Struktura koda

(iz knjige Code Complete, poglavlje 19 - General Control Issues)

Jelena Marković 1109/2015

Logički izrazi

- Koristiti True i False izraze umesto vrednosti 0 i 1

```
Dim printerError As Integer
Dim reportSelected As Integer
Dim summarySelected As Integer
...
If printerError = 0 Then InitializePrinter()
If printerError = 1 Then NotifyUserOfError()

If reportSelected = 1 Then PrintReport()
If summarySelected = 1 Then PrintSummary()
```

Logički izrazi

```
Dim printerError As Boolean
Dim reportSelected As ReportType
Dim summarySelected As Boolean
...
If ( printerError = False ) Then InitializePrinter()
```

```
If ( printerError = True ) Then NotifyUserOfError()

If ( reportSelected = ReportType_First ) Then PrintReport()
If ( summarySelected = True ) Then PrintSummary()

If ( printerError = False ) Then CleanupPrinter()
```

Logički izrazi

- Porediti logičke vrednosti sa True i False implicitno

    while (not done)

    while (a = b)

    Bolje nego:

    while (done = False)

    while ((a = b) = True)

Logički izrazi

U jezicima koji nemaju logički tip podatka, poput C-a, korisno je definisati makroe:

#define TRUE (1 == 1)

define FALSE (!TRUE)

Logički izrazi

- Uprošćavanje komplikovanih logičkih izraza

Komplikovane izraze sa mnogo termova razbijemo na podizraze čije vrednosti dodeljujemo nekim logičkim promenljivama.

Ukoliko se logički izraz često ponavlja i skreće pažnju sa osnovnog toka koda, napravimo funkciju za valuaciju tog logičkog izraza.

Logički izrazi

- Formiranje pozitivnih logičkih izraza

Mnogo negacija u logičkom izrazu je teško za razumevanje.

“I ain’t not no undummy” - Homer Simpson

Logički izrazi

```
if (!statusOk) {
        // do something
}
else {
        // do something else
}
```

Logički izrazi

Korišćenje DeMorganovih zakona za uprošćavanje logičkih izraza:

not A and not B	not (A or B)
not A and B	not (A or not B)
A and not B	not (not A or B)
A and B	not (not A or not B)
not A or not B*	not (A and B)
not A or B	not (A and not B)
A or not B	not (not A and B)
A or B	not (not A and not B)

Logički izrazi

Zagrade povećavaju čitljivost logičkog izraza.

if (a < b == c == d) ???

if ((a < b) == (c == d)) malo bolje

Logički izrazi

- Evaluacija logičkih izraza u raznim programskim jezicima

Većina programskih jezika koristi “lenjo izračunavanje”.

Razlika između && i &, || i | operatora u Javi i C++ ?

```
if (y != 0  &&  x/y > 0) …
```

```
if (y != 0  &  x/y > 0) ...
```

Logički izrazi

- Numeričke izraze pišemo u rastućem redosledu:

MIN_ELEMENTS <= i and i <= MAX_ELEMENTS
i < MIN_ELEMENTS or MAX_ELEMENTS < i

umesto

i >= MIN_ELEMENTS and i <= MAX_ELEMENTS
i < MIN_ELEMENTS or i > MAX_ELEMENTS

Logički izrazi

- Poređenje sa nulom i NULL pokazivačem

U programskim jezicima 0 može da predstavlja numeričku vrednost, logički False, prvi indeks u nizu, terminator stringova…

Logički izrazi

Dobra praksa je eksplicitno poređenje numeričkih izraza sa 0 i pokazivača sa NULL:

while (balance != 0) umesto while (balance)

while (*charPtr !=’\ 0’) umesto while (*charPtr)

while (bufferPtr !=NULL) umesto while (bufferPtr)

Ugnježđivanje koda

Mali broj ljudi može brzo da pročita i razume više od 3 nivoa ugnježđenih if-ova (Noam Čomski, Džerald Vinberg)

Ugnježđivanje koda

- Pojednostavljenje ugnježđenog koda re-testiranjem kondicionala

```
if ( inputStatus == InputStatus_Success ) {
   // lots of code
   ...
   if ( printerRoutine != NULL ) {
      // lots of code
      ...
      if ( SetupPage() ) {
         // lots of code
         ...
         if ( AllocMem( &printData ) ) {
            // lots of code
            ...
         }
      }
   }
}
```

Ugnježđivanje koda

```
if ( inputStatus == InputStatus_Success ) {
   // lots of code
   ...
   if ( printerRoutine != NULL ) {
      // lots of code
      ...
   }
}

if ( ( inputStatus == InputStatus_Success ) &&
   ( printerRoutine != NULL ) && SetupPage() ) {
   // lots of code
   ...
   if ( AllocMem( &printData ) ) {
      // lots of code
      ...
   }
}
```

Ugnježđivanje koda

- Refaktorisati ugnježđeni if u skup if-then-else naredbi

```
if ( 10 < quantity ) {
   if ( 100 < quantity ) {
      if ( 1000 < quantity ) {
         discount = 0.10;
      }
      else {
         discount = 0.05;
      }
   }
   else {
      discount = 0.025;
   }
}
else {
   discount = 0.0;
}
```

Ugnježđivanje koda

```
if ( 1000 < quantity ) {
   discount = 0.10;
}
else if ( 100 < quantity ) {
   discount = 0.05;
}
else if ( 10 < quantity ) {
   discount = 0.025;
}
else {
   discount = 0;
}
```

Ugnježđivanje koda

- Refaktorisati ugnježđeni if u case naredbu

```
Select Case quantity
   Case 0 To 10
      discount = 0.0
   Case 11 To 100
      discount = 0.025
   Case 101 To 1000
      discount = 0.05
   Case Else
      discount = 0.10
End Select
```

Kompleksnost koda

Kompleksne probleme rešavamo tako što ih izdelimo na potprobleme manje kompleksnosti.

Dobra praksa: pisati razumljiv kod, koji ne zahteva da čitalac istovremeno pamti više od 5 entiteta.

Kompleksnost koda

- Kako merimo kompleksnost koda?

Obično je intuitivno jasno da li je neki kod previše komplikovan.

Tom MekKejb algoritam: brojanja “tačaka odluke” u kodu

1. Polazimo od 1
2. Dodajemo 1 svaki put kad naiđemo na neku od sledećih ključnih reči ili njihovih ekvivalenata: if while repeat for and or
3. Dodajemo 1 za svaki od slučajeva u case naredbi

Kompleksnost koda

Primer:

if (((status = Success) and done) or (not done and (numLines >= maxLines)))
then ...

Ovaj segment koda ima 5 tačaka odluke.

Kompleksnost koda

Analiza kompleksnosti funkcije ili bloka koda:

0 - 5 u redu

6 - 10 možda je potrebno redukovanje kompleksnosti koda

10 + deo koda funkcije izmestiti u drugu funkciju, koju pozivamo iz prve

(kompleksnost u smislu čitanja koda, na koliko stvari istovremeno moramo da budemo koncentrisani dok čitamo kod)

Rezime

Kvalitetan kod mora da bude čitljiv i razumljiv.

Jasni i čitljivi logički izrazi doprinose kvalitetu koda.

Ugnježđene if naredbe je relativno lako izbeći, a ako ih ne izbegnemo, dobili smo teško razumljiv kod.

Minimizovanje kompleksnosti je ključno kod pisanja kvalitetnog koda.

HVALA NA PAŽNJI!