

1. Uvod	3
1.1. .NET platforma – osnove	4
2. SINTAKSA PROGRAMSKOG JEZIKA C#.....	6
2.1. Promenljive, konstante i tipovi podataka	6
2.2. Operatori i funkcije.....	7
2.2.1. Primarni operatori.....	7
2.2.2. UNARNI operatori	8
2.2.3. ARITMETIČKI operatori.....	8
2.2.4. operatori POMERANJA.....	8
2.2.5. Operatori relacija i LOGIČKI OPERATORI	8
2.2.6. Operatori dodeljivanja	8
3. RADNO OKRUŽENJE PROGRAMSKOG JEZIKA Visual studio .Net IDE – C#.....	10
3.1. Izbor radnog foldera	10
3.2. Radno okruženje.....	11
3.2.1. Radno okruženje ZA Windows Form Application.....	12
3.2.2. Sadržaj palete sa alatima (Toolbox)	13
4. POSTUPAK IZRADE WIN FORM APLIKACIJE.....	15
4.1. PRIMER 1 – formiranje jednostavne forme.....	15
4.1.1. prilagođavanje izgleda forme	15
4.1.2. Dodavanje LABELLE.....	17
4.1.3. RAD SA Textbox – OM.....	17
4.1.4. Komandna dugmad.....	18
4.2. Programiranje događaja.....	19
4.2.1. Programiranje događaja Form.ACTIVATED	19
4.2.2. Programiranje događaja Form. KEYDOWN / KeYPRESS	21
4.2.3. Programiranje događaja click objekta button	21
4.3. PRIMERI LISTE (<i>ListBox</i>) , PADAJUĆE LISTE (<i>COMBOBox</i>).....	22
4.3.1. PRIMER FORME sa LISTAMA	22
4.3.2. PRIMER NIZ i LISTE.....	25
4.3.3. PRIMER NIZ i padajuća lista.....	31
4.4. KLASE i WIN FORME	33
4.4.1. Klase u C#	34
4.4.2. PRIMER KlasA u C#	34
4.5. RAD SA FOLDERIMA I datotekaMA	41
4.5.1. Metode klase za rad sa folderima	41
4.5.2. Metode klase za rad sa DATOTEKAMA.....	42

4.5.3. RAD SA FOLDERIMA I PODFOLDERIMA	43
4.5.4. RAD SA FOLDERIMA I PODFOLDERIMA – TREEView	48
4.6. RAD SA DATOTEKAMA.....	51
4.6.1. Formiranje DATOTEKE. UČITAVANJE i ČITANJE PODATAKA	51
4.7. RAD SA VIŠE FORMI	56
4.7.1 Rad sa više formi	57
4.7.2. Rad sa više formi. MENU. LISTView objekat.....	61
4.8. RAD SA GRAFIČKIM KLASAMA.....	75
4.8.1. KREIRANJE OSNOVNIH GEOMETRISKIH OBJEKATA	78
4.8.2. FORMA ZA CRTANJE GEOMETRIJSKIH OBLIKA	81
5. Rad sa bazama podataka u C#	89
5.1. RAD SA LOKALNOM BAZOM PODATAKA.....	89
5.2. POVEZIVANJE NA SERVERE BAZE PODATAKA. ADO.NET okruženje.....	103
5.2.1. Objektni model ADO.NET	104
5.2.1. POVEZIVANJE SA ACCESS-ovom bazom podataka	105

Microsoft Visual C# se koristi za razvoj aplikacija namenjenih da rade pod operativnim sistemom Windows, odnosno za Internet aplikacije. Prilikom razvoja aplikacija, u C# , se upotrebljava se radni okvir Microsoft. NET. Jezik C# koristi najbolje osobine jezika C++ i Microsoft Visual Basic jezika, pri čemu se može reći da je , sa jedne strane vrlo moćan, a sa druge strane jednostavan programski jezik za učenje.

Jezik C# 2.0 pojavio se 2005. godine, i tada je uveden generički koncept, dok prva verzija C# se pojavila 2001. god. Verzija 3.0 je dostupana u okviru Microsoft Visual Studija 2008, i ona je uključila Language Integrated Query – LIN–Q. Razvijanjem Visual Studio 2008 C# postaje brži, to jest više korisnički orijentisan za korišćenje, što je omogućilo brže i efikasnije pravljenje aplikacija za različite namene, aplikacije za rad sa bazama podataka, kao i aplikacija za Internet.

Za C# se može reći da je poboljšana verzija programskog jezika C/C++, odnosno da je u potpunosti baziran na pomenutim programskim jezicima. C# je relativno nov programski jezik baziran na modernom i najnovijem konceptu kako da programeri napišu programe koji su pouzdani i efektivni.

Takodje, C# je u potpunosti napravljen i usaglašen za upotrebu na Internetu. Pod ovim se podrazumeva da omogućuje ne samo da se jedan program izvršava na jednom kompjuteru, već i izvršavanje programa, tako da se komunicira sa drugim kompjuterima na lokalnoj mreži računara, ili da se komunicira sa računarima preko Interneta.

Visual Studio 2010 je programsko okruženje koje sadrži niz vizuelnih alata, koji omogućuju da se prave brzo i efikasno C# projekti. Mogu se praviti i projekti, gde se kombinuju dva programska jezika.

Programski jezik C# se pojavio 2001. god. Kao što je već pomenuto, to je relativno noviji programski jezik, ali je tako koncipiran i kreiran da bude sličan sa programskim jezikom C/C++, tako da ga poznavaoi C/C++ mogu lako razumeti i naučiti. C# je korisnički orijentisan jezik, asocijativan je, jednostavan, moćan, i potpuno objektno – orijentisan.

Takođe, Visual C#, koristi vizuelne alate, *the visual development tools*. Vizuelni alati, u razvojnom okruženju Visual Studio, omogućuju da se formiraju (tehnikom povuci i pusti – drag and drop) web – stranice kao i Win forme za desktop aplikacije. Visual C#, u stvari, kombinuje efikasnost i produktivnost jezika Visual Basic sa stilom jezika C/C++.

Visual Studio je razvojno okruženje koji omogućava olakšani i brži razvoj složenih programa u jeziku C#. Ali u suštini Visual C# je jezik C#, uz dodatne vizuelne alate. Visual C# je specijalizovan za pravljenje takozvanih "formi", gde se pod "formom" podrazumeva neki "prozor" na ekranu, na kome se prikazuju podaci, struktura podataka, u unapred definisanom i pogodno dizajniranom formatu. Na tu „formu“ mogu se postaviti razni objekti, kao što je Button – dugme, Label – nalepnica, *Textbox* – okvir za unos podataka, *ComboBox* – "padajuća" lista sa podacima, *ListBox* – lista sa podacima.

I u jeziku C#, kao i u C/C++, mogu se:

- definisati promenljive
- koristiti aritmetički operatori
- definisati metod – funkcije i koristiti argumenti metoda – funkcija.

Takođe *if* – iskazi i *while* – petlje su deo jezika C#. Isto tako, u C# se koriste izuzeci – *exceptions*, da bi se procesirale greške u programu. Konačno, C# je potpuno objektno – orijentisan jezik, gde se mogu koristiti svi principi objektno–orijentisanog programiranja, nasledje klasa, sakrivanje podataka, konstruktor – metode.

Jedna od najvažnijih promena u odnosu na C++ je u tome da se zahteva fajl sa zaglavlja. Novi pristup je dobijen

korišćenjem, takozvanog, *foldi* editora, koji može da "sklopi i rasklopi" kod metoda, odnosno da ga prikaže kao deklaraciju ili kao definiciju metoda.

.NET runtime, na kome se izvršava C#, sadrži upravljanje memorijom upotrebom objekata *garbage collector*. Zbog toga je korišćenje klasičnih pointera u C# manje važno nego u C++ . Klasični pointeri mogu se koristiti u C#, ali samo gde je kod označen kao *unsafe* i to pre svega ima smisla kada su performanse najvažnije.

Postoje još neke izmene u odnosu na C++ . Na primer:

- odsustvo globalnog prostora, sve je u klasama;
- sve je izvedeno iz praklase Object. Ovo važi i za vrednosne i za referentne tipove. To je preduslov da funkcioniše garbage collector;
- nema višestrukog nasleđivanja klasa, samo interfejsa;
- sigurnost tipova: garantuje se da će u svakoj promenljivoj biti vrednost tipa za koji je promenljiva deklarirana;
- pravilo izričite dodele promenljivoj pre korišćenja, u skladu sa sigurnošću tipova (obavezna inicijalizacija).

