Ordenação por Inserção

Paulo Feofiloff

São Paulo, 2 de março de 2001

Sumário

1. Introdução

Esta é uma demonstração do sistema `CWEB` Knuth e Levy (veja `http://www.ime.usp.br/~pf/CWEB/`). Um programa `CWEB`, como este que você está lendo, é uma espécie de jogo de armar que produz um programa C. Ele contém *(1)* vários "blocos" de código C e *(2)* instruções sobre a maneira de encaixar esses blocos para construir o programa.

O leitor pode usar este pequeno exemplo como ponto de partida para escrever os seus próprios programas `CWEB`.

2. O que o programa faz

Este programa coloca em ordem crescente qualquer seqüência de números inteiros. A seqüência é dada em um arquivo, e o resultado é gravado em outro arquivo. O usuário deve indicar os nomes dos arquivos (o segundo pode ser igual ao primeiro) na linha de comando. Eis um exemplo de linha de comando que ativa o programa:

```
isort dados.txt resultados.txt
```

3. Referências

O programa usa o bem-conhecido algoritmo de ordenação-por-inserção (= insertion sort). Minha inspiração é o programa 6.3 (p.264) do livro *Algorithms in C* (3rd. ed., Addison-Wesley, 1998) de R. Sedgewick. Pode ser um exagero dar referências para um programa tão simples, mas em geral é importante citar suas fontes.

4. Insertion sort

Ao contrário de um programa C crú, um programa CWEB pode ir direto ao assunto central e deixar os assuntos burocráticos e administrativos (entrada, saída, include, define, declaração de variáveis, etc.) para mais tarde.

A função *isort* rearranja os elementos de um vetor $a[l \mathinner{..} r]$ de modo que ele fique em ordem crescente, ou seja, de modo que tenhamos $a[l] \le a[l+1] \le \cdots \le a[r]$.

⟨ Função *isort* 4 ⟩ ≡

```
void isort(int a[], int l, int r)
{
  int i;
  for (i = l + 1; i ≤ r; i++)
    ⟨ Execute uma iteração 5 ⟩
}
```

Este código é usado no bloco 8.

5. Para entender como o algoritmo funciona, basta entender que no início de cada iteração o subvetor

$$a[l \mathinner{..} i-1] \text{ está em ordem crescente.}$$

Esta propriedade é trivialmente verdadeira quando $i \equiv l + 1$. Quando i atingir o valor $r + 1$, o vetor $a[l \mathinner{..} r]$ estará ordenado!

⟨ Execute uma iteração 5 ⟩ ≡

```
{
  int j = i, v = a[i];
  while (j > l ∧ v < a[j − 1]) {
    a[j] = a[j − 1];
    j−−;
  }
  a[j] = v;
}
```

Este código é usado no bloco 4.

6. O símbolo ∧ que aparece em "**while** $(j > l \land v < a[j-1])$" representa o operador lógico &&. No programa C gerado a partir deste programa CWEB teremos "while (j > l && v < a[j-1])".

7. Geração do header file isort.h

Vou gerar um header file isort.h, que servirá de interface para outros programas que porventura queiram usar minha função *isort*.

⟨ isort.h 7 ⟩ ≡

```
extern void isort(int[], int, int);
```

8. Estrutura geral do programa e função main

A estrutura do programa isort.c gerado por este programa CWEB é muito simples. Como freqüentemente acontece, os assuntos administrativos (entrada de dados, gravação de resultados, etc.) consomem mais espaço que a parte "importante" do programa.

```
⟨ Inclusão de header files 16 ⟩
⟨ Função isort 4 ⟩
int main(int numargs, char *arg[])
{
  ⟨ Variáveis locais da main 13 ⟩
  ⟨ Processamento da linha de comando 9 ⟩
  ⟨ Entrada de n e a[0 .. n − 1] 10 ⟩
  isort(a, 0, n − 1);
  ⟨ Gravação dos resultados 14 ⟩
  return 0;
}
```

