RUTINE VISOKOG KVALITETA

Code complete,poglavlje 7

Peleksic Ljubica
Razvoj Softvera 2

- Šta je rutina?
- Rutina je individualni metod ili procedura koja se poziva za odredjenu svrhu
- Šta je rutina visokog kvaliteta?
- Teže pitanje

Primer rutine

C++ Example of a Low-Quality Routine

```
void HandleStuff( CORP_DATA & inputRec, int crntQtr, EMP_DATA empRec,
   double & estimRevenue, double ytdRevenue, int screenX, int screenY,
   COLOR_TYPE & newColor, COLOR_TYPE & prevColor, StatusType & status,
   int expenseType )
{
int i;
for ( i = 0; i < 100; i++ ) {
   inputRec.revenue[i] = 0;
   inputRec.expense[i] = corpExpense[ crntQtr ][ i ];
   }
UpdateCorpDatabase( empRec );
estimRevenue = ytdRevenue * 4.0 / (double) crntQtr;
newColor = prevColor;
status = SUCCESS;
if ( expenseType == 1 ) {
     for ( i = 0; i < 12; i++ )
           profit[i] = revenue[i] - expense.type1[i];
     }
else if ( expenseType == 2 )  {
          profit[i] = revenue[i] - expense.type2[i];
          }
else if ( expenseType == 3 )
          profit[i] = revenue[i] - expense.type3[i];
          }
```

loše ime

previše argumenata koji nisu dokumentovani i poredani

magične konstante

menja se vrednost argumenta

nisu korišćene u rutini

- nema dokumentacije
- loše formatiran kod
- korišćenje globalnih promenljivih
- nema jednu svrhu

može doci do deljenja sa nulom

ne menja se vrednost argumentu

- Rutine čine programe lakšim za čitanje i razumevanje
- Čuvaju prostor i poboljšavaju performanse
- Kod bi bio mnogo veći i poboljšavanje performansi bi bilo znatno teže

Razlozi za kreiranje rutina

- Smanjivanje kompleksnosti
- Dobro ime rutine dokumentuje kod
- Izbegavanje ponavljanja koda

  ubacivanje zajednickog koda u novu rutinu u baznoj klasi i prebacivanje specijalizovanog koda u potklase ili stavljanje zajedničkog koda u rutinu koje pozivaju specijalizovane rutine

  modifikacije su jednostavnije

  manje se prostora koristi

  kod je pouzdaniji zato što proveravamo jedno mesto

  modifikacije su pouzdanije zato što se sve prave na jednom mestu

Razlozi za kreiranje rutina

- Treba manje koda za override kratkog i dobro napisanog koda i manja je mogućnost greške u potklasama
- Skrivanje redosleda po kome se izvršavaju dogadjaji
- Skrivanje operacija sa pokazivačima

  zbog njihove kompleksnosti kao i zbog eventualnog menjanja tipa podatka
- Poboljšavanje prenosivosti koda

Razlozi za kreiranje rutina

- Pojednostavljivanje kompleksnih testova

  sklanja detalje testova i davanje imena rutini po kome znamo šta testovi rade

- Poboljšavanje performansi

  optimizacija koda na jednom mestu,umesto na više

  lakše se nalaze nedostaci ako je kod na jednom mestu

  može da se upotrebi efikasniji algoritam ili efikasniji programski jezik

- Ne da bi rutine bile male

Operacije koje se čine previše jednostavnim da bi bile stavljene u rutine

- Praviti rutinu zbog par linija se ne čini opravdano, ali iskustvo pokazuje koliko one mogu biti korisne
- Poboljšavaju čitljivost koda
- Male operacije često prerastaju u veće

Dizajn

- Kohezija – koliko su povezane operacije unutar rutine
- Cilj je da rutina dobro radi jednu stvar i ništa drugo
- 50% rutina sa visokom kohezijom nije imalo grešaka i 18% rutina sa niskom kohezijom

Dizajn

- Funkcionalna kohezija – rutine imaju jednu operaciju

  sin(),eraseFile(),ageFromBirthDate()

- Sekvencijalna kohezija – rutine sadrze operacije koje moraju da se izvršavaju u odredjenom poretku i dale podatke

  od datuma rodjenja se računa godište zaposlenog i vreme do penzije

Dizajn

- Komunikaciona kohezija – kada operacije u rutini koriste iste podatke a nisu povezane na drugi način

  štampanje izveštaja i inicijalizovanje podataka – odvajamo u 2 rutine

- Vremenska kohezija – kada se operacije nalaze u istoj rutini zato što se obavljaju zajedno

  startUp(),shutDown()

Dizajn

Neke od neprihvatljivih kohezija su:

- Proceduralna kohezija

  operacije koje su u specifičnom redosledu a ne treba da budu zbog nekog razloga

- Logička kohezija

  više operacija koje su u jednoj rutini a koja se koristi odredjuje prosledjeni parametar,a one logički nisu povezane – kao da imaju if

  event-handler

- Slučajna kohezija

  operacije nemaju primetnu medjusobnu vezu

Imena rutina

- Opišite sve što rutina radi

  izlaze iz rutine, posledice
- Izbegavajte beznačajne ili često korišćene glagole

  HandleCalculation(),PerformServices(),OutputUser()
- Ne razlikujte rutine samo po brojevima
- Neka imena budu dugačka ako treba
- Koristite povratnu vrdnost rutine kao ime

Imena rutina

- Korisite jak glagol,pa iza njega objekat

  ne mora kod objektno orijentisanih jezika,zato sto je objekat uključen u pozivanje rutine

- Koristite suprotne reči za suprotne rutine

  add/remove ili insert/delete

- Ustanovite konvenciju za učestale operacije

Koliko dugačka rutina treba da bude?

