Leiaute

"First, beautiful software ought to be visually beautiful —
the written code I mean — and almost a pleasure to read."
— Don Sherwood Olson, engenheiro de software

"Let us change our traditional attitude
to the construction of programs.
Instead of imagining that our main task is
to instruct a computer what to do,
let us imagine that our main task is
to explain to human beings
what we want a computer to do."
— Donald E. Knuth, *Literate Programming*

"Any fool can write code that a computer can understand.
Good programmers write code
that *humans* can understand."
— Martin Fowler,
Refactoring: Improving the Design of Existing Code

Programas precisam ser entendidos não só por computadores, mas também (e principalmente) por seres humanos. Por isso, o leiaute (= *layout*) de um programa, ou seja, a disposição do programa na folha de papel e na tela do monitor, é tão importante. Os dois elementos fundamentais do leiaute são

- a *indentação*, que é o recuo das linhas em relação à margem esquerda da página, e
- os *espaços* entre as palavras (e outros símbolos) de uma linha.

O leiaute é, portanto, a *administração dos espaços em branco* no texto de um programa. Ele segue essencialmente os mesmos princípios que o leiaute de qualquer outro tipo de texto.

É fácil habituar-se a produzir um bom leiaute. Com um pouco de prática, os dedos do programador, dançando sobre o teclado, passarão a fazer a coisa certa de maneira autônoma, deixando a mente do programador livre para cuidar de assuntos mais importantes.

Um bom leiaute

Eis uma amostra do melhor leiaute que conheço. Ele está de acordo (ou quase) com o leiaute sugerido por Making The Best Use of C no *GNU Coding Standards* e com o *Elements of Programming Style* de Kernighan e Plauger.

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

```
    }
    return n;
}
```

Veja como é fácil conferir o casamento de "{" com "}". Também é fácil remover linhas e inserir novas linhas com um editor de textos, quando isso for necessário.

Tenha compaixão por seus leitores e *não use fonte de espaçamento variável* (a fonte de texto normal) para escrever programas. O resultado é ruim pois muitos caracteres (como "i", "-", e o espaço em branco) são estreitos demais:

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

Um leiaute compacto

O leiaute abaixo ocupa menos espaço vertical que o anterior. Ele é bom para escrever programas com lápis-e-papel (mas não tão bom para fazer alterações com o auxílio de um editor de textos). Indentação correta é essencial nesse leiaute.

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

Mau exemplo

O que você acha do leiaute no seguinte exemplo?

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

```
            }
        }
        return n;
    }
```

Este leiaute é *péssimo* porque deixa espaços onde não deve e engole os espaços onde eles são necessários. Lembre-se: os espaços no código são tão importantes quanto as pausas na música!

Infelizmente, alguns bons programadores e autores cometem esse tipo de atrocidade gratuita. Espero que você não queira imitar esses maus exemplos.

Sugestões sobre leiaute

Quer uma sugestão? Use as mesmas regras de leiaute que livros, revistas e jornais seguem ao publicar contos, artigos e reportagens:

- 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 cada parêntese direito;
- deixe um espaço antes, mas não depois, de cada parêntese esquerdo.

A expressão "`while(j < n)`" tem o mesmo sabor que "`enquantoj for menor que n`". Portanto,

- jamais escreva `while(j < n)` no lugar de `while (j < n) ;`
- jamais escreva `else{` no lugar de `else { ;`
- não escreva `for (i=1;i<n;++i)` no lugar de `for (i = 1; i < n; ++i) ;`
- etc.

Há três exceções notórias às regras acima: escreva

- `x->prox` e não `x -> prox`,
- `x[i]` e não `x [i]`,
- `x++` e não `x ++`.

Além disso, é usual não deixar espaços antes nem depois dos operadores de multiplicação e divisão. Assim, é usual escrever `x*y` e `x/y` em lugar de `x * y` e `x / y` respectivamente.

Recomendo deixar um espaço entre o nome de uma função e o parêntese esquerdo seguinte, ainda que isso contrarie a notação tradicional da matemática. Por exemplo, recomendo escrever

```
funcao (argumento1, argumento2)
```

em vez de `funcao(argumento1, argumento2)`, pois o primeiro leiaute é mais fácil de ler que o segundo.

