ERROS MAIS COMUNS COMETIDOS EM PROGRAMAS E ALGORITMOS

Compilação dos erros mais comuns cometidos por estudantes de Introdução a Algoritmos

Marcos Portnoi

Edição 19.4.2007

Universidade Salvador – UNIFACS

2004

Sumário

ERROS MAIS COMUNS EM ALGORITMOS

Marcos Portnoi

Última atualização: 19/4/2007

Este documento compila alguns erros mais comuns cometidos na confecção de algoritmos pelos estudantes de Introdução a Algoritmos. A linguagem utilizada para os exemplos é o Português Estruturado, conforme a sintaxe do programa IC (Interpretador C, da Universidade São Francisco[1]).

1. Não declarar variáveis utilizadas dentro do programa.

No programa abaixo, a variável "soma", utilizada em um cálculo, não foi declarada.

ERRADO	CERTO

principal(){
float parc, total;
escreva("Digite a parcela:");
leia(parc);
soma=parc+50;
...
	principal(){
	float parc, total, **soma**;
	escreva("Digite a parcela:");
	leia(parc);
	soma=parc+50;
	...

2. Utilizar uma variável de um modo incompatível com o seu tipo declarado.

No programa abaixo, a variável "opcao" foi declarada como INT, mas está sendo comparada com letras, como se fosse uma variável CHAR.

ERRADO

```
principal(){
   int opcao, x, i; ←
   escreva("Digite um numero:");
   leia(x);
   escreva("Digite um operador (s = somar, b = subtrair, m = multiplicar):");
   leia(opcao);
   se (opcao=='s') {
   ...
```

CERTO

```
principal(){
   int x, i;
```

[1] http://www.usf.br

```
char opcao; ←
escreva("Digite um numero:");
leia(x);
escreva("Digite um operador (s = somar, b = subtrair, m = multiplicar):");
leia(opcao);
se (opcao=='s') {
...
```

3. Fazer comparações com uma variável do tipo CHAR e um caractere e não colocar este caractere entre aspas simples.

As comparações entre variáveis tipo CHAR e caracteres devem trazer os caracteres entre aspas simples. Caso contrário, o interpretador considerará o caractere como uma variável ou um erro de sintaxe. Tipicamente, o que se quer é testar se o *conteúdo* da variável tipo CHAR é igual a alguma *letra* ou *símbolo*. Omitindo as aspas simples, o interpretador considera as letras como *variáveis*.

ERRADO	CERTO
... char esc; ... se (esc != s && esc != b) { ...	... char esc; ... se (esc != 's' && esc != 'b') { ...

No exemplo errado, o interpretador entende que a variável *esc* está sendo comparada com as variáveis *s* e *b*, pois as aspas simples não foram corretamente incluídas.

4. Usar atribuições no lugar de comparações ou vice-versa.

O comando de atribuição de valor de expressões a variáveis é o sinal de igual (=). O comando de comparação (o teste "se o conteúdo da variável é igual à expressão") é feito com dois sinais de igual (==).

ERRADO	CERTO

...
int i==0;
float calc==0;
...
i==i+1;
enquanto (calc=0) {
...
	...
	int i=0;
	float calc=0;
	...
	i=i+1;
	enquanto (calc==0) {
	...

5. Não terminar os comandos corretamente com ";" ou chaves (}).

Todo comando não seguido por chaves ({}) deve ser terminado com **ponto-e-vírgula** (;). Os comandos acompanhados de chaves **não** devem ser terminados pelo ponto-e-vírgula.

ERRADO	CERTO
... i++ custo=custo+valor enquanto(custo<1000); ← { escreva(“<1000”); ...	... i++**;** ← custo=custo+valor**;** enquanto(custo<1000) { escreva(“<1000”); ...

6. Uso incorreto das chaves em comandos PARA, SE, SENÃO, ENQUANTO, FAÇA-ENQUANTO, etc.

Os comandos acima executam um grupo de comandos envolvidos entre chaves, se uma determinada condição for verdadeira. Se as chaves não forem utilizadas para agrupar os comandos, ou se um ponto-e-vírgula for colocado erroneamente antes das chaves, os comandos não funcionarão corretamente.