- 50 -150 linija
- Staje na jedan ekran
- U modernom programiranju se kombinuju kratke rutine sa par dugačkih
- (Basili i Perricone 1984) kako se broj linija povećavao (do 200),smanjivao se broj grešaka
- (Shen et al 1985) broj grešaka i linija koda nije povezan

Koliko dugačka rutina treba da bude?

- (IBM 1986) Rutine koje su najviše podložne greškama su one sa više od 500 linija koda
- (Selby i Basili 1991) Od 450 rutina male(manje od 143 linije koda) su imale 23% manje grešaka ali su bile 2.4 puta jeftinije za popravljanje
- (Lind i Vairavan 1989) Kod najmanjemora biti menjan kada rutine imaju izmedju 100 i 150 linija koda

Koliko dugačka rutina treba da bude?

- Šta je sa rutinama u objektno orijentisan programima?
- Veliki procenat rutina će biti kratke
- Nekada će kompleksni algortmi voditi do većih rutina i tada treba da budu do 100-200 linija
- Dokazano je da takve rutine nisu više podložne greškama od onih kratkih
- Sa rutinama dužim od 200 linija koda treba biti oprezan

Argumenti

- Interfejsi izmedju rutina stvaraju najviše grešaka u programu
- Argumente treba slagati tako da idu prvo oni koji su samo ulazni,pa ulazno-izlazni,pa izlazni

  Ovakav redosled prikazuje redosled operacija u rutini
- Napravite sopstvene in i out reči

  kao u jeziku Ada,koristimo ih samo u dokumentacione svrhe

  dosta ljudi neće moći da razume ovo

  trebalo bi korsititi ovu tehniku na celom projektu

  kompajler neće proveravati da li se IN parametri modifikuju

Argumenti

- Ako više rutina koristi slične argumente, postavite ih u odredjen redosled

  kao funkcije prinf() fprintf()

- Koristite sve prosledjene parametre

  neiskorišćeni parametri povećavaju mogućnost greške

- Statusne i promenljive za greške stavljajte poslednje

- Ne koristite argumente kao promenljive u rutini

  za to koristite lokalne promenljive

Java Example of Improper Use of Input Parameters

```
int Sample( int inputVal ) {
   inputVal = inputVal * CurrentMultiplier( inputVal );
   inputVal = inputVal + CurrentAdder( inputVal );
   ...
   return inputVal;
}
```

Java Example of Good Use of Input Parameters

```
int Sample( int inputVal ) {
   int workingVal = inputVal;
   workingVal = workingVal * CurrentMultiplier( workingVal );
   workingVal = workingVal + CurrentAdder( workingVal );
   ...

   ...
   return workingVal;
}
```

Argumenti

- Dokumentujte ako postoje pretpostavke o argumentima

  ako su samo ulazni,samo izlazni..

  merne jedinice – inč,metar..

  interval u kome se mogu naći vrednosti

  specifične vrednosti koje ne treba da se pojave
- Koristite najviše 8 argumenata
- Konvencija input,modify,output

Funkcija ili procedura?

- C++,Java i Visual Basic podržavaju i funkcije i procedure
- Koja je razlika?
- Funkcija je rutina koja vraća vrednost
- Procedura je rutina koja ne vraća vrednost
- Smatra se da funkcija treba da ima jednu povratnu vrednost, kao u matematici

Funkcija ili procedura?

- Procedura bi mogla da ima ulazne, ulazno-izlazne i izlazne argumente – koloko god je potrebno
- Praksa je da se funkcija koristi kao procedura i da vraća samo jednu statusnu vrednost
- Može se koristit i procedura koja kao argument ima statusnu vrednost

Funkcija ili procedura?

```
if ( report.FormatOutput( formattedReport ) = Success ) then ...
```

```
report.FormatOutput( formattedReport, outputStatus )
if ( outputStatus = Success ) then ...
```

```
outputStatus = report.FormatOutput( formattedReport )
if ( outputStatus = Success ) then ...
```

Povratna vrednost

- Proverite da se uvek vraća vrednost, kojom god putanjom se izvršava
- Može se inicijalizovati povratna vrednost na početku
- Ne vraćajte pokazivače na lokalne podatke
- Lokalni podaci nestaju čim se završi izvršavanje rutine

Makro rutine

- Pazite šta se dešava sa makro-om za sve moguće vrednosti u kodu
- Ako makro ima više linija, treba koristiti vitičaste zagrade
- Korišćenje makroa umesto funkcija se smatra lošom praksom – riskantno je i teško za razumevanje
- Dajte naziv makrou kao rutini, da bi se eventualno prebacio u rutinu ako se počne širiti

Inline rutine

- Inline rutina omogućava programeru da gleda kod kao rutinu, a kompajler će konvertovati svaku instancu u inline kod u trenutku kompajliranja
- Inline rutine narušavaju enkapsulaciju u jeziku C++ zato što se pišu u header fajlovima
- Inline rutine zahtevaju da se generiše ceo kod kada god se rutina pozove, pa to povećava veličinu koda

Dodatni saveti

- Duboko ugnježđene petlje su znak da deo treba izvući u novu rutinu
- Ako rutina nema konkretnu svrhu, slabo ime je simptom
- Trudite se da rutina ima funkcionalnu koheziju
- Koristite makroe sa oprezom
- Koristite funkcije kada je primarna svrha funkcije da vrati vrednost koja se sadrži u njenom imenu