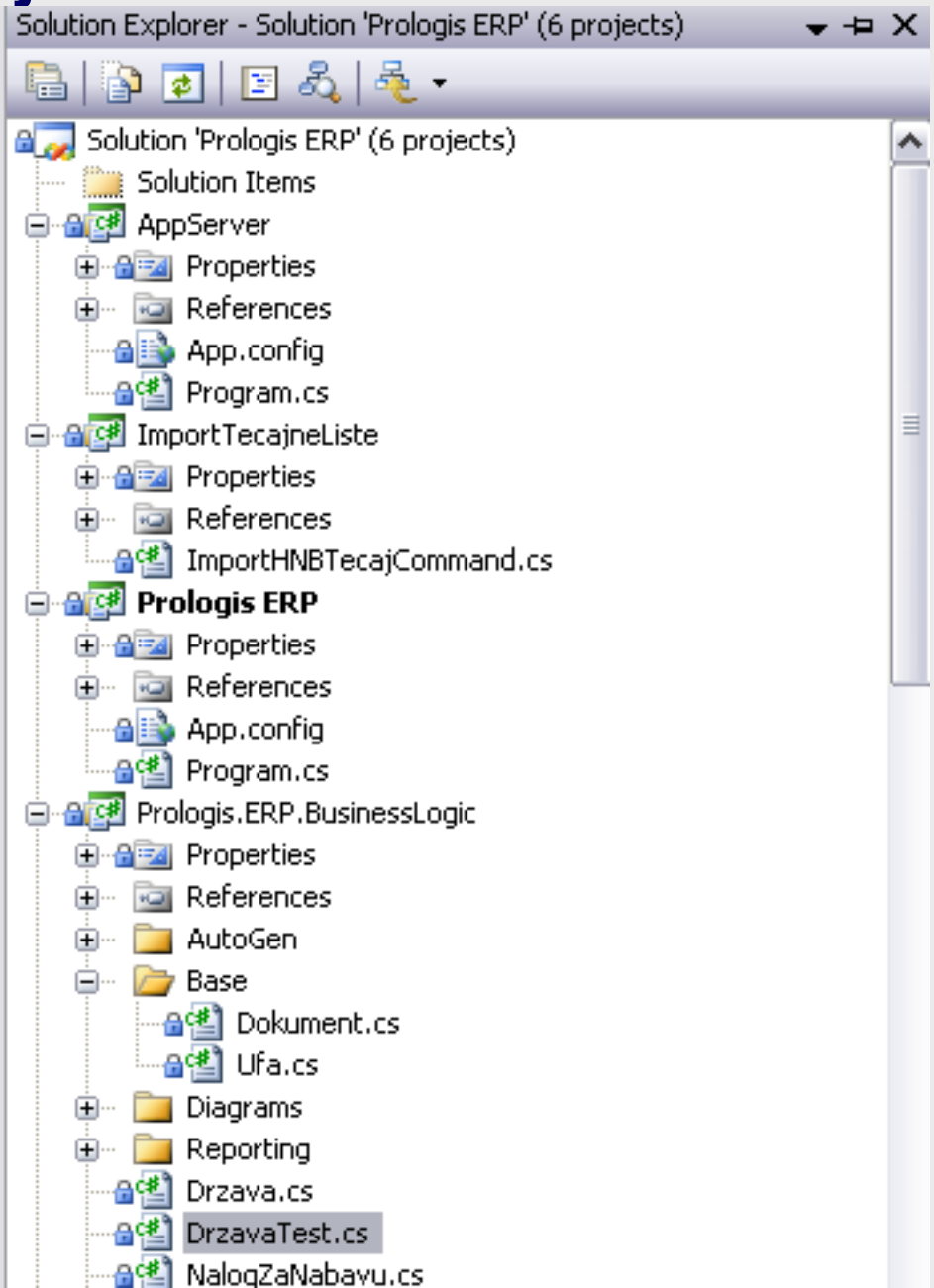
Standardi kodiranja - smjernice

2014/15.04

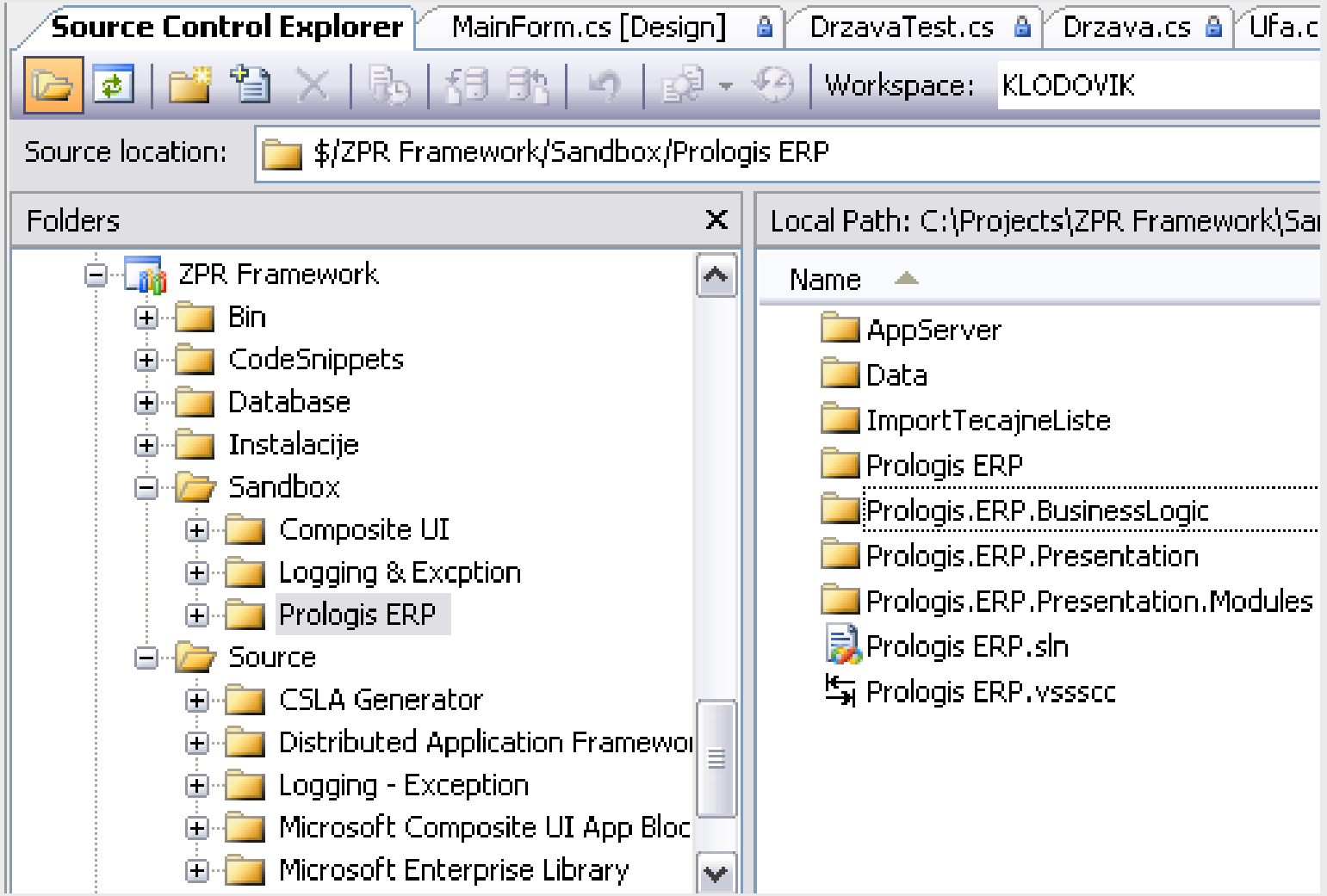
Organizacija datoteka

- **Izbjegavati preduge datoteke**
  - datoteke s više od 300-400 programskih redaka restrukturirati (refactor) uvođenjem pomoćnih (helper) razreda
  - najbolje je staviti po jedan razred u zasebnu datoteku, naziva jednakog nazivu razreda
- **Komponente imenovati sukladno hijerarhiji**
  - <Company>.<Component>.dll, npr. Prologis.ERP.BusinessLogic.dll
  - Menus.FileMenu.Close.Text



