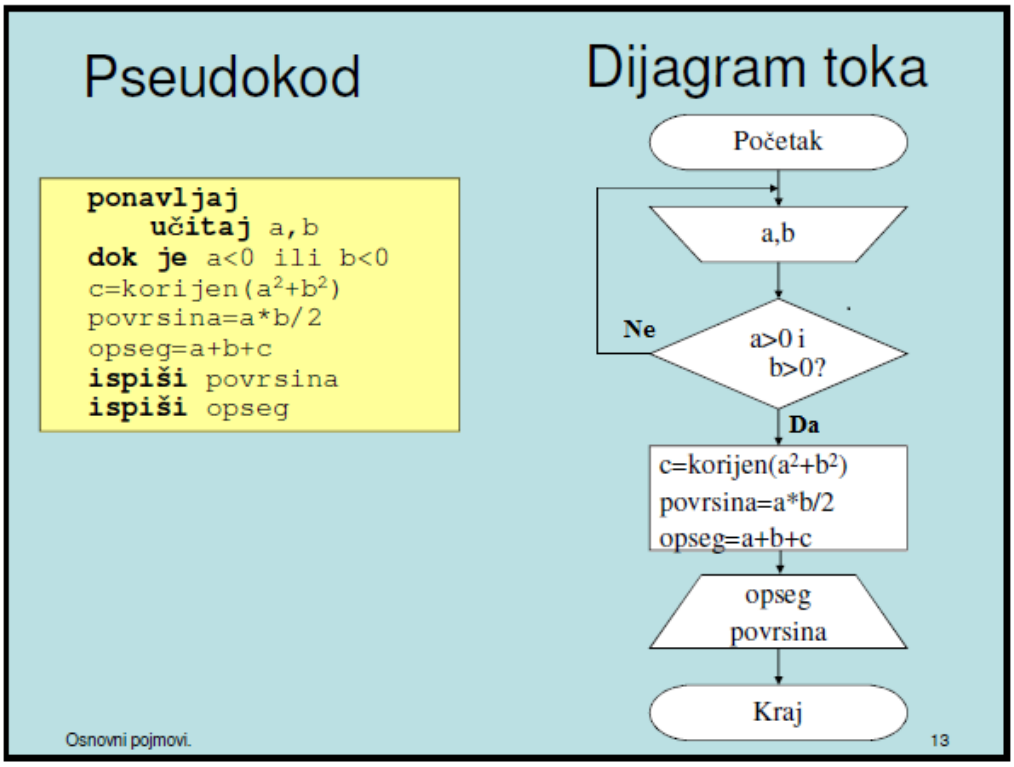


Slika 20 - Dijagram toka pri izračunu površine kuće

[7] izvor : Ivo Beroš, VERN' pdf – POINT – Uvod u programiranje – 2010. – stranica 2

c. Primjer pseudko kôda i dijagrama toka

U ovom primjeru, zadatak je izračunati površinu i opseg pravokutnog trokuta na osnovu veličina stranica koje upisuje korisnik te u konzolu ispisati vrijednosti površine i opsega.



Slika 21 - Primjer pseudo kôda i dijagrama toka [8]

[8] izvor : Ivo Beroš pdf materijali – POINT – Uvod u programiranje – 2010. – stranica 2

9) Komentari

Komentari su jako važan dio programiranja. Sve što obilježimo kao komentar se ne izvodi. Posebice su korisni jer olakšavaju programeru posao u vidu razumijevanja napisanog koda.

```
rezultat = (a + b) * (a - b)
' Ovaj kod će prvo riješiti zagrade, a zatim će ih pomnožiti.
```

Komentare započinjemo znakom apostrofa ('). Sve napisano iza tog znaka je komentar.

Visual Studio nudi korisnu kraticu za komentiranje više linija odjednom. Obilježimo linije koje želimo komentirati i pritisnemo Ctrl+K. Ako želimo maknuti komentare, obilježimo linije koda s kojih to želimo napraviti i pritisnemo Ctrl+U.

Detaljnije: https://msdn.microsoft.com/en-us/library/bx185bk6.aspx

10) Prekid ispisa u novi red

Ako želimo napraviti prekid ispisa (primjerice u TextBox kontrolu koja ima postavljeno Multiline svojstvo na True) i prijeći u novi red, koristimo **Enviornment.NewLine.**

```
txtFor.Text += Environment.NewLine
```

Ovaj kod će u svojstvo Text, kontrole TextBox imena txtFor dodati prekid u novi red.

```
        txtFor.Text = "Ovo je vježba s"
        txtFor.Text += Environment.NewLine
        txtFor.Text += "prekidima u novi red"
```

Ovaj kod će ispisati u textBox kontrolu (s postavljenim MultiLine svojstvom na True) u prvom redu tekst „Ovo je vježba s“, a u drugi red „prekidima u novi red“.