Kao što je već sspomenuto, u programskom jeziku C# nema ograničenja u pogledu toga kakve sve aplikacije mogu formirati. Najčešći tipovi aplikacija su:

- Windows aplikacije – aplikacije tipa Microsoft Office koje imaju izgled Windowsa i odlično se slažu sa njim. Ovo je uprošćeno korišćenjem modula *Windows form* unutar *.NET* okruženja, koji u stvari čini biblioteka upravljačkih objekata (kao što su dugmad, palete alatki, meniji i tako dalje) koje se koriste pri izradi Windows korisničkog interfejsa (UI).
- Web aplikacije – Web strane koje se mogu videti kroz bilo koji web pregledač (čitač). U okviru *.NET* okruženja, postoji sistem dinamičkog generisanja Web sadržaja, dozvoljavajući prilagođavanja, obezbeđujući sigurnost i slično – *ASP.NET (Active Server Pages .NET)*. Uz pomoć, programskog jezika C#, mogu se kreirati *ASP.NET* aplikacije korišćenjem Web formi.
- Web servisi – za kreiranje raznovrsnih distribuiranih aplikacija. Formiranjem ovih aplikacija, a pomoću Web servisa, preko Interneta se mogu razmenjivati bilo koje vrste podataka, uz primenu veoma jednostavne sintakse, ne vodeći računa o tome u kom sistemu je aplikacija postavljena, niti u kom je programskom jeziku napisana.

Naravno, u C# je omogućen i pristup bazama podataka, bez oobzira o kojoj je, od gore pobrojanih, aplikacija reč. U tu svrhu se koristi – ADO.NET (*Active Data Objects .NET*).

1.1. *.NET* PLATFORMA – OSNOVE

Microsoft proizvod *.NET* (dot-net) predstavlja za softversku tehnologiju budućnosti. Svedoci smo, da u današnje vreme, svaka aplikacija i programski jezik, razvijeni od strane Microsoft-a, su u *.NET* okruženju. U samom imenu ove tehnologije se krije i namena aplikacija razvijenih uz pomoć nje, a to je – dostupnost u svakom trenutku na svakom mestu. Zbog prethodnog rečenog, ova platforma predstavlja dugoročni i strategijski plan razvoja ne samo u Microsoftu, tako da se može zaključiti da *.NET* ima realne osnove da postane osnovna platforma za razvoj modernih aplikacija. Radni okvir *.NET* razvijen je sa ciljem da obezbedi okruženje za razvoj svih modernih aplikacija na Windows operativnom sistemu.

Osnovna, ako ne i glavna, osobina ove tehnologije je njena orijentacija ka distribuiranim aplikacijama preko Interneta. Ovo sa sobom ima i sledeće prednosti:

- distribuirane aplikacije su više objektno orijentisane;
- ubrzava se stvaranje kolekcije specifičnog koda na jednom mestu, suprotno stvaranju redundantnih (nepotrebnih ili obimnih) kopija na mnogo mesta;
- raspoloživost aplikacija na različitim uređajima preko interfejsa;
- upravljanjem pristupu, u realnom vremenu (*real-time*), ka distribuiranim čvorovima (delovi jedne softverske celine) moguće je lakše kontrolisanje rada takvih aplikacija – *services provided*;

- dostupna biblioteka klasa omogućava dobar dizajn, dosledno i dobro definisanje osnovnih tipova;
- jezička nezavisnost – ovo je omogućeno postojanjem međujezika – IL (*Intermediate Language*, ili *MSIL*), kod napisan na bilo kom jeziku, koji ima podršku za *.NET*, prevodi se na kod razumljiv tom međujeziku;
- podrška za *Web (XML)* servise – *.NET* ima razvijene alate za jednostavno pisanje XML servisa;
- poboljšan pristup dinamičkim *Web* stranicama baziran na *ASP.NET* tehnologiji;
- efikasan pristup podacima preko *ADO.NET* klasa;
- *.NET* postavlja i novi pristup za zajedničko korišćenje koda – suprotno tradicionalnim *dll* bibliotekama, uveden je koncept sklopova (*assembly*);

Za rad ovakvih aplikacija, podrazumeva se da postoji *.NET Framework*. U okviru *.NET Framework* ugrađen je *CLR (Common Language Runtime)* kao i kolekcije klasa ovog okruženja, među kojima treba istaknuti *ASP.NET* (nova generacija *Active Server Pages* tehnologije) i *WinForms* – za razvoj desktop aplikacija.

CLR izvršava kod koji je kompajliran na *.NET* platformi. *CLR* specifikacija je otvorena za sve platforme, uključujući i ne –Windows aplikacije. Takođe, svi ostali programski jezici se mogu koristiti za rad sa *.NET* klasama. Ove karakteristike obezbeđuju pravljenje distribuiranih aplikacija čije celine mogu da rade na nezavisnim platformama i čak pisane različitim programskim jezicima.

Svi jezici koji su obuhvaćeni novim paketom *Microsoft VisualStudio.NET* imaju podršku za *.NET* klase. Kako svi oni imaju “osnovu *CLR* to ih čini ravnopravnim više nego ikada. Međutim, C# je posebno napravljen programski jezik upravo za *.NET*.

2. SINTAKSA PROGRAMSKOG JEZIKA C#

Sintaksa programskog jezika podrazumeva skup pravila za pisanje programskih instrukcija. Sama sintaksa, *C#*, je veoma slična programskim jezicima *C/C++*.

Osnova pravila sintakse programskog jezika *C#*:

- programska linija – naredba je iskaz kojim se zadaje komanda računaru da izvrši neku operaciju,
- kraj svake programske linije se završava simbolom tačka – zarez (;),
- naredbe se mogu pisati jedna za drugom u istom redu, ali najčešće se pišu u posebnim redovima,
- naredbe se pišu u programski blokovima koji su oivičeni vitičastim zagradama {},
- blokovi određuju oblast vidljivosti promenljivih, odnosno oblast dejstva nekog koda,
- moguće je da postoji više ugnježenih blokova,
- programski kod *C#* jezika razlikuje velika i mala slova – *case sensitive*,
- komentari se koriste da bi se objasnio neki deo programskog koda:
// za red ili
/* */ za blok.

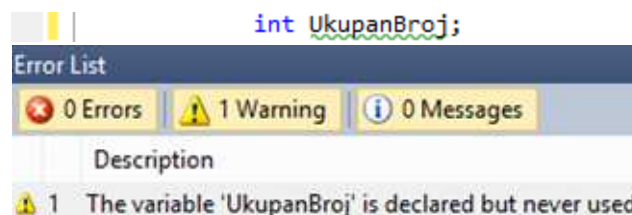
2.1. PROMENLJIVE, KONSTANTE I TIPOVI PODATAKA

Promenljive, kao i u *C/C++*, predstavljaju simboličko ime kojem će biti pridružena neka vrednost. Vrednost tih promenljivih, tokom izvršavanja programa, se može menjati. Same promenljive se definišu **imenom, memorijskom lokacijom, tipom podatka i vrednošću**.

Prilikom definisanja promenljive razlikuje se:

- deklaracija promenljive
- inicijalizacija promenljive – dodeljivanje početne vrednosti promenljivoj.
- definicija promenljive – uključuje deklaraciju i inicijalizaciju.

Deklaracija promenljive podrazumeva određivanje tipa i imena promenljive, na primer *int UkupanBroj*;, gde *UkupanBroj* predstavlja celobrojni tip podatka, odnosno u toj promenljivoj se čuva vrednost tipa celi broj – *integer*. Inicijalizacija podrazumeva da se deklarisanom promenljivoj dodeli početna vrednost, na primer *UkupanBroj = 0*;. Prednost *C#*, u odnosu na *C/C++*, je ta što nije obavezna inicijalizacija promenljive. Ukoliko se u programskom kodu pojavi promenljiva koja nije inicijalizovana, *C#* je detektuje, za vreme programiranja i tokom prevođenja i šalje poruku o nekorišćenim promenljivima, slika 1.



Slika 1. Poruka da promenljiva nije inicijalizovana, odnosno da nije upotrebljena

C# razlikuje tri tipa promenljivih, i to:

- *value type* – ime promenljive ukazuje na lokaciju gde se nalaze podaci, prilikom dodele vrednosti podaci se dodeljuju direktno;
- *reference variable* – ime promenljive ukazuje na lokaciju koja sadrži pokazivač koji pak pokazuje na lokaciju gde se nalaze podaci nekog složenog tipa – nizovi, matrice, strukture, ...;
- *pointer* – pokazivači – promenljiva sadrži pokazivač.

Tipovi podataka, koji se javljaju u *C#*, su u suštini isti kao i u *C/C++* i mogu biti:

- prosti – skalarni:
 - o pokazivač,
 - o bit, bajt,
 - o Boolean,
 - o brojevi (int, float, double),
 - o string, text;
- složeni – vektori – tipovi podataka koji predstavljaju uređene kolekcije prostih podataka, koje programeri definišu kao tipove podataka:
 - o niz,
 - o slog,
 - o skup,
 - o klasa,
 - o objekat,
 - o interfejs,
 - o stek,
 - o red,
 - o lista.