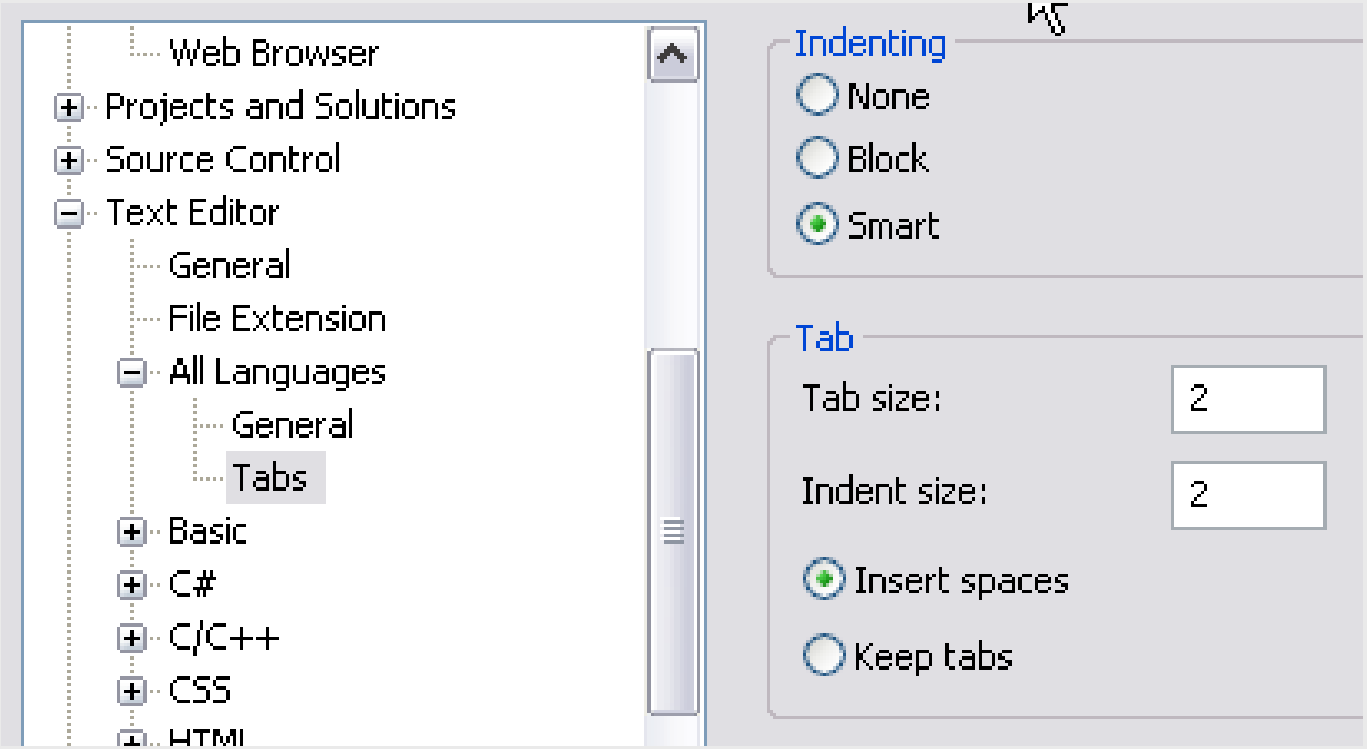
Organizacija datoteka

- **Organizacija datoteka treba slijediti organizaciju komponenti**



Formatiranje izvornog programskog koda

- različito označavanje elemenata jezika, kao što su rezervirane riječi, identifikatori, komentari, opcije prevoditelja (različita boja)
- izbjegavanje redaka koji duljinom prelaze širinu zaslona (80 znakova)
- pisanje najviše po jedne programske naredbe u retku
- podjela slijedova naredbi na odsječke koji su u cjelini vidljivi na zaslonu
- tzv. "uvlačenje" (*indentation*) kôda unutar programskih struktura



Formatiranje bloka naredbi

- Redak proreda za odvajanje logičkih grupa naredbi
- Vitice u zaseban redak poravnate s naredbom strukture
- Vitica se preporuča i kada blok nema ili ima samo jednu naredbu

```
  bool VratiPare(string name)
  {
    string fullMessage = "Vrati pare " + name;
    DateTime currTime = DateTime.Now;
    fullMessage += " u " + currTime.ToString();
    MessageBox.Show(fullMessage);

    if (showResult)
    {
      for (int i = 0; i < 10; i++)
      {
        // radi
      }
    }

    return true;
  }
}
```