11) Uvjetna kontrola toka

Programski kod se izvršava odozgora prema dolje. U tom postupku želimo moći kontrolirati što se i na koji način izvodi. Dodatno tu želimo ubaciti mogućnost izvršavanja koda na jedan način ako je korisnik unio jednu vrijednost i izvođenje na drugi način ako je unio neku drugu vrijednost.

a. If

If nam omogućava kontrolu toka uz kontroliranje uvjeta. Primjerice, želimo ispisati nešto u aplikaciju samo ako je neki uvjet zadovoljen. Primjerice, želimo ispisati tekst „Pozitivan rezultat“ samo ako je rezultat pozitivan, ali ne i ako je negativan.

```
a = 10
b = 2

rezultat = a - b
If rezultat > 0 Then
    MessageBox.Show("Pozitivan je")
End If
```

Ovaj kod prikazuje da će se ispisati tekst „Pozitivan je“ ako je vrijednost rezultata veća od nule odnosno pozitivna. Ako je ta vrijednost manja od nule, taj dio koda se neće izvršiti.

If naredbu započinjemo s ključnom riječi **If** nakon čega slijedi uvjet te ključna riječ **Then**. Nakon toga pišemo kod koji će se izvršiti ukoliko je uvjet zadovoljen. Cijelu If naredbu završavamo ključnom riječi **End If**.

Detaljnije: https://msdn.microsoft.com/en-us/library/752y8abs.aspx

b. Else

Često poželimo da se izvrši jedan kod ako je neki uvjet zadovoljen, no ako nije neka se izvrši neki drugi kod.

```
rezultat = a - b
```

```
If rezultat > 0 Then
    MessageBox.Show("Pozitivan je")
Else
    MessageBox.Show("Negativan je")
End If
```

U ovom slučaju će se testirati vrijednost varijable „rezultat“ te ako je veća od 0 onda će se ispisati „Pozitivan je“. Ako to nije slučaj, izvršit će se kod koji je naveden unutar **Else** bloka. U ovom slučaju, u Else bloku se nalazi naredba za ispis teksta „Negativan je“.

Detaljnije: https://msdn.microsoft.com/en-us/library/752y8abs.aspx

c. Else If

Često ćemo imati potrebu kontrolirati više uvjeta. Primjerice, u donjem primjeru testiramo mogućnosti da je vrijednost varijable „rezultat“ veća od nule, jednaka nuli ili manja od nule.

```
rezultat = a - b
If rezultat > 0 Then
    MessageBox.Show("Pozitivan je")
ElseIf rezultat = 0 Then
    MessageBox.Show("Rezultat je nula")
Else
    MessageBox.Show("Negativan je")
End If
```

Detaljnije: https://msdn.microsoft.com/en-us/library/752y8abs.aspx

d. Select Case

Select Case je ono što u većini drugih jezika nazivamo Switch naredbom. Ova naredba nam omogućava kontrolu toka nad više mogućih rezultata, slično kao i kombinacija više Else If naredbi, no korištenjem Select Case kontroliramo vrijednosti sam

```
Select Case rezultat
      Case 1 To 10
          MessageBox.Show("Između 1 i 10")
      Case 0
          MessageBox.Show("Nula je")
      Case 11 To 20
          MessageBox.Show("Između 11 i 20")
      Case Is < 0
          MessageBox.Show("Manje od nule")
End Select
```

Ako je vrijednost varijable rezultat između 1 i 10, ispisat će se tekst „Između 1 i 10“. Ako je vrijednost jednaka nuli, ispisat će se tekst „Nula je“, ako je između 11 i 20, ispisat će se tekst „Između 11 i 20“, a ako je vrijednost negativna, ispisat će se „Manje od nule“.

Detaljnije: https://msdn.microsoft.com/en-us/library/cy37t14y.aspx

12) Petlje

Petlje su često korištene u programiranju jer omogućavaju izvršavanje ponavljajućeg koda.

a. For

For petlja je jedna od najčešće korištenih petlji. Njena sintaksa je takva da počinjemo s ključnom riječi For nakon čega pišemo ime nove varijable koja će služiti kao brojač. U ovom slučaju je to varijabla „i“. Potom pišemo ključnu riječ „As“ i definiramo tip podatka kojeg će biti ta varijabla što je u ovom slučaju Integer. Nakon tipa podatka pišemo znak jednakosti, zatim vrijednost koja će biti dodijeljena toj varijabli. Nakon početne vrijednosti pišemo ključnu riječ „To“ koja određuje do kojeg broja ćemo brojati i na kraju navodimo vrijednost do koje će se brojati.

For petlju završavamo ključnom riječi „Next“ a sve između te riječi i početka petlje je kod koji će se izvršavati u svakom krugu petlje. Taj svaki krug se zove **iteracija**.

```
For i As Integer = 1 To 3
    MessageBox.Show("Ponavljanje")
Next
```

Pogledajmo što se ovdje točno događa korak po korak. Prvo je i=1. Provjerava se da li je vrijednost „i“ manja od 3. Uvjet je zadovolje i petlja se počinje izvršavati. Ispisuje se na ekran „Ponavljanje“ i vrijednost varijable „i“ se povećava za jedan. Sada je i=2 što je još uvijek zadovoljeno. Ponovno se ispisuje „Ponavljanje“ nakon čega se „i“ ponovno povećava i sada je 3 što još uvijek zadovoljava uvjet jer nije veće od 3. Ponovno se ispisuje „Ponavljanje“ te se „i“ povećava na 4. Sada je i=4 i ta vrijednost je veća od 3 te se For petlja prestaje izvršavati i nastavlja se daljnje izvršavanje koda koji je napisan iza ključne riječi Next.