Kao i u *C/C++* postoje pravila za dodeljivanje naziva promenljivih, tako da naziv promenljive mora da počne sa slovom ili sa "_", odnosno sa "@", dok ostali simboli u nazivu mogu, pored slova i "_", mogu biti i brojevi. Naravno, za nazive promenljivih ne mogu se koristiti takozvane službene reči tipa *using*, *bool* i slično.

U prethodnom tekstu je pomenut tip promenljive *reference variable*, i u ovaj tip promenljivih spadaju:

- standardni tipovi:
 - o **Object** – bazna klasa za sve tipove podataka (*System.Object* klasa),
 - o **Dynamic** – opšti tip podatka, koji se može konvertovati u konkretan,
 - o **String** – niz karaktera;
- korisnički definisani tipovi:
 - o **Class**,
 - o **Interface**,
 - o **Delegate**.

U ovaj tip promenljivih spadaju i niz, lista, tipizirana lista.

Ovde treba napomenuti da postoji razlika između tipa *string* i *String*. Kada se deklarise promenljiva tipa *string*, sa njom se radi kao i sa svakom drugom prostom promenljivom vrednosnog tipa. Međutim, kada se deklarise promenljiva tipa *String*, tada se, u stvari, inicijalizuje objekat klase *System.String* i sa tom promenljivom, u stvari objektom, se radi korišćenjem atributa (*properties*) i metoda (*events*).

2.2. OPERATORI I FUNKCIJE

Operatori i funkcije u potpunosti imaju isto značenje i primenu kao što je to u programskim jezicima *C/C++*. Iz tog razloga, u ovom delu, daje se samo pregled i neke karakteristike značenja, odnosno primene operatora i funkcija.

2.2.1. PRIMARNI OPERATORI

- Pozivanje funkcije: $f(x)$
- Pristup članu niza – indeksirane strukture, preko indeksa: $A[i]$
- **postfix increment**, vraća vrednost x , a nakon toga menja vrednost memorijske lokacije: $x++$
- **postfix decrement**, vraća vrednost x i nakon toga na lokaciji x umanjuje vrednost x za 1: $x--$
- Iniciranje objekta klase: *New*

- Provera tipa promenljive ili objekta: *Typeof()* – *System.Type*
- Postavljanje *default* vrednosti promenljive ili objekta tipa *T* i to *null* za promenljive tipa reference, nulu (**0**) za numeričke tipove, a za strukturne tipove vraća promenljivu sa elementima koji imaju *null* ili nulu (**0**): – *default(T)*
- Provera veličine u bajtovima za promenljivu ili objekat *A*: *Sizeof(A)*

2.2.2. UNARNI OPERATORI

- Numerička negacija: $-x$
- Logička negacija: $!x$
- Prefix increment, uvećava vrednost *x* za 1 i vraća tu novu vrednost $x + 1$
- Prefix decrement, umanjuje vrednost *x* za 1 i vraća tu novu vrednost $x - 1$
- Memorijska adresa promenljive *x* – *address of*: $\&x$.

2.2.3. ARITMETIČKI OPERATORI

- Sabiranje: $x + y$
- Oduzimanje: $x - y$
- Spajanje stringova: $string1 + string2$
- Množenje: $x * y$
- Deljenje: x / y
- Modul – ostatak deljenja dva cela broja: $int1 \% int2$

2.2.4. OPERATORI POMERANJA

- Pomeranje bitova levo i popunjavaje nulom sa desne strane: $x \ll y$.
- Pomeranje bitova desno – ako je levi operand *int* ili *long*, levi bitovi se popunjavaju bitom znaka, a ako su *uint* ili *ulong*, levi bitovi se popunjavaju nulom (**0**): $x \gg y$.

2.2.5. OPERATORI RELACIJA I LOGIČKI OPERATORI

Operatori poredjenja:

- Manje od ($<$): $x < y$, izraz vraća *true* (tačno) ako je *x* manje od *y*.
- Veće od ($>$): $x > y$, izraz vraća *true* (tačno) ako je *x* veće od *y*.
- Manje ili jednako ($<=$): $x <= y$, izraz vraća *true* (tačno) ako je *x* manje ili jednako od *y*.
- Veće ili jednako ($>=$): $x >= y$, izraz vraća *true* (tačno) ako je *x* veće ili jednako od *y*.
- Jednakost ($==$): $x == y$, izraz vraća *true* (tačno) ako je *x* jednako *y*.
- Različito ($!=$): $x != y$, izraz vraća *true* (tačno) ako je *x* različito od *y*.

Logički operatori:

- Logičko I (*AND*): $x \& y$ – vraća vrednost *true* (tačno) ako su i *x* i *y* *true*, u suprotnom je *false* (pogrešno).
- Logičko ILI (*OR*): $x | y$ – vraća vrednost *true* (tačno) ako su ili *x* ili *y* *true*, vraća *false* (pogrešno) ako su i *x* i *y* *false*.
- Logičko XOR: $x \wedge y$ – vraća vrednost *true* (tačno) ako su *x* i *y* različite, vraća *false* (pogrešno) ako su i *x* i *y* iste.
- Uslovno logičko I (*AND*): $x \&\& y$ – ako je *x* operand *false* (pogrešno), *y* se ne proverava.
- Uslovno logičko ILI (*OR*): $x || y$ – ako je *x* operand *true* (tačno), *y* se ne proverava.

2.2.6. OPERATORI DODELJIVANJA

- Inkrementalno dodavanje vrednosti: $x += y$ – uvećava vrednost *x* za vrednost *y*.
- Inkrementalno umanjivanje vrednosti: $x -= y$ – umanjuje vrednost *x* za vrednost *y*.

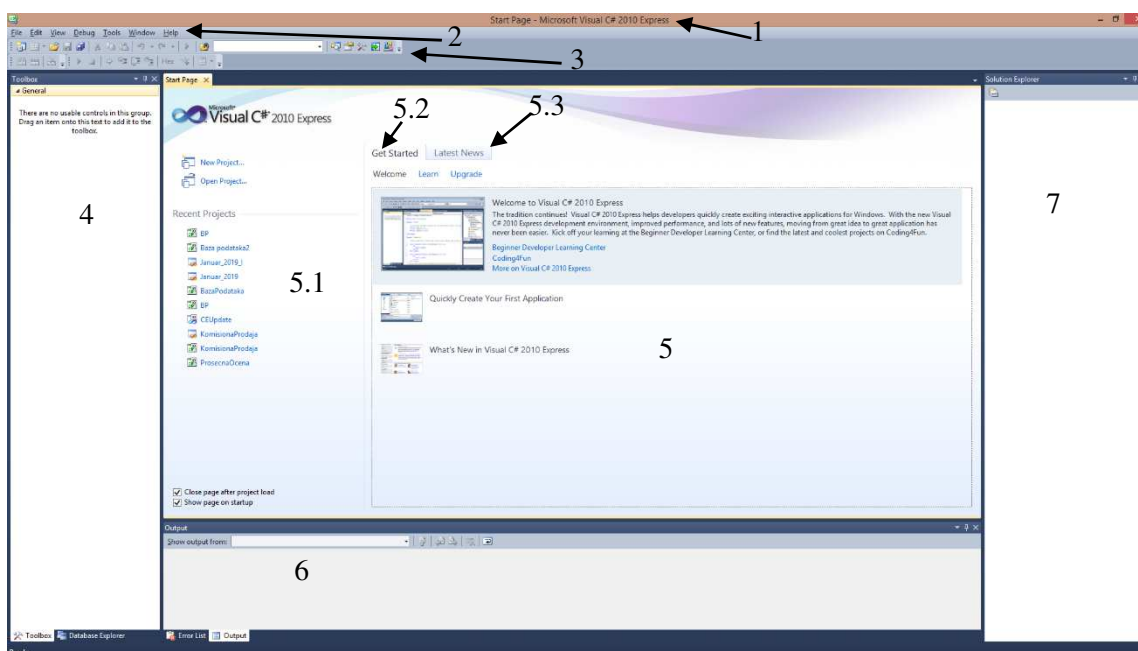
- Inkrementalno multipliciranje vrednosti: $x *= y$ – množi vrednost x sa y i smešta tu novu vrednost u x .
- Inkrementalno deljenje vrednosti: $x /= y$ – deli vrednost x sa y i smešta tu novu vrednost u x .
- Dodeljivanje ostatka deljenja: $x \% = y$ – određuje ostatak deljenja x i y smešta u x .

3. RADNO OKRUŽENJE PROGRAMSKOG JEZIKA Visual Studio .NET IDE – C#

Pre nego što se krene u objašnjavanje kreiranja aplikacija u Visual Studio .NET IDE – C#, daje se prikaz radnog okruženja u C#, (Висуал Студио 2010). Kao što je već pomenuto, C# je deo paketa MS Visual Studia, mada se može instalirati i samo C#.

