Code Complete – Opšti problemi pri korišćenju promenljivih

Profesor: dr Vladimir Filipović

Student: Marija Đurić

Deklarisanje promenljivih na lak način

Implicitna deklaracija - jedna od najopasnijih karakteristika dostupna u nekom jeziku.

Šta raditi ako je vaš program na jeziku sa implictnim deklaracijama?

- Isključiti implicitne deklaracije(VisualBasic – Option Explicit)
- Deklarisati sve promenljive
- Korišćenje konvencija imenovanja
- Provera imena promenljivih

Smernice za inicijalizovanje promenljivih

Neispravna inicijalizacija podataka je jedan od najplodnijih izvora grešaka u kompjuterskom programiranju.

Problemi potiču iz toga da promenljiva sadrži neku početnu vrednost koju vi ne očekujete da ona sadrži. Ovo može da se desi za bilo koji od sledećih razloga:

- Promenljivoj nikad nije bila dodeljena vrednost. Njena vrednost je bilo nešto što se desilo sa bitovima u njenoj oblasti memorije u trenutku startovanja programa.
- Vrednost promenljive je zastarela. Promenljivoj je dodeljena vrednost u nekom trenutku, ali ta vrednost nije više validna.
- Delu promenljive je dodeljena vrednost a delu nije.

Smernice za inicijalizovanje promenljivih(2)

- Inicijalizovanje svake promenljive prilikom deklarisanja
- Inicijalizovanje svake promenljive blizu mesta gde je prvi put korišćena
  - *Deklarisati i definisati svaku promenljivu blizu gde se koristi*

PRIMER

Visual Basic primer loše inicijalizacije:

```
' deklarisanje svih promenljivih
Dim accountIndex As Integer
Dim total As Double
Dim done As Boolean
' inicijalizovanje svih promenljivih
accountIndex = 0
total = 0.0
done = False
' kod koji koristi accountIndex
' kod koji koristi total
```

Visual Basic primer dobre inicijalizacije:

```
Dim accountIndex As Integer
accountIndex = 0
' kod koji koristi accountIndex
Dim total As Double
total = 0.0

' kod koji koristi total
Dim done As Boolean
done = False                    d
' kod koji koristi done
```

Smernice za inicijalizovanje promenljivih(3)

- *Posebno obratiti pažnju na brojače i akumulatore*
- *Inicijalizovati podatke člana klase u njenom konstruktoru*
- *Proveriti potrebu za reinicijalizacijom*
- *Koristiti podešavanja kompajlera koja automatski inicijalizuju sve promenljive*
- *Iskoristite prednosti poruka upozorenja vašeg kompajlera*
- *Provera validnosti ulaznih parametara*
- *Korišćenje kontrolora pristupa memoriji za proveru loših pokazivača*
- *Inicijalizovati radnu memoriju na početku vašeg programa*

Obim

Obim, ili vidljivost, upućuje na opseg u kome su promenljive poznate.

- Mali obim – indeks petlje
- Veliki obim - tabela informacija o zaposlenima koja se koristi širom programa

Smernice koje se primenjuju na obim:

- Lokalizovati reference na promenljive držanjem ih blisko zajedno.

  Jedan metod za merenje koliko su bliske reference na varijable je da se izračuna "raspon" promenljive.

  Java primer raspona promenljivih:

  a = 0;

  b = 0;

Obim(2)

- Održavanje promenljivih živim najkraće moguće

  "doživljeno vreme" – vreme od prvog do poslednjeg referenciranja na promenljivu.

- Merenje doživljenog vremena promenljive

1 // inicijalizovanje svih promenljvih

2 recordIndex = 0;

3 total = 0;

4 done = false;

. . .

26 while (recordIndex < recordCount) {

27 . . .

28 recordIndex = recordIndex + 1; *poslednja referenca na recordIndex*

. . .

64 while (!done) {

. . .

69 if (total > projectedTotal) { *poslednja referenca na total*

70 done = true; *poslednja referenca na done*

Ovo su doživljena vremena za promenljive u ovom primeru:

recordIndex	(linija 28 - linija 2 + 1) = 27
total	(linija 69 - linija 3 + 1) = 67
done	(linija 70 - linija 4 + 1) = 67

. . .

25 recordIndex = 0; *Inicijalizacija recordIndex-a*

26 while (recordIndex < recordCount) {

27 ...

28 recordIndex = recordIndex + 1;

. . .

62 total = 0; *Inicijalizacija total i done 4*

63 done = false;

64 while (!done) {

. . .