Ako je potrebno, moguće je odrediti i za koliko se vrijednost brojača varijable mijenja u svakom koraku korištenjem ključne riječi **Step**.

```
For i As Integer = 1 To 5 Step 3
    MessageBox.Show("Ponavljanje")
Next
MessageBox.Show("Kraj")
```

Vrijednost varijable „i“ je 1 i petlja će se izvršavati sve dok „i“ ne prijeđe 5. U prvoj iteraciji se ispiše „Ponavljanje“ te se vrijednost varijable „i“ poveća na 4. Kreće izvođenje druge iteracije jer je 4 manje od 5. Ispiše se „Ponavljanje“ te je sada vrijednost varijable „i“ jednaka 7 te se petlja prestaje izvršavati.

Detaljnije: https://msdn.microsoft.com/en-us/library/5z06z1kb.aspx

b. While

While petlja je uglavnom jednostavnija (gledano po sintaksi) od For petlje. Obje rade identičnu stvar, odnosno izvršavaju neki kod neki određeni broj puta dok god je uvjet zadovoljen.

```
b = 2

While b > 0
    MessageBox.Show("Ispis")
    b -= 1
End While
```

Vrijednost varijable „b“ je 2. Korištenjem While petlje želimo ispisivati na ekran tekst „Ispis“ dok god je vrijednost varijable „b“ veća od nule. U prvoj iteraciji b=2, ispisuje se „Ispis“ i „b“ se smanjuje za jedan. „b“ je sada 1 što i dalje zadovoljava uvjet pa se ponovno ispisuje „Ispis“ i „b“ se smanjuje za jedan. „b“ je sada 0 što nije veće od 0 i petlja prestaje s izvršavanjem.

Detaljnije: https://msdn.microsoft.com/en-us/library/zh1f56zs.aspx

c. Do While / Until

Ova petlja je jedina koja će se uvijek izvršiti barem jednom. Razlog tome je što se prvo izvršava kod, a tek se zatim radi provjera uvjeta. Postoje dvije varijacije ove petlje – Do While i Do Until.

Do While se izvršava dok god je uvjet zadovoljen odnosno True. **Dow Until** se izvršava dok god je uvjet False.

```
b = 2
Do
    MessageBox.Show("Ispis")
    b -= 1
Loop While b > 0
```

Do While - Prvo se ispisuje „Ispis“ nakon čega se „b“ umanjuje za jedan. „b“ sada iznosi 1 i tek se sada ide u provjeravanje uvjeta. S obzirom da je 1 veće od 0, uvjet je zadovoljen odnosno True. Kreće druga iteracija i ponovo se ispisuje „Ispis“ nakon čega se „b“ umanjuje za jedan i sada iznosi 0 što nije veće 0 i prestaje izvršavanje.

```
b = 2
Do
    MessageBox.Show("Ispis")
    b -= 1
Loop Until b > 0
```

Do Until - Ispisuje se „Ispis“ i „b“ se umanjuje za jedan. Sada „b“ iznosi 1. S obzirom da koristimo „Do Until“ petlja prestaje s izvršavanjem kada je uvjet True odnosno kada je zadovoljen, 1 je veće od 0 što daje True rezultat zadanog uvjeta i petlja prestaje s izvršavanjem.

Detaljnije: https://msdn.microsoft.com/en-us/library/eked04a7.aspx

13) Ugniježđene petlje

Do sada smo vidjeli kako koristiti pojedinačno petlje no često ćemo se susresti s potrebom postavljanja jedne petlje unutar druge. Ovaj princip se zove **ugniježđene petlje**.

```
For i As Integer = 1 To 10

            For j As Integer = 1 To 10
                txtFor.Text += (i * j).ToString() + " "
            Next

            txtFor.Text += Environment.NewLine
        Next
```

Ovaj primjer pokazuje dvije *for* petlje, jedna unutar druge koje će kreirati tablicu množenja od 1 do 10. Prvo ulazimo u vanjsku for petlju i u prvom koraku i=1. Zatim se ulazi u unutarnju for petlju gdje je j=1. Ta dva broja se pomnože i dobije se rezultat 1. Zatim se pokreće drugi krug (iteracija) unutarnje petlje pa je sada i=1, a j=2. U trećoj iteraciji i=1, j=3. Zadnji krug unutarnje petlje će biti kada je i=1, a j=10. Ta dva broja se pomnože i ispišu te se završava unutarnja petlja i dalje se izvršava nadolazeći kod koji se nalazi unutar vanjske for petlje i tu nailazimo na *Enviornment.NewLine* što će generirati prelazak u novi red i time završava prva iteracija vanjske petlje.

Ponovno se pokreće vanjska petlja, ali sada njen drugi krug. Sada je i=2 a j=1. Ponovno se pokreće drugi krug unutarnje petlje pa je sada i=2, j=2. po tom principu će se kod izvršavati sve dok se vanjska petlja ne izvrti od 1 do 10, a unutarnja će se za svaku iteraciju vrtjeti od 1 do 10.

Naravno, moguće je raditi ugniježđene petlje kombiniranjem različitih petlji. Primjerice, unutar FOR petlje postavimo While petlju.