Kada se pokrene C#, dobija se ekran kao što je prikazano na slici 2. Kao što se može uočiti, okruženje ovog programskog jezika, u potpunosti odgovara standadima koje je postavio Microsoft. Sa slike 2, uočavaju se, verovatno već poznate celine iz drugih MS aplikacija:

1. Traka sa nazivom projekta – Title bar;
2. Meni;
3. Paleta sa alati – Tool bar;
4. Alati – Toolbox;
5. Startna strana
 - 5.1. Rad sa projektima: Pokretanje novog projekta – New Project; Otvaranje postojećeg projekta – Open Project; Izbor aktuelnih projekata: Recent Project;
 - 5.2. Get Started – Učenje – Learn, Nadogradnje – Upgrade
 - 5.3. Poslednje vesti: Latest News
6. Prozori za praćenje toka programa – Output, odnosno za prikaz greški – Error List
7. Prozor sa elementima projekta




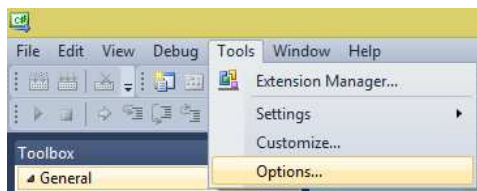
Slika 2. Početni ekran Visual Studio .NET IDE – C#

Naravno, pošto je ovo početni ekran, to jest nije izabran projekat, Prozor sa alatima (4), Prozor za praćenje toka programa i grešaka (6), kao i prozor sa elementima projekta (7) su prazni. Ovde je važno napomenuti da pri kreiranju novog projekta, potrebno izabrati radni folder gde će se taj projekat čuvati.

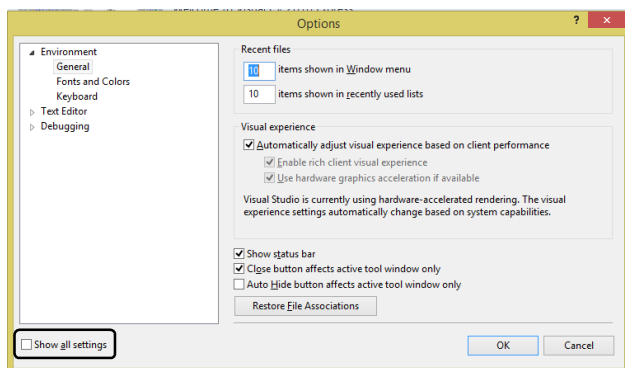
3.1. IZBOR RADNOG FOLDERA

Da bi se definisao folder u kome će se čuvati projekat potrebno je slediti sledeću proceduru.

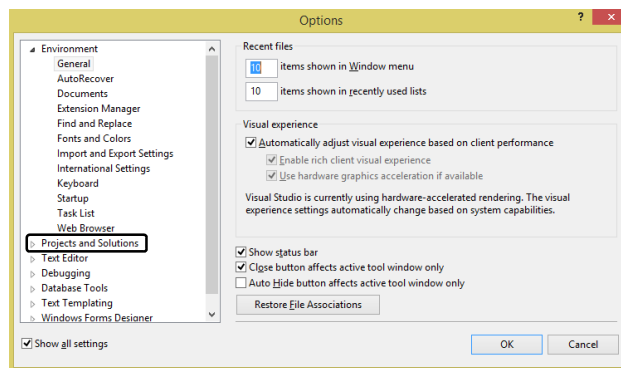
U meniju kliknuti na **Tools**, pa na **Options**, slika 3. Nakon ove akcije, pojaviće se prozor – **Options**, kao što je prikazano na slici 4. U donjem levom uglu ovog prozora, ukoliko nije čekirano Show all settings, slika 4.a), potrebno je kliknuti na Show all settings, i pojaviće se dodatne opcije, slika 4.b). Sada je potrebno kliknuti na Project and Solutions, slika 4.b), i dobiće se mogućnost da se izabere folder u kome će se čuvati projekat. Izbor foldera se dobija klikom na , slika 5.a).



Slika 3. Prvi korak u podešavanju radnog foldera

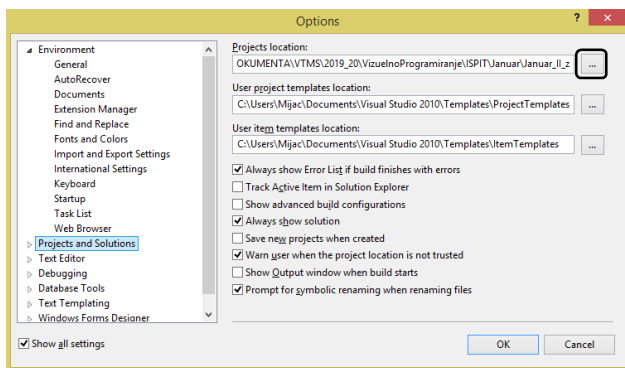


a)

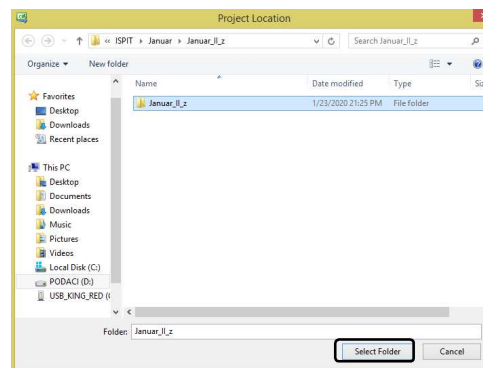


b)

Slika 4. Podešavanje radnog foldera



a)



b)

Slika 5. Izbor radnog foldera

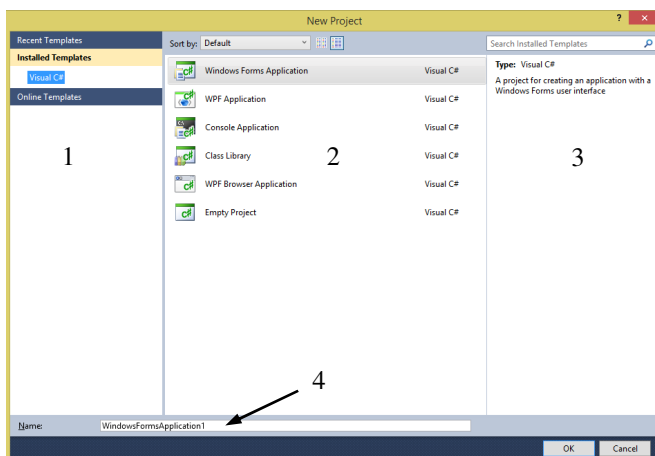
Iz prozora **Project Location**, slika 5.b), treba izabrati lokaciju foldera ili kreirati nov folder, i kliknuti na Select Folder čime se završava postupak izbora radnog foldera, odnosno foldera projekta.

3.2. RADNO OKRUŽENJE

Nakon pokretanja novog projekta, klikom na **New Project**, ili otvaranje postojećeg projekta, klikom na **Open Project**, slika 2., **C#** otvara forma za izbor tipa projekta, slika 6. Na ovoj formi se uočavaju 4 celine:

1. Šabloni – Template: urađeni obrasci i šabloni sa mogućnošću korisničkog prilagođavanja.
2. Tip aplikacije – u zavisnosti od izabranog šablona, ukoliko se radi u **MS Visual Studio** aplikaciji, u zavisnosti od izbora softvera (**C++**, **C#**, **Visual Basic**, **.NET**) pojavljuju se različiti šabloni, a pošto se ovde radi o **C#**, dati su šabloni karakteristični za **C#**:

- 2.1. **Windows Form Application** – za kreiranje aplikacija zasnovane na **Windows** formama.
- 2.2. **WPF Application** – za kreiranje Aplikacija na bazi win prezentacija **WPF** – **Windows Presentation Application**.
- 2.3. **Console Application** – za konzolne aplikacije.
- 2.4. **Class Library** – formiranje klase.
- 2.5. **WPF Browser Application** – za kreiranje Aplikacija na bazi web prezentacija.
- 2.6. **Empty Project** – prazan projekat – korisnik piše kompletan kod
3. Prikaz namene izabranog šablona
4. Unos naziva projekta

