Boas Práticas de Programação em C
Artistic Style e Doxygen

Filipe A. N. Verri

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

SCC0202 - Algoritmos e Estruturas de Dados I

Professor Thiago A. S. Pardo

31 de agosto de 2014

Sumário

Exemplo

Legível? [1]

```
typedef struct n{int a:3,b:29;struct n*c;}t;t*
f();r(){}m(u)t*u;{t*w,*z;z=u->c,q(z),u->b=z->b*10,
w=u->c=f(),w->a=1,w->c=z->c;}t*k;g(u)t*u;{t*z,*v,*p,
*x;z=u->c,q(z),u->b=z->b,v=z->c,z->a=2,x=z->c=f(),x
->a=3,x->b=2,p=x->c=f(),p->c=f(),p->c->a=1,p->c->c=
v;}int i;h(u)t*u;{t*z,*v,*w;int c,e;z=u->c,v=z->c,q(
v),c=u->b,e=v->b,u->b=z->b,z->a=3,z->b=c+1,e+9>=c&&(
q(z),e=z->b,u->b+=e/c,w=f(),w->b=e%c,w->c=z->c,u->c=
w);}int(*y[4])()={r,m,g,h};char*sbrk();main(){t*e,*p,*
o;o=f(),o->c=o,o->b=1,e=f(),e->a=2,p=e->c=f(),p->b=2,
p->c=o,q(e),e=e->c,(void)write(1,"2.",2);for(;;e=e->c)
{q(e),e->b=write(1,&e->b["0123456789"],1);}}t*f(){
return i||(i=1000,k=(t*)sbrk(i*sizeof(t))),k+=i;}q(p)
t*p;{(*y[p->a])(p);}
```

[1] Best complex task done in a complex way.
http://www.ioccc.org/winners.html

Exemplo

Legível?

```
void foo(Pilha *q){ /* funcao foo */
elem x,y[10]; /* cria elem x */
int a,b=-1,j; /* create integers a, b, c*/
  while(!IsEmpty(P))
{Pop(q,&x,&a);
 y[++b]=x;}
 for(j=0;j<=i;++j)Push(q,&y[j],&a);
}
```

Problema

- C ignora espaços em branco, tabulações e quebras de linha
- C não define nomenclatura padrão de variáveis, tipos e funções
- C não especifica como o código deve ser documentado

- Python e Go, por exemplo, não ignoram espaços
- Java, Go, Python, por exemplo, especificam nomenclaturas para variáveis, tipos e funções
- Java e Python, por exemplo, especifica como deve ser a documentação (Javadoc e Docstring)

Solução

Usar convenções de codificação!
Guias de Estilo C

Notação Húngara [2]

[2] http://msdn.microsoft.com/en-us/library/aa260976(VS.60).aspx

História

- Convenção de nomes criada pelo arquiteto chefe da Microsoft, Dr. Chales Simonyi
- Difundida no livro *Programming Windows* de Charles Petzold
- Recebeu esse nome porque o código não parecia mais estar escrito em inglês
- Consiste em adicionar informações de tipos, parâmetros e retornos nos nomes de variáveis e funções

Exemplo

Inverter Pilha - Notação Húngara

```
void InverterPPilha(Pilha *pPilhaArgumento) {
    elem elemAuxiliar, rgElemAuxiliar[TamPilha];
    int iErro, iNumeroDeElementos=-1, iIterador;

    /* Armazena os elementos da pilha original em um vetor
       auxiliar */
    while (!iIsEmptyStack(pPilhaArgumento)) {
        PopPPilhaPElemPI(pPilhaArgumento, &elemAuxiliar, &iErro)
        ;
        rgElemAuxiliar[++iNumeroDeElementos] = elemAuxiliar;
    }

    /* Retorna os elementos para a pilha na ordem inversa */
    for (iIterador = 0; iIterador <= iNumeroDeElementos;
       iIterador++) {
        PushPPilhaPElemPI(pPilhaArgumento, &rgElemAuxiliar[
       iIterador], &iErro);
    }
}
```

Guia de Estilo C de Indian Hill [3]

[3] http://www.maultech.com/chrislott/resources/cstyle/indhill-cstyle.pdf

História

- Escrito por um comitê formado na AT&T de Indian Hill
- Última revisão em junho de 1990
- Bastante abrangente (28 páginas)

Características Principais

- Limite de 1000 linhas por arquivo e 79 colunas por linha
- Nome dos arquivos: *.c, *.h, *.s, *.o, Makefile, README, etc. . .
- Include guards e restrições severas para os .h
- Convenção de nomes
- Abuso de espaços em branco

Exemplo

Inverter Pilha - Indian Hill

```c
/*
 * INPUT: Pilha a ser invertida.
 */
void
pilha_inverter(pilha_t *pilha)
{
    elem_t v[TAM_PILHA];
    elem_t x;
    int    erro;
    int    i = -1
    int    j;

    /* Armazena os elementos da pilha original em um vetor auxiliar */
    while (!pilha_is_empty(pilha)) {
        pilha_pop(pilha, &x, &erro);
        v[++i] = x;
    }

    /* Retorna os elementos para a pilha na ordem inversa */
    for (j = 0; j <= i; j++)
        pilha_push(pilha, &v[j], &erro);
}
```

Padrão de Codificação GNU [4]

