Cvičenie z PTS

13.4.2010

Zrozumiteľnosť kódu

Atribúty kvality softvéru

Externé atribúty kvality:

- spoľahlivosť
- flexibilnosť
- znovupoužiteľnosť
- kompatibilnosť
- efektívnosť
- portabilnosť
- verifikovateľnosť
- jednoduchosť používania a správy
- ...

Interné atribúty kvality:

- modulárnosť
- **zrozumiteľnosť kódu** („kvalita v malom“)
- ...

Prečo zrozumiteľný kód ?

Autor je len prvý v rade zainteresovaných, nasledujú:

- kolegovia (zastupovanie),
- quality assurance,
- **údržba** (t.j. oprava chýb, rozširovanie, úpravy),
- opakované použitie (niekedy),
- partnerská firma, zákazník, verejnosť (občas),
- ...

Bežné výhovorky:

- efektívnosť „optimalizovaného“ kódu
- nedostatok času
- „spravím to neskôr“
- ubíjanie kreativity

```
/*************************************************************************/
/* int16        CNT_Read (int32 *R1, int32 *R2)                          */
/* - nacita zakladne informacie z CNT suboru                             */
/* - vracia 1, resp. FATAL_ERROR                                         */
/*************************************************************************/

int16    CNT_Read(int32 *R1, int32 *R2) {

int16           chyb;
int32           sss;
CNT_Record  Cnt1,Cnt2;

chyb = fread(&Cnt1,sizeof(CNT_Record),1,CNTs);
if (chyb != 1)
        {
        return (FATAL_ERROR);
        }
if ( Cnt1.Idtype != 1 ) /* Heh, the first record is for tree 2 */
        {
        sss = 0;        /* Ha Ha Ha :( */
        chyb = fseek(CNTs,sss,SEEK_SET);     /* rewind the file */
        if (chyb != 0)
                {
                return (FATAL_ERROR);
                }
        chyb = fread(&Cnt2,sizeof(CNT_Record),1,CNTs);

/* ??????? */
        if (chyb == 1 )
                chyb = fread(&Cnt1,sizeof(CNT_Record),1,CNTs); /* ??? */
        }
else
        chyb = fread(&Cnt2,sizeof(CNT_Record),1,CNTs);
```

```
/* ------------------------------------------------------- NACITAJ_NOVU_VERZIU

  Spyta sa na oznacenie novej verzie pri opusteni programu a nastavi ziskanu hodnotu */

void nacitaj_novu_verziu ()
{
        char c, buffer [VELKOST_VERZIE+1];

        /*
        **  Nacita z klavesnice nove cislo verzie; ak pouzivatel
        **  stlaci ESC alebo nezada verziu, vrati sa bez vykonania zmeny.
        */

        zobraz_stav (VSTUP_Z_KLAVESNICE);
        gotoxy (1, video_rows);
        c = getline (buffer, VELKOST_VERZIE,
                     " Oznacenie novej verzie INFO-suboru: ",
                     verzia_info_suboru);
        if (c == KEY_ENTER)
          obnov_obrazovku ();
        if (c == KEY_ESC || *buffer==0)
          return;
        zobraz_stav (CAKAJTE_PROSIM);

        /*
        **  Ak je pouzivatelom vlozene cislo verzie ine ako aktualne, zmeni
        **  cislo verzie v systeme.
        */

        if (strcmp (verzia_info_suboru, buffer))
        {
          strcpy (verzia_info_suboru, buffer);
          zmen_verziu ();
        }
}
```

Základné pravidlá ...

- univerzálne pravidlá neexistujú
- kód má byť ľahko čitateľný, ľahko písateľný
- vychádzať z existujúcich konvencií
- jednoduchosť
- nič nie je dogma

O čo konkrétne ide...