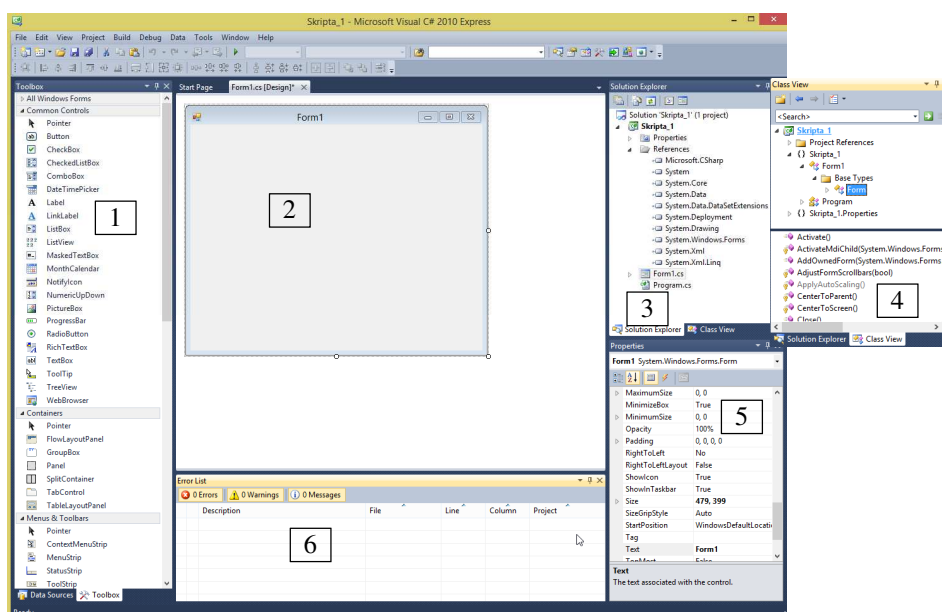


Slika 6. Prozor za izbor tipa projekta

Nako izbora tipa projekta upisuje se naziv aplikacije u *textbox* (4) i kliče se na OK, slika 6.

3.2.1. RADNO OKRUŽENJE ZA WINDOWS FORM APPLICATION

Bez obzira da li je pokrenut novi projekat ili je otvoren posto jeći, kada se radi **Windows Form** aplikacija, radno okruženje **C#** postaje kao što je prikazano na slici 7.



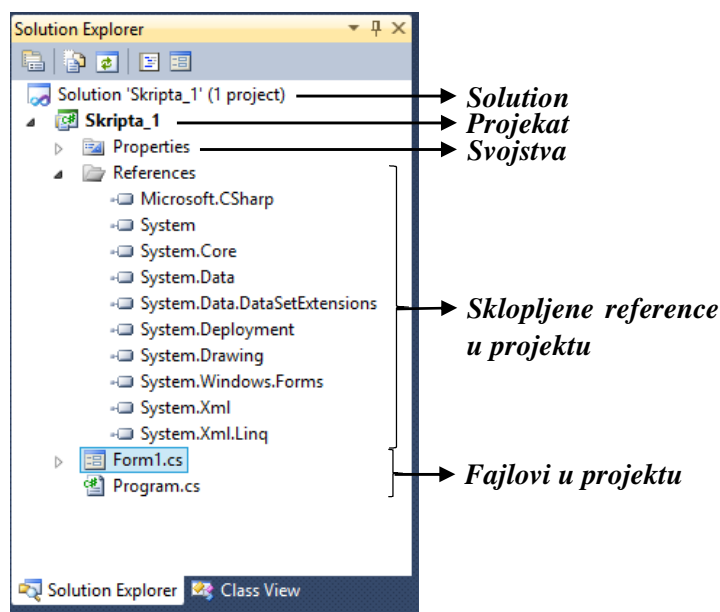
Slika 7. Radno okruženje

Sa slike 7., pored standardnog menija i palete sa standarnim komandama, mogu se uočiti 6 celina:

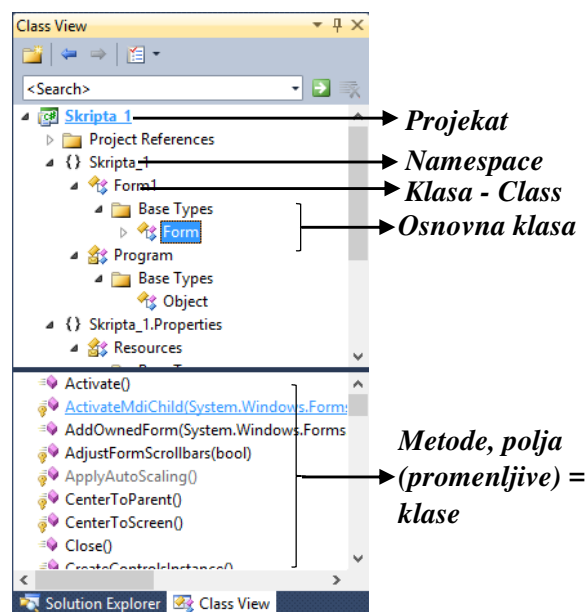
1. **Toolbox** – prozor sa alatka za vizuelno formiranje formi.

- Win forma – vizualna prezentacija klase *System.Windows.Forms.Form* na osnovu koje se kreira željeni izgled forme.
- Solution explorer* – prikazuje rešenje (*Solution*) i njegove projekte. Svaki projekat sadrži fajlove (*file*) i reference (*references*).
- Class View* – pregled kreiranih i/ili dodatih klasa. Daje hijerarhijski pregled metoda (*methods*), polja (*fields*), svojstava (*properties*) i osnovnih klasa za svaki objekat u projektu.
- Properties window* – prozor sa svojstvima. U ovom prozoru se daje mogućnost za pregled i prilagođavanje svojstava bilo kog objekta u projektu, odnosno za definisanje (*programiranje*) događaja (*events*) karakterističnih za izabrani objekat.
- Prozor za pregled grešaka (*errors*), upozorenja (*warnings*) i poruka (*messages*) koje šalje kompajler. U toku pisanja programskog koda, kompajler šalje poruku ukoliko se napravi greška u pisanju koda.

Na slikama 8 i 9 daje se značenje sadržaja prozora (3) i (4) sa slike 7.



Slika 8. Sadržaj *Solution Explorer*-a



Slika 9. Sadržaj *Class View*-a

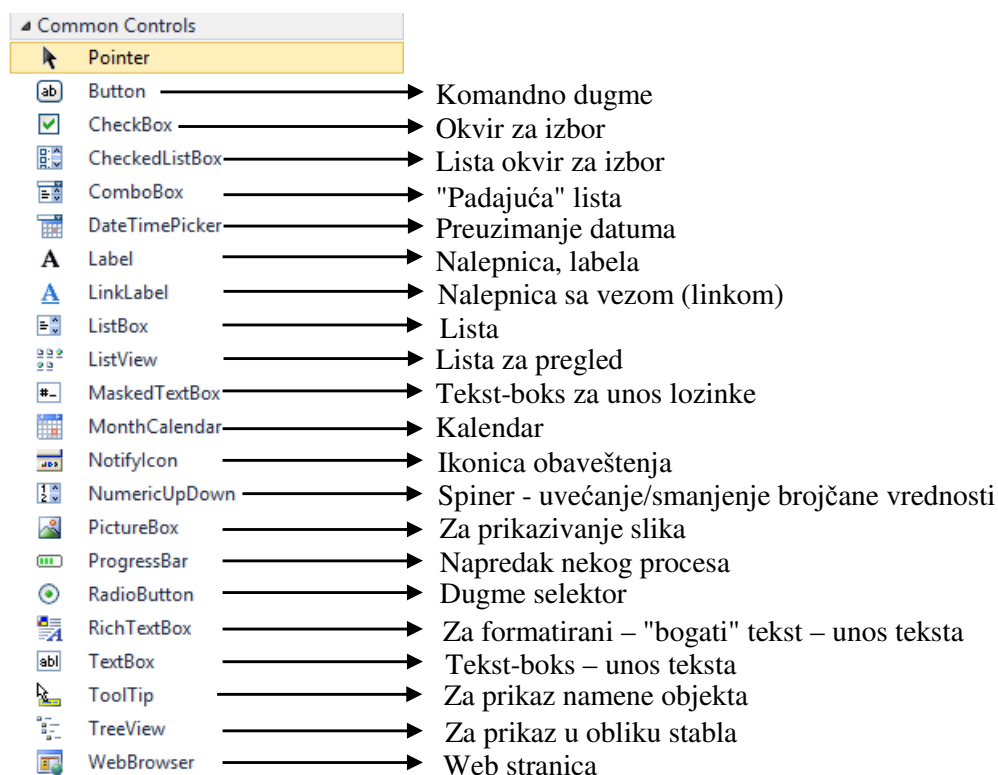
3.2.2. SADRŽAJ PALETE SA ALATIMA (TOOLBOX)

Toolbox, odnosno prozor sa alatima za formiranje objekata na formi, sastoji se od nekoliko celina:

- All Windows Forms – sve alatke na jednom mestu.
- Common Controls – univerzalne (uobičajene) alatke, slika 10.
- Containers – alatke za grupisane objekte.
- Menus & Toolbars – alatke za formiranje menija i paleta sa alatima.
- Data – alatke za formiranje objekata za rad sa podacima.
- Components – alatke za formiranje komponenti.
- Printing – alatke za formiranje objekata vezanih za štampanje.
- Dialogs – alatke za formiranje objekata za poruke – dijaloge.
- Components – alatke za formiranje komponenti.
- WPF Interoperability

U delu **Toolbox** – a, nalaze se alatke koje se najčešće koriste za kreiranje formi, **Common Controls**, slika 10. Ostale alatke se mogu naći u gore navedenim celinama. Tehnika rada je veoma jednostavna, iz palete alata se izabere objekat, klikne se na željeno mesto u formi i ne puštajući levi taster miša, razvuče se približno do željenih dimenzija. Naravno,

poziciju, dimenzije, kao i izgled željenog objekta mogu se naknadno menjati, odnosno prilagođavati u prozoru *Properties*, slika 5.