[4] http://www.gnu.org/prep/standards/standards.pdf

História

- Escrito por Richard Stallman e voluntários do projeto GNU
- Projeto ativo, última atualização em maio de 2014
- Extremamente abrangente (85 páginas)

Características Principais

- Limite de 79 colunas por linha
- Abrir e fechar chaves de funções na coluna 1
- Comentário e código em inglês
- Indentação com dois espaços
- Nomes de variáveis separados por *underline*
- Variáveis globais com nomes descritivos e locais com nomes curtos

Exemplo

Inverter Pilha - GNU

```
void invert(stack *p)
{
    elem x, v[STACK_SIZE];
    int error, i = -1, j;

    /* stores stack elements in an auxiliary vector */
    while (!is_empty(p))
      {
        pop (p, &x, &error);
        v[++i] = x;
      }

    /* push the elements in reverse order */
    for (j = 0; j <= i; j++)
      push (p, &v[j], &error);
}
```

Estilo de Codificação do Kernel Linux [5]

"First off, I'd suggest printing out a copy of the GNU coding standards, and NOT read it. Burn them, it's a great symbolic gesture." Linus Torvalds.

[5] `https://www.kernel.org/doc/Documentation/CodingStyle`

História

- Escrito por Linus Torvalds como *sugestão* para manter o código do Kernel organizado
- Muito simples
- Filosofia: *"Avoid tricky expressions."*

Características Principais

- Indentação com 8 espaços
- Uma declaração por linha
- "Limite" de 80 colunas por linha
- Não use typedefs em estruturas e ponteiros
- Não use espaço em chamada de funções, mas use depois de if, switch, ...
- Variáveis globais com nomes descritivos e locais com nomes curtos
- Único retorno por função
- Funções com "no máximo" 24 linhas e 10 variáveis locais (*function-growth-hormone-imbalance syndrome*)

Exemplo

Inverter Pilha - Kernel

```
void invert(struct stack *p)
{
        elem x, v[STACK_SIZE];
        int error, i = -1, j;

        /* stores stack elements in an auxiliary vector */
        while (!is_empty(p)) {
                pop(p, &x, &error);
                ++i;
                v[i] = x;
        }

        /* push the elements in reverse order */
        for (j = 0; j <= i; ++j) {
                push(p, &v[j], &error);
        }
}
```

Qual Convenção Usar?

Dentre dezenas de convenções, qual devo escolher?

Boas Práticas de Programação

- Mantenha o código simples

Boas Práticas de Programação

- Mantenha o código simples
- Documente

Boas Práticas de Programação

- Mantenha o código simples
- Documente
- Indente

Boas Práticas de Programação

- Mantenha o código simples
- Documente
- Indente
- Mantenha o estilo uniforme em todo o projeto

Boas Práticas de Programação

- Mantenha o código simples
- Documente
- Indente
- Mantenha o estilo uniforme em todo o projeto
- Mantenha um único idioma (preferencialmente em inglês)

Boas Práticas de Programação

- Mantenha o código simples
- Documente
- Indente
- Mantenha o estilo uniforme em todo o projeto
- Mantenha um único idioma (preferencialmente em inglês)
- **Bom senso, bom senso, e mais bom senso**

Ferramentas

Diversas ferramentas foram desenvolvidas para auxiliar a manutenção, padronização e documentação de códigos.

Artistic Style

Um formatador automático livre, rápido e pequeno para códigos-fonte em C, C++, C++/CLI, C# e Java. [6]

[6] http://astyle.sourceforge.net/

Exemplo

Original

```
void Inverter(Pilha *P) {
elem x;
int erro,i=-1,j;
elem v[TamPilha];
while (!IsEmpty(P)){Pop(P,&x,&erro);i++;v[i]=x;}
for (j=0;j<=i;j++)
Push(P,&v[j],&erro);
}
```

Exemplo

--style=gnu

```
void Inverter(Pilha *P)
{
    elem x;
    int erro,i=-1,j;
    elem v[TamPilha];
    while (!IsEmpty(P))
        {
            Pop(P,&x,&erro);
            i++;
            v[i]=x;
        }
    for (j=0; j<=i; j++)
        Push(P,&v[j],&erro);
}
```

Exemplo

--style=linux

```
void Inverter(Pilha *P)
{
    elem x;
    int erro,i=-1,j;
    elem v[TamPilha];
    while (!IsEmpty(P)) {
        Pop(P,&x,&erro);
        i++;
        v[i]=x;
    }
    for (j=0; j<=i; j++)
        Push(P,&v[j],&erro);
}
```

Doxygen

Doxygen é a ferramenta padrão para gerar documentação de códigos-fonte anotados escritos em C++. [7]

[7] http://doxygen.org

Makefile

Exemplo

```
CC = gcc
CFLAGS = -I.
DEPS = hellomake.h
OBJ = hellomake.o hellofunc.o

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

hellomake: $(OBJ)
	gcc -o $@ $^ $(CFLAGS)
```

Argumentos da main

Escreva um programa que imprima na tela a soma dos argumentos passados para o programa.

Operação Destroy

```
/**
 * @brief Libera a memória alocada para a pilha.
 *
 * Após liberar a memória alocada no endereço apontado
      pelo ponteiro
 * apontado por \p stack, atribui NULL em valor de \p
     stack.
 *
 * @param[in,out] stack Ponteiro de ponteiro para
     estrutura a ser desalocada.
 */
void Destroy(Stack **stack)
{
    /* ... */
}
```