14) Prekidanje petlji

Ponekad ćemo željeti prekinuti izvođenje petlje iako nisu svi koraci izvršeni. Primjerice, želimo ispisati sve brojeve od 1 do nekog broja kojeg korisnik unese, ali želimo zeznuti korisnika i čim petlja dođe do broja 5 želimo prekinuti njeno izvođenje. Za ovu potrebu koristimo ključnu riječ **Exit** u kombinaciji s imenom petlje koju želimo prekinuti.

a. Exit

```
b = 1
While b < 100
    MessageBox.Show("Test")
    If b = 5 Then
        Exit While
    End If
    b += 1
End While
```

Petlja kreće s izvršavanjem i ispisuje se „Test“ i nakon svakog ispisa se vrijednost varijable „b“ povećava za jedan. U trenutku kada varijabla „b“ poprimi vrijednost 5 i uđe se u izvršavanje petlje biti će zadovolje „If“ uvjet (If b = 5 Then) i tada se ulazi u If unutar kojeg piše „Exit While“ što će uzrokovati prekid u izvršavanju petlje.

Na isti princip bi koristili prekid petlje i u drugim petljama (For, Do While)

Detaljnije: https://msdn.microsoft.com/en-us/library/t2at9t47.aspx

b. Continue

Za razliku od Exit naredbe, Continue će samo prekinuti trenutnu iteraciju petlje ali neće u potpunosit prekinuti izvršavanje petlje. Pišemo ključnu riječ Continue i ime petlje kojoj pripada.

```
b = 1
While b < 3
    If b = 2 Then
        Continue While
    End If
    MessageBox.Show("Ispis 2")
    b += 1
End While
```

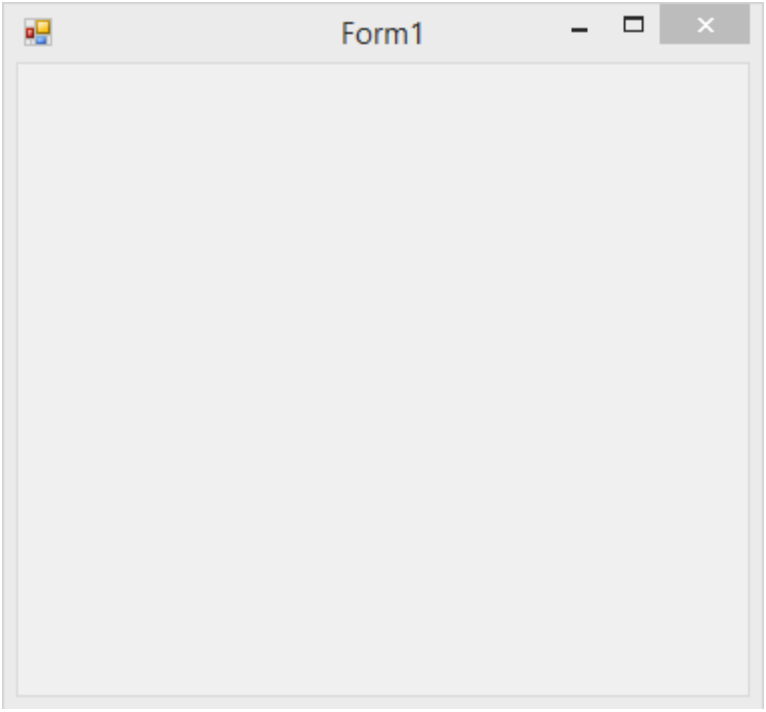
Krećemo s izvršavanje i „b“ je 1 što je manje od 100 i ulazimo u petlju. Zatim provjeravamo u „If“ dijelu je li „b“ 2. U ovom slučaju nije pa nastavljamo dalje. Ispisuje se tekst „Ispis 2“ i „b“ se uvećava za jedan. Sada je b=2 što je i dalje manje od 3 te počinjemo s izvršavanjem druge iteracije petlje. Čim smo ušli u petlju radi se provjera „If“ dijela koji je sada zadovoljen i ulazimo u taj „If“ gdje nalazimo naredbu Continue i prekida se ova iteracija petlje. Počinje novo izvršavanje no sada treba paziti jer je vrijednost varijable „b“ ostala nepromijenjena i još uvijek iznosi 2. U ovom primjeru koda, ova se petlja nikada neće prestati izvršavati jer će „b“ zauvijek ostati manji od 3.

Ovakav događaj je uvijek neželjena pojava i zove se **BESKONAČNA PETLJA**.

Detaljnije: https://msdn.microsoft.com/en-us/library/801hyx6f.aspx

15) Windows forma[9]

Windows forma je temelj za izgradnju naše aplikacije. Na nju ćemo dodavati ostale kontrole, a nerijetko ćemo imati više formi.