- mená (tried, funkcií, atribútov, premenných)
- konštanty
- komentáre
- rozmiestnenie textu (formátovanie)
- vnorené „if“

Mená

- mená tried a operácií/atribútov (interface)
  - plné mená (nie skratky), vystihujúce podstatu veci - ak treba, aj viacero slov, nie však príliš veľa

```
class suc
{
     int cs;
     long datv;
     char *vyr;
     float hmot;
     ...
}
```

  - neopakovať zbytočne názov triedy
    - `suciastka.cislo_suciastky`, `suciastka.datum_vyroby_suciastky`, `suciastka.vyrobca_suciastky`
  - použiť pojmy z aplikačnej oblasti
  - konzistentnosť (jeden objekt – jeden pojem)
    - `kontajner.vaha`, `kontajner.hmotnost_bez_obsahu`
- mená lokálnych premenných
  - nie sú vždy vhodné plné mená

```
void copy_string (char *destination, char *source)
{
   char *p, *q;      // pointers to source
                     // and destination strings
   p = source;
   q = destination;
   while (*p != 0)
       *q++ = *p++;
}
```

```
void copy_string (char *destination, char *source)
{
   char *source_pointer, *destination_pointer;

   source_pointer = source;
   destination_pointer = destination;
   while (*source_pointer != 0)
     *destination_pointer++ = *source_pointer++;
}
```

Mená 2

- konzistentné používanie veľkých a malých písmen a „_“
  - `RedCircle` vs. `redcircle` vs. `red_circle`
  - `dlhyANicNehovoriaciIdentifikator` vs. `dlhy_a_nic_nehovoriaci_identifikator`
- jednotný jazyk
- obmedziť používanie prefixov / postfixov (`m_lpstrNameS`)
- nepoužívať v jednom kontexte vizuálne podobné mená (`DlhaLinkaDevat`, `DlhaLinkaDesat`)

Mená 3:
gramatické kategórie

- triedy: podstatné mená
  - `Tlaciaren` vs. `Tlacit`

- procedúry/operácie/metódy: slovesá v neurčitku alebo v rozkazovacom spôsobe
  - `Tlaciaren.vytlac` vs. `Tlaciaren.vytlacenie`

Konštanty

Nepoužívať priamo hodnoty konštánt (s výnimkou nulového prvku a prípadne práve raz používaných reťazcov)

```
initialize_connection_table (50);

prejdi_do_stavu (4);
...
prejdi_do_stavu (1);
```

vs.

```
prejdi_do_stavu (VSTUP_Z_KLAVESNICE);
...
prejdi_do_stavu (CAKAJTE_PROSIM);
```

Komentáre

- sú nutnosťou
- komentáre na úrovni:
  - kódu samotného
  - tried (resp. modulov)
  - procedúr, funkcií, atribútov

Komentáre v kóde

```
// vytvor prázdny reťazec
String newStr = "";
// vykonaj cyklus od 0 po n-1
for (int i = 0; i < n; i++)
   // ak je v a[] na pozícii i nula, zapíš ju do newStr, inak zapíš 1
   newStr += a[i] == 0 ? "0" : "1";
```

```
// pole a[n] obsahuje binárny rozvoj čísla - prevedieme ho na reťazec
// do newStr
String newStr = "";
for (int i = 0; i < n; i++)
   newStr += a[i] == 0 ? "0" : "1";
```

- o úroveň abstrakcie vyššie než samotný kód
- nie nutne všade
- majú byť stručné

Komentáre v hlavičke modulu

- Názov modulu
- Autor(i)
- Verzia
- Dátum vytvorenia
- Subsystém, do ktorého modul patrí
- História zmien
- Kľúčové slová
- ...

} mal by obsahovať systém na správu konfigurácií

- Popis modulu:
  - popis rozhrania modulu (aké služby poskytuje)
  - popis implementácie modulu
- Väzby na iné moduly, resp. externé súbory
- Postup pri kompilácii
- Poznámky k portabilite
- Otvorené problémy („to do“, „known bugs“)
- ...