Formatiranje naredbe ili izraza

- **Jedan razmak između operatora, ali ne uz zagrade**

- **Postupanje kada naredba ili izraz premašuju poželjnu širinu retka**
  - Prekid (skok u novi redak) na separatoru liste, tj. zarezu.
  - Prekid prije operatora
  - Bolje prelomiti izraz na višoj razini nego onaj na nižoj razini
  - Poravnanje narednog retka s početkom izraza u prethodnom retku

- **Primjeri**

```
longMethodCall(expr1, expr2,
               expr3, expr4, expr5);
```

```
var = a * b / (c - g + f)
    + 4 * z;
```

Standardizacija nazivlja

- **Notacije za obilježavanje programskih objekata**
  - *PascalCase* - početno slovo svake riječi u imenu je veliko slovo
    - npr. `BackColor`
    - koristi se kod imenovanja prostora imena, razreda, sučelja, pobrojanih tipova, postupaka i svojstava, te public atributa
    - Identifikator može započeti znakom @
  - *camelCase* – početno slovo prve riječi u imenu je malo slovo, početna slova ostalih riječi u imenu su velika slova
    - npr. `backColor, jedinstveniMaticniBroj`
    - koristi se kod zaštićenih atributa i lokalnih varijabli postupaka
  - Preporuke
    - Imena sučelja uobičajeno započinju slovom I
    - Koristiti imenice za imena razreda
    - Koristiti glagolske oblike za imena postupaka

Primjer naziva programskih elemenata

- ***PascalCase*** **za imena razreda i postupaka**
- ***camelCase*** **za lokalne varijable i argumente potprograma**

```
public class Ispit
{
 int totalCount = 0;

 void Prijavi(string jmbag)
 {
  ...
  string fullMessage = "Prijavljen " + jmbag;
  ...
 }
}
```

Smjernice za nazivlje

- **Nazivlje struktura podataka**
  - pridjeljivati nazive iz kojih se vidi na što se odnose
    - Primjerice: Osoba.SifraOsobe, Mjesto.SifraMjesta
    - Umjesto: Osoba.Sifra, Mjesto.Sifra, Artikl.Sifra
  - izbjegavati uporabu posebnih znakova koje sintaksa jezika/sustava ne dozvoljava pri tvorbi identifikatora
    - pr. operatori i znakovi za palatale našeg jezika, Dat-Rođ
  - izbjegavati prekratke nazive koji, osim u nečitkost, vode u nedosljednost već pri prvoj pojavi iste kratice za različiti pojam
    - npr. SifMje za Mjeru i Mjesto
  - izbjegavati preduge nazive, pr. UvoznaCarinskaDeklaracija.RedniBrojStavkeKalkulacije, zbog
    - smanjenja čitljivosti
    - učinkovitosti ručnog kodiranja (sintaksne pogreške izazvane tipkarskim)
    - mogućih ograničenja jezika (pr. duljina identifikatora do 18, 30... znakova)
  - izbjegavati nazive dobivene rutinskim spajanjem naziva entiteta i atributa jer mogu djelovati nezgrapno unutar upita, na primjer:
    - umjesto SELECT Posao.* FROM Posao WHERE Posao.posao_datum …
    - bolje je SELECT Posao.* FROM Posao WHERE Posao.DatPosla …
  - koristiti nazive koji se daju izgovoriti
    - pr. Nstvnk.SifNstvnk→ Nastav.SifNastav ili Nastavnik.SifNastavnika

Smjernice za nazivlje

- **Nazivlje programskih varijabli**
  - koristiti smislene nazive
    - izbjegavati "jednoslovčane" varijable, pr. i, j, k ili i, ii, iii, ili, x1, x2, x3
    - osim za indekse i dimenzije polja, pr. i, n
  - nazive odabirati u skladu sa značenjem sadržaja
    - pr. max za najveću vrijednost
    - pr. len za varijablu koja određuje duljinu
  - koristiti standardne prefikse/sufikse za srodne elemente/objekte
    - pr. frmOsoba ili fOsoba za zaslonsku masku
    - pr. repOsoba, rOsoba za izvješće …
  - dozvoljeno koristiti jednoznačne kratice općih pojmova kao što su
    - pr. broj, redni broj, šifra, kratica, oznaka, datum
    - → br, rbr, sif, krat, ozn, dat