Slika 22 - Windows Forma

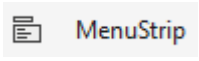
a. Windows Forma - Najčešća svojstva

Ime	Opis
(Name)	Ime forme koje koristimo u kodu
AutoSize	Forma se sama prilagođava po dimenziji ovisno o broju i položaju kontrola
AutoSizeMode	Način automatskog prilagođavanja dimenzije forme. Možemo odabrati između povećavanja ili povećavanja i smanjivanja
MaximizeBox	Kućica za maksimiziranje forme je omogućena ili onemogućena
MinimizeBox	Kućica za minimiziranje forme je omogućena ili onemogućena
ShowInTaskbar	Hoće li se ikonica od forme prikazati u Taskbaru ili ne
Size	Veličina forme
StartPosition	Odabir početne lokacije forme kada se ona prvi put prikaže
Text	Tekst koji se ispisuje na vrhu forme

[9] https://msdn.microsoft.com/en-us/library/ms172749.aspx

16) MenuStrip[10]

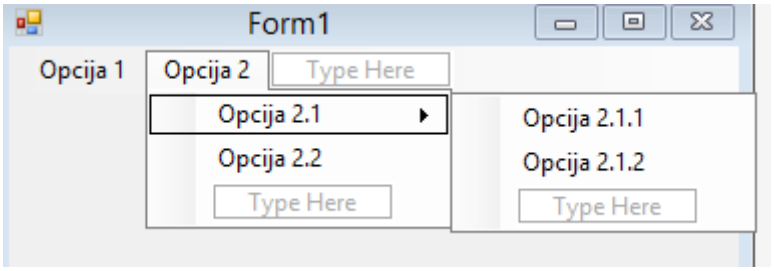
MenuStrip je kontrola koja omogućava klasičnu izradu meni izbornika koje smo naviknuli viđati na Windows platformi.



Slika 23 - MenuStrip 1

Slika 24 - MenuStrip 2

Stavke dodajemo jednostavnim upisivanjem teksta koji će predstavljati izbornike. Možemo dodavati i pod-izbornike.



Slika 25 - MenuStrip 3

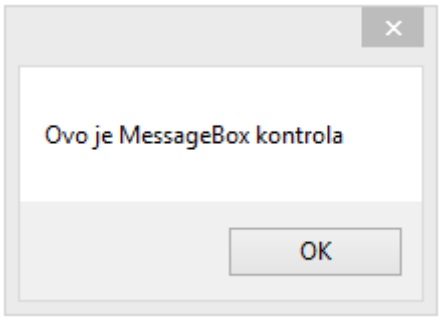
a. MenuStrip - Najčešća svojstva

Ime	Opis
(Name)	Ime MenuStrip kontrole koje koristimo u kodu
Dock	Odabir lokacije gdje će kontrola biti pričvršćena u odnosu na formu
Stretch	Hoće li se kontrola rastegnuti po dužini forme na kojoj se nalazi ili ne

[10] https://msdn.microsoft.com/en-us/library/system.windows.forms.menustrip(v=vs.110).aspx

17) MessageBox / MsgBox[11]

MessageBox možemo zamisliti kao iskočni prozor za obavijesti s mogućnošću prikaza tekstualnih informacija. MessageBox ne povlačimo na formu kao što smo do sada to radili već to radimo putem koda.



Slika 26 – MessageBox

```
MessageBox.Show("Ovo je MessageBox kontrola")
```

Obavezno koristimo Messagebox.Show(), a unutar oblih zagrada navodimo što se treba prikazati u MessageBoxu. Detalje zašto se ovako piše nećemo obraditi u ovom dijelu, ali možete zapamtiti da ćete uvijek pisati MessageBox.Show() kako bi ispisali tekst u iskočnoj poruci.

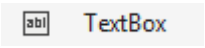
Napomena: MessageBox.Show() možemo pozvati i na kraći naziv. **MsgBox()** je ekvivalent MessageBox.Show().

[11] https://msdn.microsoft.com/en-us/library/139z2azd(v=vs.90).aspx

18) TextBox[12]

TextBox kontrola se koristi za unos tekstualnih informacija. Po početnim postavkama ova kontrola ne omogućava širenje na više redova, ali tu opciju možemo omogućiti kroz svojstva kontrole.

Slika 27 – TextBox 1



Slika 28 - TextBox 2

a. TextBox - Najčešća svojstva

Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
BackColor	Pozadinska boja kontrole
BorderStyle	Vrsta obruba kontrole
Multiline	Širenje kontrole na više redova
PasswordChar	Simbol koji će se prikazivati kao zaštita za unos lozinke
ReadOnly	Nije moguće upisivanje već samo čitanje teksta iz kontrole
TextAlign	Položaj teksta u odnosnu na kontrolu

[12] https://msdn.microsoft.com/en-us/library/19z8k5by(v=vs.90).aspx

19) Label[13]

Label koristimo za prikaz teksta. Ne očekuje se interakcija korisnika nad ovom kontrolom već služi samo za prikaz teksta.

Slika 29 - Label 1

Slika 30 - Label 2

a. Label - Najčešća svojstva

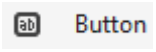
Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
BackColor	Pozadinska boja kontrole
Text	Tekst koji je upisan u kontrolu

[13] https://msdn.microsoft.com/en-us/library/9hwzeyc9(v=vs.90).aspx

20) Button[14]

Button koristimo za pokretanje neke akcije.