Existuje softvér na generovanie dokumentácie z týchto hlavičiek (napr. javadoc)

```
/*
**  MODULE NAME:
**
**      MAGNETIC TAPE
**
**  MODULE DESCRIPTION:
**
**      This program exercises a magnetic tape unit using QIO calls to create and write
**      a file and then to read it in reverse. See also the description under main().
**
**  AUTHORS:        Digital Equipment Corporation
**
**  VERSION:        1.0
**
**  CREATION DATE:  23 March 1993
**
**  FACILITY:       SYS$EXAMPLES
**
**  PORTABILITY ISSUES:
**
**      At this writing, the OpenVMS VAX File Information Block (FIB)
**      definition header file differs slightly from the corresponding
**      file for OpenVMS AXP. (...)
**
**  COMPILATION PROCEDURES:
**
**      To compile this routine, use
**
**      $ CC MAGNETIC_TAPE + SYS$LIBRARY:SYS$LIB_C /LIBRARY
**
**  MODIFICATION HISTORY:
**
**      23-Mar-1993 Conversion from MAGNETIC_TAPE.MAR
*/
```

Rozmiestnenie textu

dodržiavať konzistentný prístup pri odsadzovaní

–

if, then, else, „begin“ / „end“, ...

```
for (i=0; i<n; i++)
    {
  if (a[i] > MAX_PRVOK)
  a[i] = MAX_PRVOK;
  else if (a[i] < MIN_PRVOK) {
  a[i] = MIN_PRVOK; pocet++; }
    }
```

```
for (i=0; i<n; i++)
{
  if (a[i] > MAX_PRVOK)
    a[i] = MAX_PRVOK;
  else if (a[i] < MIN_PRVOK)
  {
    a[i] = MIN_PRVOK;
    pocet++;
  }
}
```

- používať prázdne riadky v kóde na oddelenie blokov kódu
- používanie medzier:
  - f(x) alebo f (x) – nie f(x)
  - (a, b, c) – nie (a ,b ,c)
  - ...
- 1 príkaz na riadok, neprekračovať šírku obrazovky

- existujú automatické nástroje (prettyprinters)

Vnorené „if“

Vyhnúť sa priveľkej hĺbke vnorenia – napr. nie viac ako 3 úrovne.

```
if (sirka > 30 && dlzka > 120)
{
    if (sirka <= 60 && dlzka <= 150)
        oblast = 1;
    else if (sirka <= 90 && dlzka <= 150)
        oblast = 2;
    else oblast = 0;
} else oblast = 0;
```

```
if (sirka > 30 && sirka <= 60 && dlzka > 120 && dlzka <= 150)
   oblast = 1;
else if (sirka > 60 && sirka <= 90 && dlzka > 120 && dlzka <= 150)
   oblast = 2;
else oblast = 0;
```

Triky

- vyhnúť sa programátorským „trikom“ typu
  - `while (*s++ = *t++) ;`
  - volanie nedokumentovaných funkcií knižníc
  - kód zoptimalizovaný tak, že je nečitateľný
  - …
- ak sa im nedá vyhnúť
  - uvedomiť si riziká
  - okomentovať
  - zabaliť do zvláštneho modulu

Záver

- držať sa týchto „dobrých zvykov“ je vhodné v každom prípade

- pri veľkých projektoch (resp. vo veľkých organizáciách) je potrebné mať takéto alebo podobné „guidelines“ zachytené **v písomnej podobe**

Výber platformy, nástrojov a komponentov

Príklady technológií

- programovací jazyk
- IDE (editor, kompilátor, debugger, grafický editor, nástroje na lokalizáciu, ...)
- operačný systém
- databázový server
- aplikačný server
- komunikačný middleware
- knižnice
- podporné nástroje (CM, databáza chýb, ...)

Potenciálne problémy

- nástroj nefunguje tak, ako má (funkčné chyby, chyby pri väčšej záťaži, bezpečnostné nedostatky, kompatibilita, ...)
- nedostatky v dokumentácii
- know-how (spôsob použitia, riešenie problémov, školenia, literatúra, ...)

- problémy sa dajú očakávať najmä v nových verziách nástrojov (resp. vo verziách x.0)

- všeobecné riešenie: **vyskúšať najprv na menej významných projektoch**

Výber programovacieho jazyka

- skúsenosti
- vhodnosť
- úroveň abstrakcie

Knižnice

- potenciálne chyby
- nemožnosť ovplyvniť funkčnosť
  - najmä pri zmene požiadaviek
- nemožnosť ovplyvniť ďalší vývoj
- závislosť na poskytovanej podpore