Slika 10. Alatke za kreiranje objekata u grupi *Common Controls*

U nastavku, ove skripte, daje se način rada i formiranja *Win Form* aplikacija u *C#*.

4. POSTUPAK IZRADE WIN FORM APLIKACIJE

U ovom poglavlju dat je postupak modeliranja forme, prilagođavanje izgleda i programiranje događaja. Primeri koji su dati idu od jednostavnijih ka složenijima i tehnike modeliranja formi se neće ponavljati, već će se pozivati na primer gde je ta tehnika izvedena.

4.1. PRIMER 1 – FORMIRANJE JEDNOSTAVNE FORME

U ovom primeru odradiće se :

4.1.1. Prilagođavanje forme – *Properties*

4.1.2. Dodavanje Labele – **A** Label i njeno prilagođavanje (*properties*)

4.1.3. Dodavanje dva Tekst–boksa – **abl** TextBox i njihovo prilagođavanje (*properties*)

4.1.4. Dodavanje komandnog dugmeta – **ab** Button i njegovo prilagođavanje (*properties*)

4.1.5. Programiranje događaja – *events*

4.1.5.1. Activated – događaj pri aktiviranju forme

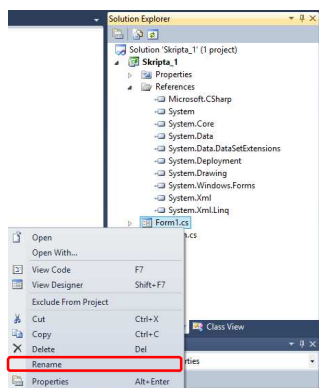
4.1.5.2. KeyDown / KeyPress – događaj (forma) čitanja pritisnutog tastera na tastaturi ili klik miša

4.1.5.3. Click – programiranje klika na dugme (*button – click*)

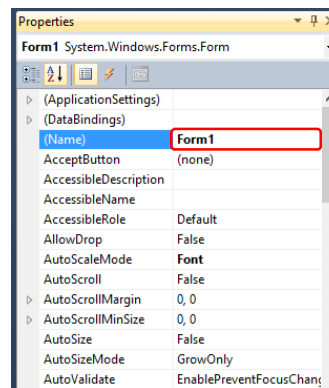
4.1.1. PRILAGOĐAVANJE IZGLEDA FORME

Kada se kreira nov projekat, pojaviće se radno okruženje sa praznom formom, kao što je dato na slici 7. Da bi se prilagodio izgled forme (ili bilo kog drugog objekta na formi) potrebno je da se ona selektuje. Znači, potrebno je kliknuti, levim tasterom miša, bilo gde u *formu*.

Poželjno je, ali ne i neophodno, da se promeni ime forme. Za to postoje dva načina. Prvi je da se u *Solution Exploreru*, klikne desnim tasterom na *Form1.cs*, slika 11., da se izabere *Rename* i unese novo ime za formu. Drugi način je da se u prozoru *Properties*, slika 12., u svojstvu (*Name*) unese novo ime za formu. Ovaj drugi način nije preporučljiv da se radi kada već postoje programirani događaji.



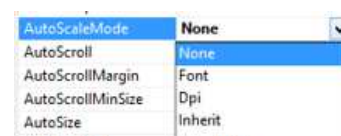
Slika 11. Promena imena forme u *Solution Explorer* – u



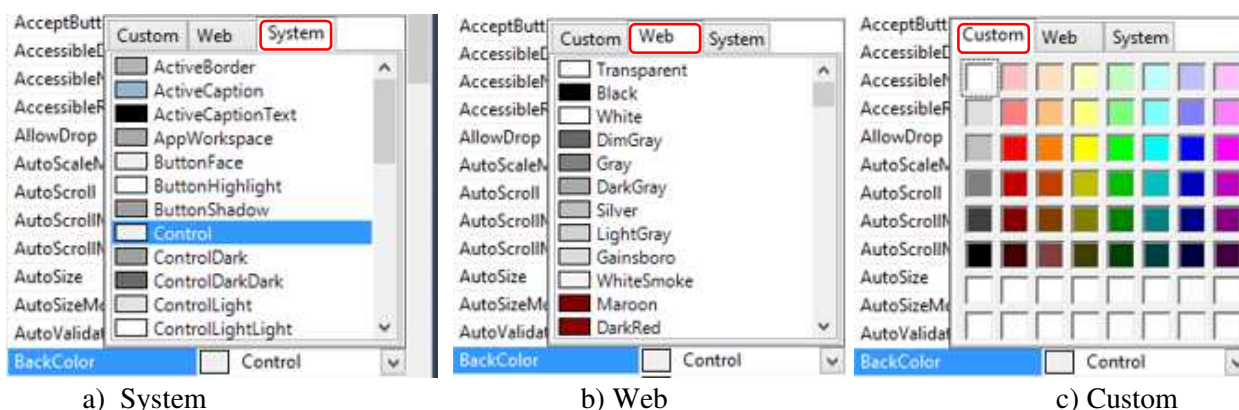
Slika 12. Promena imena forme u *Properties*

Formi će se dati ime *PrvaWinForma*. Kao što se vidi, ime objekta mora da bude samo jedna reč. Ovde je bitno napomenuti, da ako se menja naziv u *Solution Explorer* – u, ne sme se izbrisati ekstenzija *.cs*, kao i da se automatski svojstvu – (*Name*) dodeljuje isti naziv. Prilikom menjanja naziva u svojstvima (*Properties*) **ne menja se naziv forme u *Solution Exploreru*.**

Prvo svojstvo koje će se promeniti je *AutoScaleMode*. Ovim svojstvom se definiše da li će forma, proporcionalno, da menja veličinu u zavisnosti od postavljenog fonta slova za objekte na formi, odnosno od izabrane veličine fonta. Pošto ovo može da izazove negativne efekte – da se forma neželjeno poveća, ili smanji, ovo svojstvo treba postaviti na *none*.



Postavljanje boje forme se postiže izborom boje u svojstvu **BackColor**. Klikom na padajuću listu, slika 13, može se izabrati paleta boja i to **System** (a), **Web** (b) ili **Custom** (c).

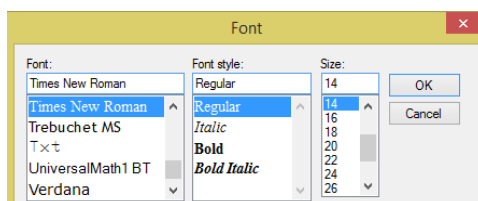


Slika 13. Paleta za izbor boje forme

Iz paleta **Web**, slika 13.b), izabrati **Navy**. Boje nisu poredene po azbučnom redu već po nijansama.

Kontrolna dugmad u formi (dugme za uklanjanje forme sa ekrana – pomeranje u taskbar – **minimized**, kao i za povećanje veličine forme – **maximized**, odnosno za gašenje forme – **close**) su postavljena po default – u, . Dugmad se može pojedinačno ukloniti tako što će se u svojstvima **MaximizeBox** i **MinimizeBox** postaviti vrednost **False**, a ako se želi da se uklone sva tri dugmeta, u svojstvu **ControlBox** takođe izabrati **False**.

Sledeće što treba definisati je oblik, veličina i boja slova – font, koji će važiti za svaki sledeći objekat, koji se dodaje u formu. Naravno, font se uvek, za izabrani objekat, može menjati. U svojstvu, **Font**, , klikom na tri tačkice, otvoriće se standardni prozor za izbor fonta, oblika i veličine slova, slika 14. Za font izabrati **Times New Roman, Regular, 14pt**.



Slika 14. Izbor fonta za formu

Boja slovima se daje izborom boje u svojstvu **FontColor**. Postupak izbora je potpuno isti kao i kod postavljanja boje pozadine, i za boju slova izabrati, iz paleta **Web**, **Navy**.

Da bi mogli da se programiraju događaji **KeyDown / KeyPress / KeyUp**, odnosno da se čitaju kodovi tastera sa tastature, posebno je da se svojstvo **KeyPreview** postavi na **True**.

Dimenzije forme se mogu postaviti korišćenjem tehnike pritisni i povuci, do željenih dimenzija, ili unosom dimenzija (širina i visina) u svojstvo **Size**. Za ovaj primer, dimenzije forme će biti **880,550**.