ERRADO	CERTO

se (cond==1)**;** ←
{
leia(valor);
}
...
enquanto(tecla!=’c’)
leia(fat);
x=x+fat;
leia(tecla);
...
faça{
escreva(x);
x++;
} enquanto (x<20)
escreva(“valor final = “, x);
...
	se (cond==1)
	{
	leia(valor);
	}
	...
	enquanto(tecla!=’c’) **{** ←
	leia(fat);
	x=x+fat;
	leia(tecla);
	} ←
	...
	faça{
	escreva(x);
	x++;
	} enquanto (x<20)**;** ←
	escreva(“valor final = “, x);
	...

7. Operar variáveis tipo CHAR com números.

Variáveis tipo CHAR devem ser operadas com caracteres (estes sempre entre aspas simples). Em algumas condições, as variáveis tipo CHAR podem ser operadas com o valor numérico do código ASCII do caractere respectivo. Operações matemáticas com variáveis tipo CHAR não fazem sentido, a não ser que o programador queira explicitamente trabalhar com o valor numérico do código ASCII do conteúdo destas variáveis.

8. Operar variáveis tipo CHAR com palavras.

Variáveis tipo CHAR trabalham com um único caractere como conteúdo, e nunca palavras ou sentenças.

ERRADO	CERTO

char senha='1234', entrada;
...
enquanto (entrada!='preto') {
...
	char senha='1', entrada;
	...
	enquanto (entrada!='p') {
	...

9. Declarar uma variável mais de uma vez, ou declarar os valores que uma variável vai receber durante o programa.

Uma variável é um espaço de memória que o programa usa para armazenar valores de acordo com o tipo declarado da variável. A declaração, no início do programa, resulta na reserva deste espaço na memória do computador. Assim sendo, uma variável deve ser declarada uma única vez.

A variável pode mudar de conteúdo várias vezes durante a execução do algoritmo. Se for lida do teclado, o usuário pode digitar virtualmente qualquer coisa para ela. O programa deve tratar os possíveis diferentes conteúdos da variável através de estruturas de teste. O algoritmo deve possuir a inteligência de saber manusear estes valores. Não é possível declarar de antemão que valores uma variável vai poder assumir na execução do programa.

ERRADO	CERTO
char opcao='p', opcao='s', opcao='q'; leia(opcao); se (opcao=='p') {...} se (opcao=='s') {...} se (opcao=='q') {...} ...	char opcao; leia(opcao); se (opcao=='p') {...} se (opcao=='s') {...} se (opcao=='q') {...} ...

10. Declarar valores soltos.

Somente variáveis devem ser declaradas, e nunca valores soltos. Valores numéricos e caracteres não precisam ser declarados, pois eles são valores, e não variáveis.

ERRADO	CERTO
int a, 2, 3, 5; char tipo, 'j', 'k', 'z';	int a; char tipo;

11. Uso incorreto do comando LEIA.

Abaixo estão exemplos do uso incorreto do comando LEIA, que serve para ler valores do teclado para as variáveis. Dentro dos parênteses de LEIA, somente variáveis devem ser usadas, nunca textos, ou comparações, ou atribuições. Lembrar que LEIA pode ler o valor de várias variáveis (e de diferentes tipos), como por exemplo *leia(a,b,c)*. O usuário digitará os três valores pedidos separados pela tecla ENTER.

ERRADO	CERTO

leia("opcao");
...
escreva("Digite a, b ou c:");
leia(let, a, b, c);
...
escreva("Digite um valor entre 1 e 10:");
leia(a<10 && a>1);
	leia(opcao);
	...
	escreva("Digite a, b ou c:");
	leia(let);
	...
	escreva("Digite um valor entre 1 e 10:");
	leia(a);

12. Uso incorreto de texto e variáveis mescladas no comando ESCREVA.

O comando ESCREVA serve para escrever textos, sempre entre aspas duplas, e também o conteúdo de variáveis. Textos podem ser entremeados com o conteúdo de variáveis, sempre separados por vírgulas.

ERRADO	CERTO
escreva("O conteudo de A eh:"a" O de B eh:"b". E o de C eh:"c);	escreva("O conteudo de A eh:"**,a,**". O de B eh:"**,b,**". E o de C eh:"**,c**);

13. Grafia incorreta dos comandos SENÃO e FAÇA.

Os comandos SENÃO e FAÇA devem ser escritos exatamente assim, com acentos e cedilha, e nunca SENAO e FACA.