Slika 31 - Button 1



Slika 32 - Button 2

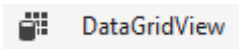
a. Najčešća svojstva

Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
FlatStyle	Određuje prikaz
Text	Tekst koji je upisan u kontrolu
TextAlign	Određuje poziciju teksta u kontroli

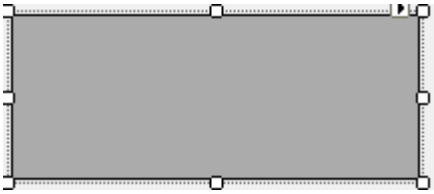
[14] https://msdn.microsoft.com/en-us/library/windows/apps/xaml/jj153345.aspx

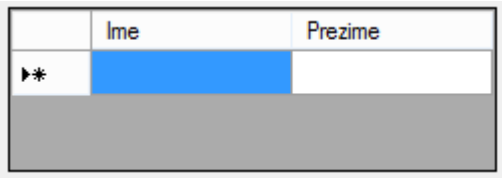
21) DataGridView[15]

DataGridView kontrola služi za tabličan prikaz kontrola.



Slika 33 - DataGridView 1


Slika 34 - DataGridView 2



Slika 35 - DataGridView 3

a. DataGridView - Najčešća svojstva

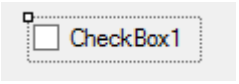
Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
AllowUserToAddRows	Dopuštamo ili onemogućavamo korisniku da dodaje redove
AllowUserToDeleteRows	Dopuštamo ili onemogućavamo korisniku da briše redove
BorderStyle	Određujemo vrstu obruba kontrole
MultiSelect	Omogućavamo ili onemogućavamo odabir više redova
ReadOnly	Određujemo je li kontrola postavljena samo za čitanje ili ne

[15] https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview(v=vs.110).aspx

22) CheckBox[16]

CheckBox nam omogućava odabir više mogućnosti.

Slika 36 - CheckBox 1



Slika 37 - CheckBox 2

a. CheckBox - Najčešća svojstva

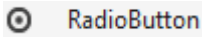
Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
CheckAlign	Odabir lokacije kućice za obilježavanje
Checked	Odabir je li postavljena kvačica u kućicu ili ne čim se pokrene aplikacija
Text	Tekst koji se ispisuje u kontroli
TextAlign	Odabir lokacije teksta u kontroli

[16] https://msdn.microsoft.com/en-us/library/kk8bb7ac(v=vs.90).aspx

23) RadioButton[17]

RadioButton nam omogućava odabir samo jedne od opcija.

Slika 38 - RadioButton 1



Slika 39 - RadioButton 2

a. RadioButton - Najčešća svojstva

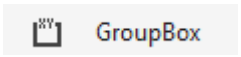
Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
CheckAlign	Odabir lokacije kružića za obilježavanje
Checked	Odabir je li postavljena kvačica u kružić ili ne čim se pokrene aplikacija
Text	Tekst koji se ispisuje u kontroli
TextAlign	Odabir lokacije teksta u kontroli

[17] https://msdn.microsoft.com/en-us/library/6xtydwb3(v=vs.90).aspx

24) GroupBox[18]

GroupBox omogućava sortiranje kontrola po smislenijem rasporedu.

Slika 40 - GroupBox 1



Slika 41 - GroupBox 2

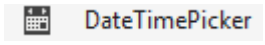
a. GroupBox - Najčešća svojstva

Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
Text	Tekst koji se ispisuje u kontroli

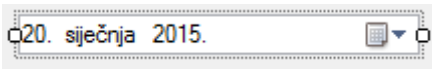
[18] https://msdn.microsoft.com/en-us/library/system.windows.forms.groupbox(v=vs.110).aspx

25) DateTimePicker[19]

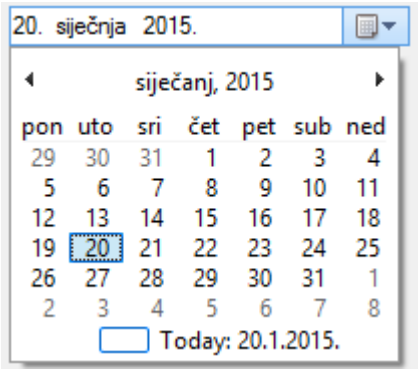
DateTimePicker kontrola nam omogućava jednostavan odabir datuma.



Slika 42 - DateTimePicker 1



Slika 43 - DateTimePicker 2



Slika 44 - DateTimePicker 3

a. Najčešća svojstva

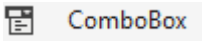
Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
Format	Određuje način prikaza datuma
MinDate	Najmanji dozvoljeni datum
MaxDate	Najveći dozvoljeni datum

[19] https://msdn.microsoft.com/en-us/library/aa231249(v=vs.60).aspx

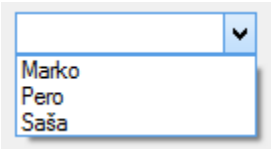
26) ComboBox[20]

ComboBox kontrola omogućava prikaz podataka korištenjem padajućeg izbornika.

Slika 45 - ComboBox 1



Slika 46 - ComboBox 2



Slika 47 - ComboBox 3

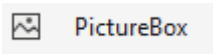
a. Najčešća svojstva

Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
DropDownStyle	Određuje način prikaza padajućeg izbornika

[20] https://msdn.microsoft.com/en-us/library/aa240832(v=vs.60).aspx

27) PictureBox

PictureBox je kontrola za prikaz slike unutar apliakcije.