Standardno je da forme u Win aplikacijama budu pozicionirane u sredini ekrana. Ovo se postiže postavljanjem svojstva **StartPosition** na **CenterScreen**.

Ostalo je još da se u **Title bar** – u, odnosno u traci sa nazivom, definiše tekst koji će biti prikazan kada se startuje forma. Obično se za ovu vrednost daje naziv koji dovoljno asocira šta se u stvari radi u toj formi. Naziv se postavlja tako što se u svojstvu **Text** unese željeni naziv, za ovaj primer uneto je **Prva Win FORMA**.

U tabeli, T-1., dat je pregled svojstava koje su definisane za ovaj primer.

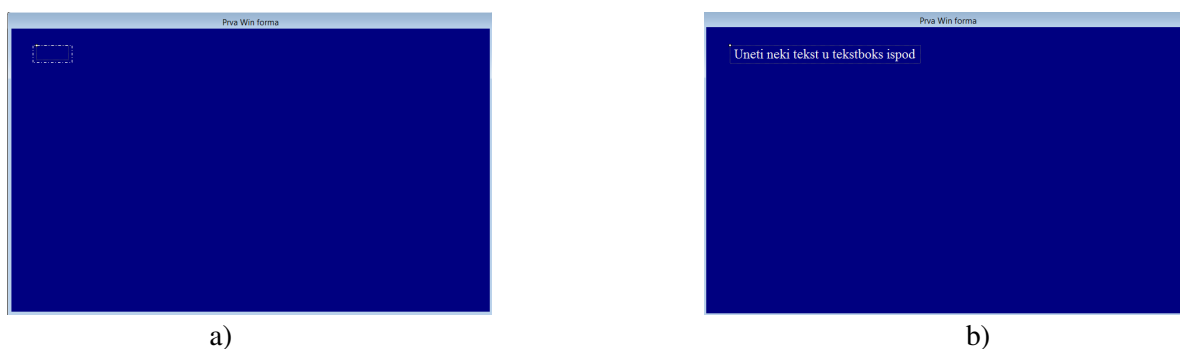
Tabela T – 1. Podešena svojstva fome

Svojstvo	Vrednost
(Name)	PrvaWinForma (podešeno promenom imena u Solution Explorer)
AutoScaleMode	None
BackColor	Navy
ControlBox	False
Font	Times New Roman, Regular, 14pt
ForeColor	Navy
KeyPreview	True
Size	880,550
StartPosition	CenterScreen
Text	Prva Win Forma

Svojstva za svaki objekat na formi se postavlja na potpuno isti način, tako da će se u narednim primerima davati samo tabela sa svojstvima koje treba postaviti.

4.1.2. DODAVANJE LABELLE

Prvi objekat koji se dodaje u formu, u ovom primeru biće labela, **A Label**. U paleti alata, slika 10, pronaći ikonicu za labele, kliknuti na nju i pa, klikom levim tasterom miša, pozicionirati je u formu, slika 15.a).



Slika 15. Postavljena labela u formi

Sa slike 15.a), se uočava da umesto da nešto piše, stoji samo pravougaonik. Ovo je iz razloga zao što je u prethodnom poglavlju za boju slova izabrana ista boja kao i boja pozadine, **Navy**, što znači da treba da se postavi nova boja, kako bi natpis labele bio vidljiv. U tabeli T – 2., dat je popis svojstava koje treba postaviti. Nakon postavljanja svih svojstava labele forma dobija izgled kao na slici 15.b).

Tabela T – 2. Podešena svojstva labele

Svojstvo	Vrednost
(Name)	Lbl1
Font	Times New Roman, Regular, 18pt
ForeColor	WhiteSmoke – Web paleta
Location	45,35
Text	Uneti neki tekst u tekst–boks ispod

I svaki sledeći objekat, koji se bude dodavao u formu, se dodaje na potpuno isti način.

4.1.3. RAD SA TEXTBOX – OM

Da bi se dodao textbox, potrebno je u paleti sa alatkama kliknuti na **Textbox** – **abl** **TextBox**. Namena ovog objekta je da se unose i prikazuju podaci, odnosno promenljive. Nakon izbora objekta i klika na proizvoljno mesto u formi postaviti vrednosti svojstava date u tabeli T – 3.

Tabela T – 3. Podešena svojstva tekst–boksa

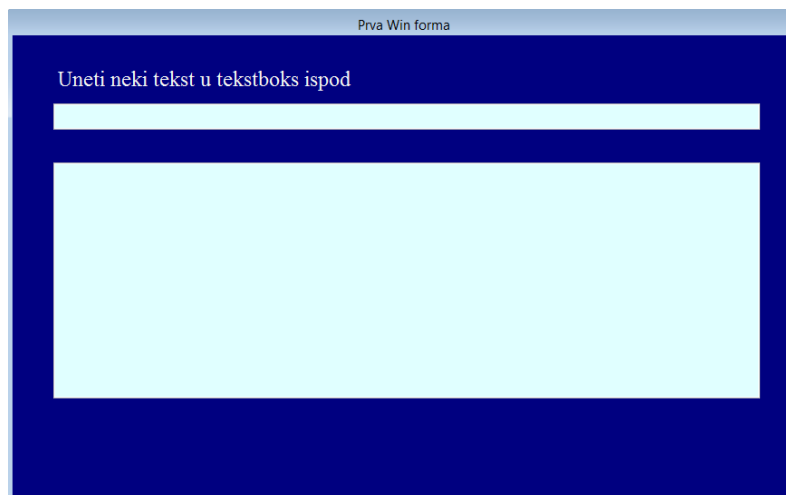
Svojstvo	Vrednost
(Name)	UnosTeksta
BackColor	LightCyan – Web paleta
Font	Times New Roman, Regular, 18pt
ForeColor	Navy – Web paleta
Location	45,75
Size	780, 29

Sada dodati, na isti način i drugi textbox i podesiti mu svojstva prema tabeli T – 4.

Tabela T – 4. Podešena svojstva drugog tekst–boksa

Svojstvo	Vrednost
(Name)	PrikazKoda
BackColor	LightCyan – Web paleta
Font	Times New Roman, Regular, 18pt
ForeColor	Navy – Web paleta
Location	45,140
Multiline	True
Size	780, 260

Nakon podešavanja, forma dobija izgled sa slike 16.



Slika 16. Forma nakon postavljanja i podešavanja textbox – ova

4.1.4. KOMANDNA DUGMAD


Komandna dugmad su objekti kojima se izvodi neka akcija. Podešavanje je u potpunosti isto kao i kod drugih objekata, a svojstva koja su podešena data su u tabeli T – 5. Komando dugme u paleti alata ima simbol:  Button .

Tabela T – 5. Podešena svojstva komadnog dugmeta

Svojstvo	Vrednost
(Name)	Izlaz
Font	Times New Roman, Bold, 16pt
Location	45,440
Size	780, 30
Text	Za izlazak iz programa

Nakon podešavanja i ovog objekta, dobija se izgled forme kao što je prikazano na slici 17.




Slika 17. Forma nakon postavljanja i podešavanja komandnog dugmeta

4.2. PROGRAMIRANJE DOGAĐAJA

Nakon što je forma kreirana, potrebno je sada i programirati željene događaje, kako za formu, tako i za objekte. Sintaksa pisanja programskog koda se, u suštini ne razlikuje od programskih jezika *C* i *C++*, tako da se, u ovom radnom materijalu, neće posebna pažnja posvećivati pisanju sintakse, ali tamo gde je potrebno, zbog specifičnosti programskog jezika *C#*, biće posebno objašnjena.

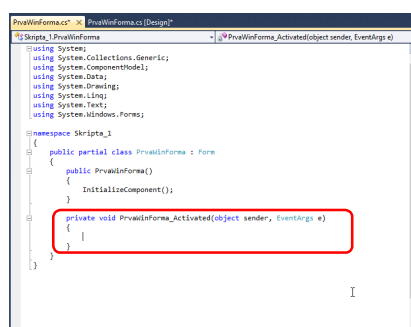
4.2.1. PROGRAMIRANJE DOGAĐAJA FORM.ACTIVATED

Za programiranje događaja (*event*), odnosno pisanja instrukcija za određeni događaj, potrebno je, u prozoru *Properties*, kliknuti na munju: , kako bi se prikazali svi događaji, za izabrani objekat, koji se mogu programirati.

Prvi programski kod koji će se napisati odnosi se na događaj kada se aktivira forma – *Activated*. Naime, ideja je, da kada se aktivira forma, da se u textbox – u: **PrikazKoda**, prikaže poruka:

U ovom textbox – u prikazaće se kod pritisnutog tastera sa tastature!

Da bi ovaj događaj programirao, potrebno je kliknuti na formu (NE NA OBJEKAT) i dva puta levim tasterom miša kliknuti na *Activated* u prozoru *Properties – Events*. Otvoriće se prozor za pisanje koda, slika 18.



