Introdução à Computação-II

Documentação

Neste tópico abordaremos a documentação de programas em C.

Prof. Ciro Cirne Trindade

UNISANTOS

Por quê documentar?

- A documentação de um programa complementa seu código para deixá-lo mais compreensível
- Durante a sua vida profissional você dificilmente irá desenvolver código sozinho
- Depois de algum tempo, nem você lembrará de certas decisões de projeto que tomou

Formas de documentação

- Comentários no próprio código
  - Nesta aula iremos tratar da documentação no próprio código através de comentários
- Documentos complementares
- É importante que a apresentação do seu código facilite a leitura
  - Comentários
  - Bom leiaute

"Qualquer idiota pode escrever um programa que um computador pode entender. Bons programadores escrevem programas que humanos podem entender." - Martin Fowler

Aonde por comentários

- Não se deve comentar o óbvio
  - Evite “sujar” o código com comentários desnecessários
  - Por exemplo:
    - `x = 0; // atribui zero a variável x`

Aonde por comentários

- Seu programa deve possuir, no mínimo, os seguintes comentários:
  - Nome do arquivo fonte
  - Descrição sucinta do que o programa faz
  - Identificação do(s) programador(es), curso, disciplina, data de última alteração
  - Explicar o que cada uma das funções que compõem o programa faz

Documentação inicial do programa

```
/* maior.c
 *
 * Programa que determina o maior entre
 * dois números inteiros.
 *
 * Fulano de Tal (Sistemas de Informação)
 * Beltrano da Silva (Ciência da Computação)
 *
 * Disciplina: Introdução à Computação-II
 *
 * 02/09/2014
 */
...
```

Documentação de funções

- Deve dar instruções precisas e completas sobre o uso correto da função
  - Especificar o que entra: que dados a função recebe
  - E o que sai: que informações a função devolve
  - Descrever a relação entre o que entra e o que sai
  - Informa as possíveis restrições sobre as entradas
- Deve-se ressaltar ***o que*** a função faz e não ***como*** ela faz

Exemplo de documentação de funções

```
/* função que recebe um inteiro n >= 1 e um
 * vetor de n inteiros v e devolve o valor
 * de um elemento máximo de v */
int max (int v[], int n)
{
   int i, x = v[0];
   for (i = 1; i < n; i++) {
      if (v[i] > x) {
         x = v[i];
      }
   }
   return x;
}
```

Documentação de funções

- A documentação diz o que a função faz, mas não perde tempo tentando explicar como a função faz o que faz
- A documentação menciona todos os parâmetros da função e não faz menção a quaisquer outras variáveis
- Observe também a ausência de comentários inúteis, como, `"o índice i vai percorrer o vetor"`

Exemplos de má documentação (1/2)

- Considere o cabeçalho da função

```
int max (int v[], int n) { … }
```

  - `/* a função devolve o valor de um elemento máximo de um vetor */`: muito vago, não cita os parâmetros
  - `/* a função devolve o valor de um elemento máximo de um vetor v*/`: ainda muito vago, não explica o papel de `n`

Exemplos de má documentação (2/2)

- Considere o cabeçalho da função

```
int max (int v[], int n) { … }
```

  - `/* a função devolve o valor de um elemento máximo de um vetor v que tem n elementos*/`: melhor, mas sonega a importante restrição `n >= 1`

Exercícios

1) Escreva a documentação correta da função abaixo:

```
int soma (int n, int v[])
{
    int i, x = 0;
    for (i = 0; i < n; i++) {
        x += v[i];
    }
    return x;
}
```

2) Critique a seguinte documentação de uma função:

```
/* função que recebe números inteiros p, q, r, s
 * e devolve a média aritmética de p, q, r */
```

Invariantes

- O corpo de muitas funções contém um ou mais processos iterativos (tipicamente controlados por um **`for`** ou **`while`**)
- A documentação do programa pode ser enriquecida dizendo quais os invariantes dos processos iterativos
- Um **invariante** é uma relação entre os valores das variáveis que *vale no início de cada iteração* do processo iterativo

Exemplo de documentação de invariantes

```
int max (int v[], int n)
{
   int i, x = v[0];
   for (i = 1; i < n; i++) {
      /* x é um elemento máximo de v[0..i-1] */
      if (v[i] > x) {
         x = v[i];
      }
   }
   return x;
}
```

Exercícios

3) Qual o invariante do processo iterativo da função soma do exercício 1?

4) Escreva uma documentação completa para a função abaixo.

```
int menor_pos(int n, int v[])
{
    int i, pos = 0;
    for (i = 1; i < n; i++) {
        if (v[i] < v[pos])
            pos = i;
    }
    return pos;
}
```

Bom leiaute

- Um leiaute adequado facilita muito a leitura e compreensão de um programa
- Observe especialmente:
  - O recuo das linhas em relação à margem esquerda da página,também conhecido como indentação (*indentation*)
    - 3 a 7 espaços
  - Os espaços que separam as palavras dos símbolos

Sugestões (1/6)

- A chave de abertura que marca o início do corpo de uma função deve estar na coluna 1
- Exemplo:

```
int max (int v[], int n)
{
    …
}
```

Sugestões (2/6)

- Use um espaço para separar cada palavra da palavra seguinte
  - Note que os símbolos `=`, `<=`, `while`, `if`, `for` , etc. contam como palavras
- Deixe um espaço depois, mas não antes, de cada sinal de pontuação
- Deixe um espaço depois, mas não antes, de fechar um parêntese
- Deixe um espaço antes, mas não depois, de abrir um parêntese

Sugestões (3/6)

- Portanto:
  - Jamais escreva `while(j < n)` no lugar de `while (j < n)`
  - Jamais escreva `else{` no lugar de `else {`
  - evite escrever `for(i=1;i<n;i++)` no lugar de `for (i = 1; i < n; i++)`

Sugestões (4/6)

- Exceções:
  - `x.dado` e não `x . dado`
  - `x->prox` e não `x -> prox`
  - `x[i]` e não `x [i]`
  - `x++` e não `x ++`
- É usual suprimir o espaço antes do parêntese na chamada a uma função
  - `max(v, 10)` e não `max (v, 10)`

Sugestões (5/6)

- Se os argumentos de uma função não cabem em uma linha, formate a definição assim:

```
int muitos_args(int a_int, double a_double,
                float a_float, char a_char)
{
    …
}
```

Sugestões (6/6)

- Quando você dividir uma expressão em mais de uma linha, divida-a antes de um operador, não após um
- Por exemplo:

```
if (isto_eh_longo && bar > win(x, y, z)
    && resto_da_condicao)
```

Exercício

- Reescreva o trecho de programa abaixo com leiaute “civilizado”.

```
int func(int n,int v[]){int
i,j;i=0;while(i<n){
if(v[i]!=0) ++i;else{for(j=i+1;j<n;++j)
v[j-1]=v[j];--n;}}return n;}
```

Referências

- FEOFILOFF, Paulo. *Algoritmos em Linguagem C*. Rio de Janeiro: Elsevier, 2009
- GNU Coding Standards. *Make the best use of C*. Disponível em: <http://www.gnu.org/prep/standards/html_node/Writing-C.html>.