Deklaracija varijabli

- **Lokalne varijable**
  - izbjegavati deklaraciju više varijabli u istom retku zbog komentiranja
  - inicijalizaciju provesti na mjestu deklaracije

```
int level; // razina rekurzije
int size; // dimenzije matrice

int a, b; //monolog ili recitacija ?
```

```
string name = mojObjekt.Ime;
int hours = vrijeme.Sati;
```

- **Članske varijable**
  - deklarirane na vrhu razreda
  - privatne
  - nad njima javna/zaštićena svojstva

```
class Osoba
{
  private string jmbg;
  public string JMBG {
    get { return jmbg; }
    set { jmbg = value; }
  }
}
```

Inicijalizacija varijabli

- **Inicijalizaciju provesti na što jednostavniji način**

```
// v1
bool pos;
if (val > 0)
{
pos = true;
}
else
{
pos = false;
}

// v2
bool pos = (val > 0) ? true : false;

// v3
bool pos;
pos = (val > 0);

// v4
bool pos = (val > 0);
```

Inicijalizacija varijabli

- **Inicijalizacija polja referenci**

```
public class Krumpir
{ }

public class PoljeKrumpira
{
  const int VelicinaPolja = 100;
  Krumpir[] polje = new Krumpir[VelicinaPolja];

  PoljeKrumpira()
  {
    for (int i = 0; i < polje.Length; ++i)
    {
        polje[i] = new Krumpir();
    }
  }
}
```

Metode

- **Metoda**
  - naziv metode mora upućivati na to što metoda radi
  - metode s više od 25 redaka preoblikovati podjelom u više metoda
  - izbjegavati metode s više od 5 argumenata – koristiti strukture

```
void SavePhoneNumber (string phoneNumber)
{
  // spremi fon
  ...
}
```

```
// sprema broj telefona
void SaveData (string phoneNumber)
{
  // spremi fon
  ...
}
```

Metoda mora obavljati samo jedan posao

```
// spremi adresu
SaveAddress (address);

// posalji mail obavijesti promjene
SendEmail (address, email);

void SaveAddress (string address)
{
 // spremi
 // ...
}

void SendEmail(string address,
               string email)
{
 // posalji
 // ...
}
```

```
// spremi i salji
// mail obavijesti
SaveAddress(address, email);
void SaveAddress(
  string address,
  string email)
{
  // 1: spremi
  // ...
  // 2: salji
  // ...
}
```

Komentari

- **Preporuke**
  - paziti da komentari budu ažurni, tj. da odgovaraju stvarnom stanju
  - ne pretjerivati u pisanju komentara
  - loš kôd bolje je iznova napisati, nego (bezuspješno) pojašnjavati
  - komentirati smisao naredbi (izbjegavati "prepričavanje")

- **Komentari bloka trebaju biti na istoj razini gdje i kôd**
- **Izbjegavati blok komentare i “staromodne” retkovne C komentare**

```
/*************************************************\
* windowsx.h -  Macro APIs, window message crackers,  *
*                          and control APIs           *
*               Version 3.10                          *
\*************************************************/

/* slijede neke f-je */
```

Primjer komentara

- Opis, argumenti, trag izmjena, retkovni / komentar bloka
- Oznake XML oblika za generiranje dokumentacije

```
/// <summary>
/// Execute a SqlCommand (that returns ...)
/// </summary>
/// <remarks>
/// e.g.:
///  int result = ExecuteNonQuery(connString, ...
/// </remarks>
/// <param name="connectionString">A valid string</param>
...
/// <returns>An int representing ...</returns>
/// <created></created>
/// <modified>K.Fertalj, 25.11.2007. 00:13<modified>
public static int ExecuteNonQuery(string connectionString,
                  CommandType commandType, string commandText)
{
  // Pass through the call ...
  return ExecuteNonQuery(connectionString,
           commandType, commandText, (SqlParameter[])null);
```

Konstante, skraćenice, tipovi podataka

- `const` koristiti samo za stvarno nepromjenjive vrijednosti
- Koristiti sva velika slova u nazivu varijable ukoliko predstavlja kraticu naziva dvije do tri riječi ili jest duljine do tri znaka
- Koristiti tipove podataka definirane programskim jezikom, a ne sistemske (definirane u imeniku System)
  - object, string, int (**ne** Object, String, Int32)

```
public class MojaMatka
{
  private const short BrojDanaTjedna = 7;
  public const double PI = 3.14159;
  public const float PDV = 0.22f;
}
```