Exercícios

1. Qual das duas linhas abaixo tem leiaute mais civilizado?

```
para j variando de 1 até n de 1 em 1, faça
paraj variando de1 até n de 1em1,faça
```

2. Qual das duas linhas abaixo tem leiaute mais decente?

```
for(j=0;j<n;++j){
for (j = 0; j < n; ++j) {
```

3. Qual das duas linhas abaixo tem melhor leiaute?

```
for (j = 0; j < n; ++j) {
for(j = 0; j < n; ++j) {
```

4. Qual das duas linhas abaixo tem leiaute mais civilizado?

```
for (j = 0;j < n; ++j){
for (j = 0; j < n; ++j) {
```

5. Reescreva o trecho de programa abaixo com bom leiaute.

```
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;}
```

6. Reescreva a seguinte linha de código com leiaute decente. Depois, escreva uma versão mais simples da linha.

```
for(i = 23; i --> 0;)
```

7. Corrija os erros de leiaute no texto abaixo.

```
A computação não é a ciência dos computadores,da
mesma formaque a astronomia não  é a ciência dos
telescópios.
```

8. Corrija os erros de leiaute do texto abaixo.

```
Em 1959 e nas décadas seguintes   nenhum programador
Cobol poderia imaginar que os programas de computador
que estava criando ainda estariam em operação no fim
do século.Poucos se lembram hoje que há menos de
quinze anos os primeiros PCs possuíam apenas 64Kbytes
de memória.Como o custo dos dispositivos de
armazenamento era alto e a quantidade de memória
disponível era pequena,usavam-se muitos truques para
economizar esse recurso. Um deles foi o uso de dois
dígitos   para representar o ano:armazenava-se(por
exemplo ) 85 em vez de 1985.com a chegada do ano
2000,essa codificação"econômica"transformou-se em
um erro em potencial .
```

SOLUÇÃO:

```
Em 1959 e nas décadas seguintes nenhum programador
Cobol poderia imaginar que os programas de computador
que estava criando ainda estariam em operação no fim
do século. Poucos se lembram hoje que há menos de
quinze anos os primeiros PCs possuíam apenas 64Kbytes
de memória. Como o custo dos dispositivos de
armazenamento era alto e a quantidade de memória
disponível era pequena, usavam-se muitos truques para
economizar esse recurso. Um deles foi o uso de dois
dígitos para representar o ano: armazenava-se (por
exemplo) 85 em vez de 1985. Com a chegada do ano
2000, essa codificação "econômica" transformou-se em
um erro em potencial.
```

9. Reescreva o trecho de programa abaixo usando leiaute decente.

```
esq= 0; dir=N-1;
i=(esq+dir)/2;     /*indice do "meio"de r[]*/
while(esq <= dir && r[i] != x){
   if(r[i]<x) esq = i+1;
   else dir = i-1;  /* novo indice do "meio"de r[] */
   i= (esq + dir)/2;
   }
```

SOLUÇÃO:

```
esq = 0; dir = N - 1;
i = (esq + dir)/2;    /* índice do "meio" de r[] */
while (esq <= dir && r[i] != x) {
   if (r[i] < x) esq = i + 1;
   else dir = i - 1;  /* novo índice do "meio" de r[] */
   i = (esq + dir)/2;
}
```

Recomendações finais

- Não use linhas longas, nem em código, nem nos comentários. Linhas longas são difíceis de ler e em geral não cabem numa folha de papel impressa. Sugiro nunca passar de setenta-e-tantos caracteres por linha.
- Jamais use tabulação (tecla *tab*) no texto do seu programa. Para fazer a indentação de uma linha, use o número apropriado de espaços.
- Alguns editores de texto trocam, automaticamente, longas sequências de espaços por tabulação (*tab*). A substituição é imperceptível pois caracteres de tabulação são invisíveis, mas tende a estragar o leiaute do programa impresso. Configure o seu editor de texto de modo a desligar essa substituição automática.
- Evite caracteres desnecessários (exceto espaços em branco) no código. Em particular, evite os parênteses que as regras de precedência entre operadores tornam redundantes.
- Se necessário, você pode recorrer a algum programa que corrige os defeitos de layout do seu programa. Há vários programas desse tipo. Veja, por exemplo, o `astyle`.

Veja o verbete *Programming Style* na Wikipedia

Atualizado em 2016-08-14
http://www.ime.usp.br/~pf/algoritmos/
Paulo Feofiloff
DCC-IME-USP