9. É preciso verificar se o usuário digitou o número correto de argumentos na linha de comando.

```
⟨ Processamento da linha de comando 9 ⟩ ≡
  if (numargs ≠ 3) {
    fprintf(stderr, "\nUso:␣isort␣<nomearq1>␣<nomearq2>");
    fprintf(stderr, "\n␣␣␣␣␣<nomearq1>␣é␣o␣nome␣do␣arquivo␣de␣dados");
    fprintf(stderr, "\n␣␣␣␣␣<nomearq2>␣é␣o␣nome␣do␣arquivo␣de␣resultados\n");
    return −1;
  }
```

Este código é usado no bloco 8.

```
10. ⟨ Entrada de n e a[0 .. n − 1] 10 ⟩ ≡
  arq = fopen(arg[1], "r");
  if (arq ≡ Λ) {
    fprintf(stderr, "\nN~ao␣encontrei␣arquivo␣%s\n", arg[1]);
    return −1;
  }
  ⟨ Leitura de arq 12 ⟩;
  fclose(arq);
```

Este código é usado no bloco 8.

11. O símbolo Λ que aparece acima em "**if** $(arq \equiv \Lambda)$" é a representação de NULL. No programa C gerado a partir deste programa CWEBteremos "if (arq == NULL)".

12. O processo de leitura preenche o vetor $a[0 \mathrel{..} n-1]$ e determina o valor de n. A leitura termina quando chega ao fim do arquivo ou quando encontra algo incompatível com o formato %d. Convém lembrar que a expressão *fscanf*$(arq, f, \ldots)$ tem valor igual ao número de itens que foi possível extrair de *arq* de acordo com o formato f.

```
⟨ Leitura de arq 12 ⟩ ≡
  n = 0;
  while (fscanf(arq, "%d", &a[n]) ≡ 1) {
    ++n;
    if (n > MAX) {
      fprintf(stderr, "\nN~ao␣sei␣lidar␣com␣mais␣que␣%d␣números\n", MAX);
      return −1;
```

```
    }
  }
```
Este código é usado no bloco 10.

13. Meu programa está preparado para lidar com no máximo `MAX` números. Convém que `MAX` não passe de algumas centenas, pois o algoritmo de ordenação-por-inserção é lento. Vou reservar espaço para `MAX` + 1 números no vetor a, pois o código do bloco anterior parece exigir essa "margem de segurança".

```
#define MAX 1000
⟨ Variáveis locais da main 13 ⟩ ≡
  FILE *arq;
  int n, a[MAX + 1];
```
Veja também bloco 15.

Este código é usado no bloco 8.

14. A seqüência ordenada será gravada no arquivo $arg[2]$.

```
⟨ Gravação dos resultados 14 ⟩ ≡
  arq = fopen(arg[2], "w");
  if (arq ≡ Λ) {
    fprintf(stderr, "\nSocorro!␣N~ao␣consigo␣abrir␣arquivo␣%s\n", arg[2]);
    return −1;
  }
  for (i = 0; i < n; ++i) fprintf(arq, "%d\n", a[i]);
  fclose(arq);
```
Este código é usado no bloco 8.

15. A seção acima usou mais uma variável local.

```
⟨ Variáveis locais da main 13 ⟩ +≡
  int i;
```

16. Inclusão de header files

O header file stdio.h contém os protótipos das funções de entrada e saída e a definição do arquivo de erros *stderr*.

```
⟨ Inclusão de header files 16 ⟩ ≡
#include <stdio.h>
```
Este código é usado no bloco 8.

17. Agora que terminei, estou vendo que o programa ficou muito fragmentado (grande número de pequenos blocos) e excessivamente comentado. Mas vou deixar as coisas assim mesmo; afinal, esse programa é apenas um exemplo despretensioso!

Índice Remissivo

Para cada identificador, este índice cita as seções onde o identificador é definido (números grifados) ou usado. No caso de identificadores com uma só letra, como v, somente as seções que contêm as definições são citadas. É claro que este índice é gerado automaticamente.

Lista de Refinamentos