69 if (total > projectedTotal) {

70 done = true;

Ovo su doživljena vremena za promenljive u ovom primeru:

recordIndex	(linija 28 - linija 25 + 1) = 4
total	(linija 69 - linija 62 + 1) = 8
done	(linija 70 - linija 63 + 1) = 8

Prosečno doživljeno vreme (4 + 8 + 8) / 3 ≈ 7

Opšte smernice za minimizovanje obima

- Inicijalizovanje promenljivih koje se koriste u petlji odmah pre petlje pre nego na početku rutine koja sadrži petlju
- Nemojte dodeljivati vrednost promenljivoj sve do mesta gde ćete je koristiti
- Počnite sa najrestriktnijom vidljivošću, i širite obim promenljive samo ako je neophodno
- Grupisani iskazi

```
void SummarizeData (...) {

. . .

GetOldData( oldData, &numOldData );

GetNewData( newData, &numNewData
    );

totalOldData = Sum( oldData,
    numOldData );

totalNewData = Sum( newData,
    numNewData );

PrintOldDataSummary( oldData,
    totalOldData, numOldData );

PrintNewDataSummary( newData,
    totalNewData, numNewData );

SaveOldDataSummary( totalOldData,
```

```
void SummarizeDaily( ... ) {

GetOldData( oldData, &numOldData );

totalOldData = Sum( oldData,
    numOldData );

PrintOldDataSummary( oldData,
    totalOldData, numOldData );

SaveOldDataSummary( totalOldData,
    numOldData );

. . .

GetNewData( newData, &numNewData
    );

totalNewData = Sum( newData,
    numNewData );

PrintNewDataSummary( newData,
```

Trajnost

Promenljive mogu da traju:

- dok postoji poseban blok koda ili rutine. Deklarisane promenljive koje se javljaju u FOR petlji u C++ ili Javi su primeri ovog tipa trajnosti.

- koliko god im vi dozvolite. U Javi, promenljive koje se kreiraju pomoću ***new*** traju sve dok ih ne ocisti garbage collector. U C++-u promenljive traju sve dok ih ne izbrišete.

- dok postoji program. Globalne promenljive u većini jezika odgovaraju ovom opisu, kao sto su statičke promenljive u C++ i Javi.

- zauvek. Takve promenljive mogu da uključe vrednosti koje ste uskladištili u bazu podataka između izvršavanja

Sledeći koraci vam mogu pomoći da izbegnete ovakav tip problema:

-koristite **debug** da proverite kritične promenljive. Ako vrednosti nisu prihvatljive, prikažite na ekranu upozorenje koje vas upućuje da nađete neispravnu inicijalizaciju.

-Napišite kod koji pretpostavlja da podaci nisu trajni. Na primer, ako promenljiva ima određenu vrednost kada izađete iz rutine, ne pretpostavljajte da će imati istu vrednost kad sledeći put u nju uđete. Ovo se ne odnosi ako koristite specifične jezičke karakteristike koje garantuju da će vrednost ostati ista, kao što je statička promenljiva u C++ i Javi.

-Razvijte naviku deklarizacije i inicijalizacije svih podataka pre nego što se upotrebe. Ako primetite da se koriste podaci bez prethodne inicijalizacije, imajte sumnje u vezi toga.

Vreme povezivanja

Vreme povezivanja (spajanja) – vreme kada se promenljiva i njena vrednost spajaju.

Ono što vam najviše može koristiti jeste da ih povežete što kasnije moguće. Uglavnom, što ***kasnije*** postavite ***vreme*** povezivanja, to će ***kod*** biti ***fleksibilniji***.

Primer: *Jave kada se promenljiva povezuje u vreme pisanja koda*:

titleBar.color = 0xFF; // 0xFF is hex value for color blue

Fiksno kodiranje kao ovo je skoro uvek loša ideja!

Primer povezivanje u malo kasnijem vremenu, kada je kod kompajliran:

private static final int COLOR_BLUE = 0xFF;

private static final int TITLE_BAR_COLOR = COLOR_BLUE;(imenovana konstanta***)***

...

titleBar.color = TITLE_BAR_COLOR;

Primer povezivanja za vreme pokretanja programa:

titleBar.color = ReadTitleBarColor();(rutina koja čita vrednost dok se program izvršava**)**

Što je vreme povezivanja ranije, to su fleksibilnost i kompleksnost

Veza između tipova podataka i kontrolnih struktura

- Sekvencijalni podaci se prevode u sekvencijalne iskaze u programu.
- Selektivni podaci se prevode u IF i CASE iskaze u programu.
- Iterativni podaci se prevode u FOR, REPEAT i WHILE petlje u programu

Korišćenje svake promenljive za tačno jednu svrhu

```
// This code assumes that
   (b*b-4*a*c) is positive.


 temp = Sqrt( b*b - 4*a*c );

root[O] = ( -b + temp ) / ( 2
   * a );

root[1] = ( -b - temp ) / ( 2 *
   a );

// swap the roots


temp = root[0];
```

Ispravniji primer :

```
// Compute roots of a
   quadratic equation.

// This code assumes
   that (b*b-4*a*c) is
   positive.


discriminant = Sqrt(
   b*b - 4*a*c );
```

Korišćenje svake promenljive za tačno jednu svrhu(2)

Drugi način u kome promenljiva može biti korišćena u više od jedne svrhe jeste to da ima različite vrednosti za promenljivu koja označava razlicite stvari. Na primer:

- vrednost u promenljivoj *pageCount* može predstavljati broj odštampanih strana, osim ako ne iznosi -1, u tom slučaju bi značilo da je načinjena neka greška
- Promenljiva *bytesWritten* može biti broj bajtova napisanih na izlaznoj datoteki, osim ako je njegova vrednost negativna, u tom slučaju predstavlja broj diskova koji je korišćen za izlaz

Hvala na pažnji!