Slika 18. Prozor za unos programskih linija

C# je odmah generisao proceduru, i ostavio prostor za unos programskog koda, PK – 1.

PK – 1. Generisan prostor za unos programskog koda za događaj *Activated*

```

1 private void PrvaWinForma_Activated(object sender, EventArgs e)
2 {
3
4 }

```

Sada je potrebno pozicionirati se između vitičastih zagrada, linija 3 iz PK – 1. i početi da se pišu programske linije. Pošto je zahtev, da se prilikom aktiviranja forme u **PrikazKoda**, prikaže poruka: "U ovom textbox – u prikazaće se kod pritisnutog tastera sa tastature!", to znači da se svojstvu **Text** ovog objekta dodeljuje zahtevana poruka, PK – 2.

PK – 2. Programskog koda za događaj **Activated**

```

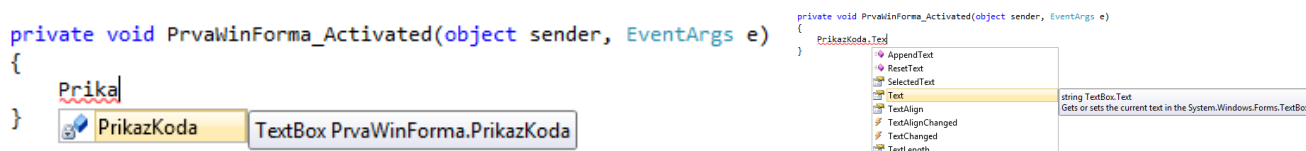
1 private void PrvaWinForma_Activated(object sender, EventArgs e)
2 {
3     PrikazKoda.Text = "U ovom tekst-boksu prikazaće se kod
4         pritisnutog tastera sa tastature!"
5 }

```

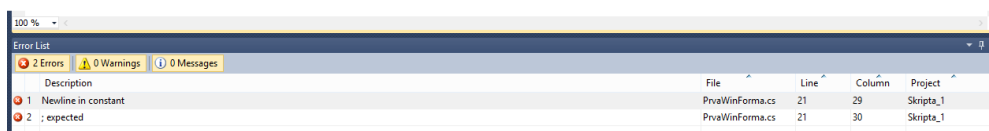
Napomena: U programskoj liniji 3, je sve u jednom redu, u originalnom programskom kodu.

Prilikom pisanja instrukcija, C# pretražuje bazu podataka objekata, klasa, funkcija i promenljivih koji se nalaze u formi koja se kreira, tako da iz ponuđene liste se može prihvatiti željeni objekat, slika 19. Ovo značajno pomaže u izbegavanju greški prilikom pisanja naziva objekata, klasa, funkcija, promenljivih i slično.

Takođe, C# prikazuje i greške sintakse koje se pojavljuju tokom pisanja koda, slika 20. Na ovaj način, programer može i pre kompajliranja programa da otkrije eventualnu grešku u sintaksi i da je ispravi. Konkretno na slici 20., jer nije završena programska linija do kraja, C# daje poruku da je se otpočelo sa pisanjem instrukcija, 1 linija i da nije zatvorena programska linija sa ";", 2 linija.

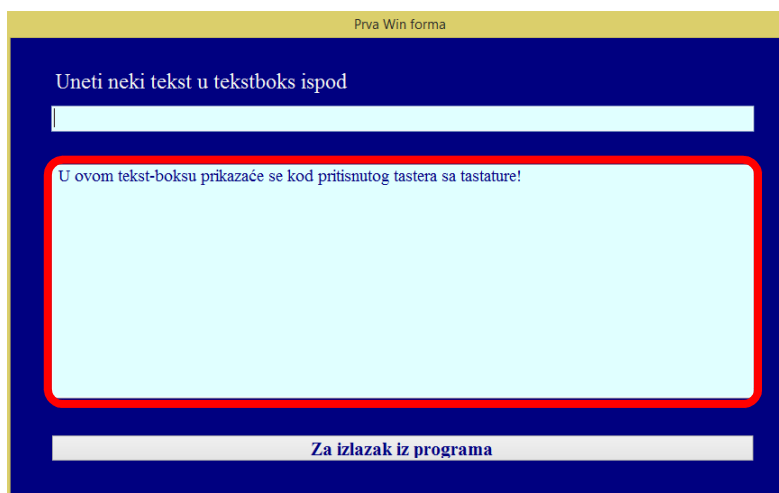


Slika 19. Case sensitive tehnika – pronalazjenja postojećih naziva objekata, klasa, funkcija i promenljivih



Slika 20. Upozorenje o greškama sintakse

Pokretanjem programa dobija se forma sa slike 21.



Slika 21. Forma sa izvršenom akcijom događaja **Activated**

U programskom kodu PK – 2., treba uočiti da je funkcija **PrvaWinForma_Activated**, sa dva argumenta: **object sender**, **EventArgs e**, i te argumente ne treba brisati, niti menjati njihovo ime, jer oni potiču iz bazne klase.

4.2.2. PROGRAMIRANJE DOGAĐAJA FORM. KEYDOWN / KEYPRESS

Procedura, za pisanje koda, je u potpunosti ista kao i prethodnom poglavlju. Ovde samo iz prozora *events*, treba izabrati događaj *KeyDown*, odnosno *KeyPress* i dva puta pritisnuti na njih kako bi se otvorio prozor za pisanje programskih linija. U PK – 3. i PK – 4., prikazane su instrukcije koje dodeljuju vrednost svojstvu *Text*, u tekst boks *PrikazKoda*, u zavisnosti koji taster je pritisnut na tastaturi. Naizgled ove dve procedure su slične. Različitost se ogleda u tome koje ugrađene funkcije se mogu u kom događaju primeniti. Tako, u *KeyDown* događaju, argument *e* vraća *KeyCode* – *ASCII* kod pritisnutog tastera, *KeyData* – podatak koji taster je pritisnut, recimo ako se pritisne broj 3 ispisaće se *D3*, ako se pritisne *Enter*, vratiće se *Enter* i *KeyValue* – taster koji je pritisnut. Kod procedure koja pokriva *KeyPress* događaj vraća samo karakter koji se nalazi u pritisnutom tasteru.

Događaj *KeyDown* se izvršava sve dok se taster, koji je pritisnut, ne otpusti, dok *KeyPress* proverava, da li je i koji je taster pritisnut. U proceduri *KeyDown* je dodato uslovno grananje, jer se želi, da kada se pritisne taster *Esc*, da se forma zatvori, odnosno da se prekine izvršavanje programa.

PK – 3. Programski kod za događaj *KeyDown*

```
1 private void PrvaWinForma_KeyDown(object sender, KeyEventArgs e)
2 {
3     if (e.KeyCode == Keys.Escape)
4         Application.Exit();
5     else
6         PrikazKoda.Text += "\r\n KOD " + e.KeyValue;
7 }
```

PK – 4. Programski kod za događaj *KeyPress*

```
1 private void PrvaWinForma_KeyPress(object sender, KeyPressEventArgs e)
2 {
3     PrikazKoda.Text += "\r\n Pritisnuti taster je '" + e.KeyChar+"'";
4 }
```

U programskoj liniji 6., u PK– 3. i 3 u PK – 4., *\r* i *\n* označavaju prelazak u nov red – potpuno isto značenje kao i kod *C/C++*. Između znaka navoda se prikazuje sve ono što je između napisano. U ovim linijama, prikazan je postupak dodeljivanja vrednosti u *textbox* *PrikazKoda*, drugim rečima, string koji se upisuje u *textbox* se dodeljuje svojstvu *Text*, *PrikazKoda.Text = string_koji_se_dodeljuje*. Naravno i preuzimanje vrednosti i upisivanje u promenljivu, sadržaja iz *textbox* a je preko svojstva *Text*, *string_promenljiva = PrikazKoda.Text*.

4.2.3. PROGRAMIRANJE DOGAĐAJA CLICK OBJEKTA BUTTON

Ostaje još da se programira događaj *click*, kada se klikne na komandno dugme *Izlaz*. Za razliku, od programiranja događaja na formi, kada se radi o ovom objektu dovoljno je kliknuti 2x levim tasterom miša na njega i otvoriće se prozor za pisanje koda. Kod za ovaj događaj je dat u PK – 5.

PK – 5. Programski kod za događaj *Click* objekta *button: Izlaz*

```
1 private void Izlaz_Click(object sender, EventArgs e)
2 {
3     Application.Exit();
4 }
```

U programskim kodovima PK – 3. i PK – 5., uočavaju se dve potpuno iste instrukcije, u linijama 4., odnosno 5. respektivno. Ova instrukcija poziva metod *Exit* iz klase *Application*, kojim se prekida izvršenje programa.

Ovime je završeno kreiranje ove jednostavne forme.