Konstante

```
enum MailType
{
  Html,
  PlainText,
  Attachment
}
void SendMail(string message,
              MailType mailType)
{
  switch(mailType)
  {
   case MailType.Html:
    // radi html
    break;
   case MailType.PlainText:
    // radi txt
    break;
   case MailType.Attachment:
    // radi attach
    break;
   default: // uvijek default!
    // radi ostalo
    break;
  }
}
```

- **Izbjegavati izravnu upotrebu konstanti (hard kodiranje)**
  - koristiti pobrojane tipove, predefinirane konstante i datoteke s parametrima i resursima

```
void SendMail(string message,
              string mailType)
{
  switch (mailType)
  {
   case "Html":
    // radi html
    break;
   case "PlainText":
    // radi txt
    break;
   case "Attachment":
    // radi attach
    break;
   default:
    // radi ostalo
    break;
  }
}
```

Pobrojane vrijednosti

- **Izbjegavati eksplicitni navod vrijednosti**

```
// Dobar
public enum Barjak
{
  Crven, Bijeli, Plavi
}

// Manje dobar
public enum Barjak
{
  Crven = 1, Bijeli = 2, Plavi = 3
}
```

Logički izrazi

- **Poziv funkcija u logičkim izrazima**

```
bool ok = IsItSafe();
if (ok) // umjesto if (IsItSafe())
{
...
```

- **Usporedba s true/false**

```
while (condition == false) // loš
while (condition != true) // loš
while (!condition) // OK
```

- **Usporedba realnih brojeva**
  - izbjegavati operatore == i !=
  - osim kada je varijabla inicijalizirana na 0 (1, 2, 4, ...) pa se kasnije želi utvrditi da je došlo do promjene

Nazivi objekata u bazi podataka

tablica	jednina, bez prefiksa/sufiksa	Osoba, Racun
tablica	spojne kao RoditeljDijete	ArtiklRabat, PartnerKontakt
polje	ŠtojeEntitetu	SifraKupca, BrojStavki
polje	IdTablice za jednoznačni identifikator	IdEvidencije
polje	bool s prefiksom Ima, Je, Zast	ImaPovijest, JeZapisan, ZastZakljucan
polje	smisao broja, zbirna imenica	RadniSati, RadnoVrijeme
ključ	pk_Tablica	pk_Osoba, pk_RadnoMjesto
ključ	fk_Roditelj_Dijete, fk_Roditelj_Uloga	fk_Osoba_MjestoRod, fk_Osoba_IdMjestaRod
ključ	ix_Tablica_Polje	ix_AutorVrste_IdVrste
ključ	ux_Tablica_Polje	ux_Tecaj_OznValutaDatTec
check	ck_Tablica_Polje_KojiCheck	ck_Stavka_Kolicina_Poz, ck_Stavka_Kolicina_Max
default	df_Tablica_Polje	df_Racun_Rabat
pogled	vw_Pogled	vw_MojiProjekti
proc	ap_StoraOpcenita	ap_Help
proc	fn_FunkcijaOpcenita	fn_PronadjiProizvode
proc	ap_Tablica_CRUDQX # kombinacija	ap_UlaznaFaktura_C
trigger	tr_Tablica_IUD	tr_Artikl_I

Stupci tablica u bazi podataka

Polje	Kratica	Tip	Duljina
Sifra	Sif	int	4
Broj	Br	int	4
Redni broj	Rbr	int	serial
Kolicina	Kol	decimal	18,4
Zastavica (flag)	Zast	bit	1
Status	Stat	varchar	2
Iznos	Izn	decimal	18,4

Primjer nekih kratica u stvarnom projektu

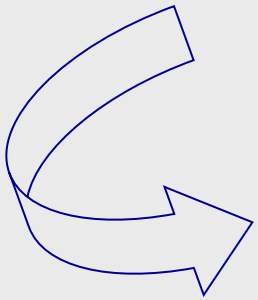
Automatski	Auto	
Cijena	Cij	
Datum	Dat	
Devize	Dev	
Generiranje	Gener	??
Grupa	Gr	
Knjizenje	Knjiz	
Koeficijent	Koef	
Način	Nac	
Namjena	Namj	??
Obračun	Obrac	
Osnovica	Osnov	??
Placanje	Plac	
Početak	Poc	
Porezni, Poreznog	Porez	
Pozicija	Poz	
Skladište	Sklad	??
Vrijednost	Vrij	
Vrsta	Vr	
Završetak	Zav	