14. Uso incorreto de comparações sem os operadores lógicos apropriados, e não aos pares.

As comparações, usadas em comandos SE, ENQUANTO, etc., devem ser feitas aos pares, e nunca como são comumente escritas matematicamente.

ERRADO	CERTO
se (2<x<10) {...} se (opcao != 'j' && 'k' && 'x') {...}	se (x>2 && x<10) {...} se (opcao != 'j' && opcao != 'k' && opcao != 'x') {...}

15. Uso de acentuação, espaço, ou caracteres especiais em nomes de variáveis.

Nomes de variáveis não devem conter quaisquer acentuação, ou cedilha, ou caracteres especiais como !, &, #, (, },@, -, +, /, ?, etc., ou espaços. Para separar palavras em nomes de variáveis, use o "sublinhado" (_).

ERRADO	CERTO
double prestação, soma total, tot a; char nome-aluno, opção;	double prestacao, soma_total, tota; char nome_aluno, opcao;

16. Uso de comparação (==) no lugar de atribuição (=) no comando PARA.

O comando PARA é formado de três campos: (a) valor ou condição inicial, estabelecido por uma atribuição à variável de controle do laço; (b) condição de repetição; (c) incremento ou decremento.

ERRADO	CERTO
para (i**==**1; i<100; i=i+2);	para (i**=**1; i<100; i=i+2);

17. Uso de variáveis incorretas no comando PARA.

Apesar de que a inclusão de todos os três campos no comando PARA não serem obrigatórias para programação avançada, estes três campos devem mencionar a mesma variável contadora de controle do PARA, à esquerda dos operadores.

ERRADO	CERTO
para (i=1; **x**<100; **j**=**j**+2);	para (i=1; **i**<100; **i**=**i**+2);

18. Não declarar a variável contadora do comando PARA.

A variável de contagem do comando PARA, como toda variável, deve ser declarada no início do programa como numérica (INT, FLOAT ou DOUBLE, a depender da necessidade).

19. Alterar o valor da variável contadora do PARA dentro do laço de repetição.

Em programação avançada, alterar o valor da variável de controle do comando PARA é permitido, mas só se deve fazê-lo se não houver uma outra opção mais elegante em termos de lógica e se se souber exatamente o que se está fazendo.

ERRADO	CERTO

principal(){
int soma, i;
...
para (i=1; i<soma; i++)
{
escreva("N. etiqueta?");
leia(i); ←
...
	principal(){
	int soma, i, **n**;
	...
	para (i=1; i<soma; i++)
	{
	escreva("N. etiqueta?");
	leia(**n**); ←
	...

20. Laços de repetição com ENQUANTO que nunca se repetem (e nunca se executam).

Laços de repetição feitos com o comando ENQUANTO, que tenham como controle o valor de uma variável, devem garantir que o valor inicial desta variável permite a execução do laço (se for desejado que o laço se repita ao menos uma vez). Lembrar que na linguagem C, principalmente, o valor de uma variável assim que declarada é uma incógnita. Ver erro seguinte.

ERRADO	CERTO

principal(){
char tipo='s'; ←
...
enquanto(tipo!='s') ←
{
escreva("N. etiqueta?");
...
	principal(){
	char tipo='**a**'; ←
	...
	enquanto(tipo!='s') ←
	{
	escreva("N. etiqueta?");
	...

21. Não definir valores iniciais pertinentes para variáveis, em especial acumuladores e contadores.

Na linguagem C, principalmente, o valor de uma variável assim que declarada é uma incógnita. Assim, variáveis que não tenham seus valores digitados pelo usuário devem tê-los definidos deterministicamente no programa, a fim de evitar erros por causa dos valores aleatórios que as variáveis assumem quando simplesmente declaradas. O Interpretador C, de fato, inicializa todas as variáveis declaradas, de qualquer tipo, para o valor zero, mas não se deve confiar nesta prática.

ERRADO	CERTO

principal(){
int soma; ←
...
leia(numero);
soma=soma+numero; ←
...
	principal(){
	int **soma=0**; ←
	...
	leia(numero);
	soma=soma+numero; ←
	...