Slika 48 - PictureBox 1

Slika 49 - PictureBox 2

a. Najčešća svojstva

Ime	Opis
(Name)	Ime kontrole koje koristimo u kodu
BorderStyle	Određujemo vrstu obruba
SizeMode	Odabir kako će se slika skalirati unutar zadanih veličina kontrole

28) Svojstva kontroli u kodu[21]

Svojstva kontroli smo upoznali kroz grafički prikaz u izborniku Properties. Sve što je tamo popisano je negdje napisano u kodnom zapisu. U ovom dijelu ćemo vidjeti kako čitati i upisivati svojstva kontrole.

a. Upisivanje vrijednosti

```
TextBox1.Text="Novi tekst"
```

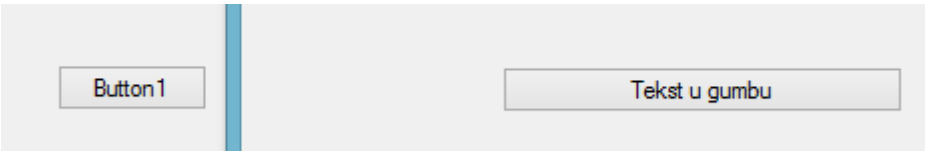
Ovaj kod će omogućiti upisivanje teksta u kontrolu TextBox1 tako što pristupamo njenom svojstvu i upisujemo vrijednost „Novi tekst“.

```
Button1.Text = "Tekst u gumbu"
```

Ovim kodom ćemo kontroli Button1 promijeniti tekst da piše „Tekst u gumbu“.

```
Button1.Width = 200
```

Ovim kodom mijenjamo širinu gumba. S lijeve strane imamo gumb prije pokretanja aplikacije. S desne strane vidimo gumb nakon pokretanja aplikacije. Vidimo da se tekst promijenio i da je širina promijenjena.



Slika 50 - Promjena svojstava Button kontrole

b. Čitanje vrijednosti

Na sličan način kako upisujemo vrijednosti za svojstva kontrole, tako ih i čitamo.

```
        Dim duzina As Integer
        duzina = Button1.Width
```

Ovaj kod će prvo stvoriti varijablu imena „duzina“ koja je po tipu Integer. U drugom koraku ćemo toj varijabli dodijeliti vrijednost jednaku dužini buttona.

```
        Dim tekst As String
        TextBox1.Text = TextBox1.Text
```

Ovaj kod će stvoriti varijablu imena „tekst“ koja je po tipu String. U drugom koraku u tu varijablu spremamo tekst koji je upisan u TextBox1 kontroli.

[21] https://msdn.microsoft.com/en-us/library/zzt5x46b(v=vs.90).aspx

29) Funkcije – općenito

Funkcije nam omogućavaju brže, jednostavnije i efektivnije iskorištavanje koda. Funkcije imaju svoje ime koje možemo pozvati i izvršit će se kod napisan u njima. Tako primjerice kod od 100 linija koji se ponavlja možemo staviti u funkciju i umjesto da na dva mjesta imamo 100 linija istog koda, samo bi pozvali funkciju što je jedna linija koda čime je povećana čitljivost koda. Uz to se nadovezuje i lakše održavanje koda jer ako vidimo da smo napravili grešku u kodu, mijenjat ćemo samo na jednom mjestu, a ne na dva ili više mjesta.

Sve funkcije imaju neke zajedničke karakteristike:

Ključna riječ kojom započinjemo - **Function**

Ime funkcije (proizvoljno ako kreiramo svoju)

Eventualni parametri unutar oblih zagrada - **( )**

Tip podatka koji funkcija vraća – **As ReturnType**

Naredba za kraj izvršavanja funkcije i vraćanje podatka – **Return**

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        MessageBox.Show(prikaziImePrezime())
    End Sub
    Function prikaziImePrezime() As String
        Return "Saša Fajković"
    End Function
End Class
```

U ovom primjeru imamo klasu Form1. Ovo je automatski generirano prilikom kreiranja forme koje se zove Form1. Unutar klase prvo nalazimo funkciju koja se pokreće na događaj „Form1_Load“ odnosno prilikom pokretanja Forme 1. Unutar ove funkcije smo u MessageBox ispisali tekst koji će vratiti funkcija „prikaziImePrezime“.

Prvo pogledajmo funkciju **prikazImePrezime()**. Korištenjem ključne riječi **Function** govorimo da želimo stvoriti novu funkciju. Zatim navodimo ime funkcije što je u ovom slučaju **prikazImePrezime**. Nakon imena navodimo ključnu riječ **As** i onda tip podatka koji funkcija vraća što je u našem slučaju **String**.

Funkciju završavamo ključnom riječi **End Function**. Sve između početka i kraja funkcije će se izvršiti kada se funkcija pozove. U našem slučaju funkcija izvršava samo jednu liniju koda. Ključna riječ **Return** označava kraj izvršavanja funkcije i vraća neki podataka. Moramo paziti da taj podatak odgovara tipu podatka koji smo odredili prilikom stvaranja funkcije (Function prikazImePrezime() As String). U ovom slučaju će se vratiti tekst **Saša Fajković**.

30) Korištenje gotovih funkcija

Iako ćemo često kreirati svoje funkcije, često ćemo i koristiti gotove funkcije. Jednu takvu operaciju smo već koristili prilikom ispisa u MessageBox.

a. Kako pozvati funkciju?

Funkciju pozivamo tako da navedemo njeno ime i eventualne parametre.