Tehnika i stil programiranja

- **Tehnika i stil programiranja**
  - postaviti početne vrijednosti varijabli prije uporabe
  - ugraditi podrazumijevane (default) vrijednosti ulaznih podataka (pdv = 25%)
  - provjeravati zahtijevanost i valjanost ulaznih podataka
  - dosljedno formatirati podatke (npr. datum, novac, …)
  - pripaziti na granične vrijednosti podataka, indeksiranih varijabli ...
  - provjeriti moguće numeričke pogreške (množenje, dijeljenje s malim brojem)
  - izbjegavati usporedbu na jednakost brojeva s pomičnim zarezom
  - koristiti zagrade radi naglašavanja redoslijeda izračunavanja izraza
  - presložiti i pojednostavniti nerazumljive izraze
  - izbjegavati nepotrebna grananja
  - izbjegavati trikove (ne žrtvovati jasnoću radi "efikasnosti")
  - ponavljajuće blokove i izraze zamijeniti potprogramima
  - rekurziju koristiti samo za rekurzivne strukture podataka
  - prvo napraviti jasno i ispravno rješenje, a zatim brzo, lijepo ... rješenje
  - neučinkoviti kôd ne usavršavati, nego naći bolji algoritam

Jedan primjer

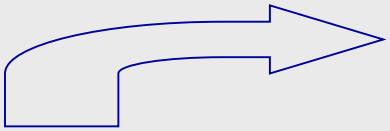
```
if ( ( a && !c ) || ( a && b && c ) ) {
  rezultat = 1;
} else if ( ( b && !a ) || ( a && c && !b ) ) {
  rezultat = 2;
} else if ( c && !a && !b ) {
 rezultat = 3;
} else {
  rezultat = 0;
}
```



```
static int[, ,] rezultatTable = {
{ {0, 3}, {2, 2} } , { {1, 2}, {1 , 1} }
};
...
rezultat = rezultatTable[a?1:0][b?1:0][c?1:0];
```

Primjer 🗁Kodiranje\StandardiKodiranja

- **Primjer 1**



```
if ( ( a && !c )
   || ( a && b && c ) ) {
  ...
```

```
rez = ( a && !c ) ;
rez |=( a && b && c ) ;
if (rez)
{ ...
```

- **Primjer 2**

```
  float f = 3.14f;
  bool b = false;
  if (f == 3.14)
  {
    b = true;
  }
// vrijednost b ?
// kada bi bila drukčija ?
```

- **Primjer 3**

```
 f = 10 * 0.01f;
 if (f == 0.1)
 {
   b = true;
 }
// vrijednost b ?
// rješenje?
```

Organizacija programskog koda

- **Programske knjižnice s funkcijama grupiranim po namjeni**
  - Održavanje, modularnost, opetovana iskoristivost , lokalizacija, ...
  - funkcije za rad s općim tipovima podataka (npr. nizovi znakova i datumi)
  - funkcije za rad s podacima u bazi podataka (npr. funkcije za upravljanje transakcijama i provjeru statusa izvedenih upita)
  - funkcije sučelja (npr. sustav izbornika, poruka i pomoći)
  - funkcije za održavanje baze podataka (npr. provjera konzistentnosti podataka i izrada rezervnih kopija)
  - funkcije za administriranje vanjskih uređaja (npr. terminali i pisači)
  - programski dio sustava zaštite (npr. definiranje programskih modula, funkcija i korisnika te rukovanje pravima pristupa programima i podacima)
- **Napraviti provjere i inicijalizaciju pri pokretanju programa**
  - Konfiguracijske datoteke
  - U nedostatku konfiguracije pokrenuti s predviđenim (default) vrijednostima
- **Suvisle poruke i sugestija popravka**
  - “Pristup bazi podataka nije moguć. Moguće neispravno ime ili zaporka.” umjesto “Pogreška”
    - u pozadini zapisati trag izvršenja radi dijagnostike problema

Reference

- **Code Complete (Second Edition), Steve McConnell (poglavlje 8)**