```
prikaziImePrezime()
```

Ovim kodom pozivamo funkciju prikaziImePrezime bez ikakvih parametara.

```
prikazImePrezime(„Saša“, „Fajković“)
```

Ovim kodom pozivamo funkciju prikazImePrezime kojoj predajemo dva String parametra. Prvi parametar je Saša, a drugi Fajković.

b. Metode = funkcije nad kontrolama

Metode su zapravo naziv za funkcije koje su vezane za objekte. Za sada ćemo objektima smatrati kontrole.

Prisjetimo se kako smo ispisivali tekst u MessageBox. Pisali smo MessageBox.Show(„Neki tekst za ispis“). Upravo ovaj **Show()** je metoda. Kao što MessageBox ima svoje metode, tako i druge kontrole imaju svoje metode.

Postoje tisuće različitih metoda pa nema smisla ih učiti na pamet niti popisivati. Uvijek imajte Google pri ruci kako bi brzo pronašli metodu koja vam treba.

31) Vlastite funkcije

Do sada smo vidjeli kako koristiti već postojeće funkcije, a sada ćemo vidjeti kako napraviti vlastite funkcije.

a. Izrada vlastitih funkcija koje vraćaju neku vrijednost

Kada kreiramo funkciju, započinjemo s ključnom riječi **Function**. Zatim navodimo ime funkcije uz eventualne parametre. Nakon toga slijedi ključna riječ **As** i tip podatka kojeg će funkcija vratiti.

Kraj funkcije obilježavamo s **End Function**. Kraj izvršavanja funkcije obilježavamo s naredbom **Return.**

```
Function prikaziImePrezime() As String
    Return "Saša Fajković"
End Function
```

Ovaj kod će kreirati funkciju **prikazImePrezime** koja ne prima niti jedan parametar, a vraća String tip podatka. Konkretno, ova funkcija će vratiti tekst „Saša Fajković“.

```
Function zbroj(ByVal prviBroj As Integer, ByVal drugiBroj As Integer)
    Dim rezultat = prviBroj + drugiBroj
    Return rezultat
End Function
```

Ovaj kod će stvoriti funkciju koja se zove **zbroj** i koja prima dva parametra. Prvi parametar je **prviBroj** koji je po tipu Integer. Drugi parametar je **drugiBroj** koji je po tipu Integer.

U tijelu funkcije smo stvorili varijablu **zbroj** koja je po tipu Integer u koju ćemo pohraniti vrijednost zbroja vrijednosti varijabli prviBroj i drugiBroj. Nakon toga dolazimo do naredbe **Return** koja označava prekid izvršavanja funkcije i vraća vrijednost koja je pohranjena u varijabli rezultat.

32) Rad s tekstualnim datotekama

U ovom poglavlju ćemo upoznati kako spremati informacije u tekstualnu datoteku i kako čitati iz tekstualne datoteke.

a. Pisanje u tekstualne datoteke

Za pisanje u tekstualne datoteke koristimo **StreamWriter**. On će nam omogućiti jednostavno pisanje u tekstualnu datoteku. Prvo ćemo morati napraviti varijablu koja je tipa StreamWriter. Nakon što kreiramo tu varijablu, moramo navesti putanju do datoteke te naziv i ekstenziju datoteke u koju upisujemo.

Napomena : ako navedemo putanju do datoteke koja ne postoji, naš program će na toj lokaciji kreirati tu datoteku.

Nakon što smo naveli putanju otvaramo poveznicu prema toj datoteci odnosno pripremamo datoteku za upis. Sljedeći korak će biti upisivanje u datoteku. Na kraju još samo trebamo završiti s upisom tako da zatvorimo poveznicu između aplikacije i datoteke.

```
Dim file As System.IO.StreamWriter
        file                                                    =
My.Computer.FileSystem.OpenTextFileWriter("C:\Users\Saša\Desktop\VB1.txt",
True)
        file.WriteLine("Moj prvi zapis iz aplikacije u TXT datoteku.")
        file.Close()
```

Primjetite u drugoj liniji koda gdje navodimo putanju do datoteke da imamo dva parametra. Prvi parametar je putanja do datoteke. Drugi parametar govori aplikaciji da li da podatke upiše na kraj datoteke ili da prebriše postojeće i napravi upis od početka datoteke.

True – piši na kraj datoteke (neće prebrisati postojeće podatke)

False – piši na početak datoteke (prebrisat će postojeće podatke).

b. Čitanje iz tekstualnih datoteka

Za čitanje iz tekstualnih datoteka koristimo **fileReader**. Slično kao kod upisivanja, prvo kreiramo varijablu koja je po tipu String, zatim navodimo adresu datoteke koju želimo čitati i na poslijetku obrađujemo ili prikazujemo podatke iz tekstualne datoteke.

```
        Dim fileReader As String
        fileReader                                              =
My.Computer.FileSystem.ReadAllText("C:\Users\Saša\Desktop\VB1.txt")
        MsgBox(fileReader)
```

U ovom primjeru prvo kreiramo varijablu imena fileReader koja je po tipu String te toj varijabli dodjeljujemo putanju do datoteke. Na kraju obrađujemo podatke. U ovom primjeru ćemo učitane podatke prikazati unutar MessageBoxa.

33) Popis slika