1. Introducão. Esta é uma demonstração do sistema `CWEB` (veja `http://www.ime.usp.br/~pf/CWEB/`) de Knuth e Levy. 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 calcula a média e o desvio padrão de n números inteiros dados. Os números podem ser digitados no teclado, lidos de um arquivo, ou gerados aleatoriamente pelo programa. O usuário deve indicar uma dessas opções na linha de comando:

`-t` para entrada pelo teclado
`-f`⟨*nomearq*⟩ se os dados estão no arquivo *nomearq*
`-r`⟨n⟩ se o programa deve gerar n números aleatoriamente.

No caso da opção `-t`, a digitação de dados termina com qualquer caracter não-numérico. Eis um exemplo de linha de comando que ativa o programa:

```
mdp -fmeusdados.txt
```

3. Referências: Este programa é uma versão modificada do Program 3.2 do livro *Algorithms in C* (3rd. ed., Addison-Wesley, 1998, p.75) de R. Sedgewick. É um tanto ridículo dar referências para um programa tão simples, mas em geral é importante citar suas fontes.

4. Média e desvio padrão. Ao contrário de um programa C cru, 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 *média_e_dp* recebe um vetor a de números inteiros e um inteiro $n \geq 1$ e determina a média e o desvio padrão de $a[0 \,..\, n-1]$. O valor da média é gravado no endereço *em* e o valor do desvio padrão é gravado em *ed*.

```
⟨ Média e desvio padrão 4 ⟩ ≡
  void média_e_dp(int a[], int n, float *em, float *ed)
  {
    ⟨ Variáveis auxiliares 7 ⟩
    ⟨ Faça *em igual à média 5 ⟩
    ⟨ Faça *ed igual ao desvio padrão 6 ⟩
  }
```

Este código é usado na seção 9.

5. A média dos elementos de um vetor $a[0 \,..\, n-1]$ é o número $m = \frac{1}{n}\sum_{i=0}^{n-1} a[i]$; é evidente que isso só faz sentido se $n \geq 1$. O cálculo dessa expressão está sujeito a overflow: de acordo com o meu computador, o valor da expressão para dez números iguais a 10^9 é *diferente de* 10^9 (mesmo que todos os números sejam **float**). Para retardar um pouco esse efeito, é melhor usar a expressão $\sum(a[i]/n)$.

```
⟨ Faça *em igual à média 5 ⟩ ≡
  m = 0.0;
  for (i = 0; i < n; ++i)
    m += ((float) a[i])/n;      ▷ o molde (float) é necessário?
  *em = m;
```

Este código é usado na seção 4.

6. O desvio padrão dos elementos de um vetor $a[0 \,..\, n-1]$ é a raiz quadrada de

$$\tfrac{1}{n}\sum_{i=0}^{n-1}(a[i]-m)^2,$$

onde m é a média; é claro que isso só faz sentido se $n \geq 1$. Como no bloco anterior, o cálculo desta expressão está sujeito a overflow. Por exemplo, se temos cinco números iguais a $+10^5$ e outros cinco iguais a -10^5, o valor da expressão é `NaN` (not-a-number) no meu computador. Na tentativa de retardar esse efeito, usarei a expressão $\sum((a[i]-m)^2/n)$.

```
⟨ Faça *ed igual ao desvio padrão 6 ⟩ ≡
  v = 0.0;
  for (i = 0; i < n; ++i) {
    d = a[i] - m;
    v += d * d/n;
  }
  *ed = sqrt(v);      ▷ sqrt(v) ≡ √v
```

Este código é usado na seção 4.

7.
```
⟨ Variáveis auxiliares 7 ⟩ ≡
  float m, v;
  float d;
  int i;
```

Este código é usado na seção 4.

8. Geracão do header file mdp.h. Vamos gerar um header file `mdp.h`, que servirá de interface para outros programas que porventura queiram usar nossa função *média_e_dp*. No arquivo `mdp.w`, a ordem para gerar um arquivo `mdp.h` é indicada por "`@(mdp.h@>=`".

⟨ `mdp.h` 8 ⟩ ≡
extern void *média_e_dp*(**int**[], **int**, **float** ∗, **float** ∗);

9. Estrutura geral do programa. A estrutura do programa `mdp.c` gerado por este programa `CWEB` será muito simples: além da função *média_e_dp* que é a alma do programa, teremos uma função *main* que cuidará da administração (processamento da linha de comando, leitura de dados, impressão de resultados). Como freqüentemente acontece, a administração consome mais espaço que a alma.

```
⟨ Inclusão de arquivos externos 23 ⟩
⟨ Média e desvio padrão 4 ⟩
void main(int numargs, char *arg[])
{
  ⟨ Variáveis locais da main 12 ⟩
  ⟨ Processamento da linha de comando 10 ⟩
  ⟨ Leitura ou geração dos dados 13 ⟩
  ⟨ Cálculo da média e desvio padrão 20 ⟩
  ⟨ Impressão dos resultados 22 ⟩
}
```

10. Linha de comando. O processamento da linha de comando usa as funções de biblioteca *strcmp* e *sscanf*. A expressão *strcmp*(a, b) vale 0 se e só se as strings a e b são iguais e a expressão *sscanf*$(s, f, \ldots)$ tem valor igual ao número de itens que foi possível extrair da string s de acordo com o formato f.

```
⟨ Processamento da linha de comando 10 ⟩ ≡
  if (numargs ≠ 2) ⟨ Instruções de uso 11 ⟩;
  teclado = arquivo = aleatório = 0;
  if (strcmp(arg[1], "-t") ≡ 0) teclado = 1;
  else if (sscanf(arg[1], "-f%s", nomearq) ≡ 1) arquivo = 1;
  else if (sscanf(arg[1], "-r%d", &n) ≡ 1) aleatório = 1;
  else ⟨ Instruções de uso 11 ⟩
```

Este código é citado na seção 12.

Este código é usado na seção 9.

11.

```
⟨ Instruções de uso 11 ⟩ ≡
  {
    fprintf(stderr, "\nUso: mdp {-t | -f<nomearq> | -r<n>}");
    fprintf(stderr, "\n      -t          teclado");
    fprintf(stderr, "\n      -f<nomearq>  lê arquivo nomearq");
    fprintf(stderr, "\n      -r<n>        gera n números aleatoriamente\n");
    return;
  }
```

Este código é usado na seção 10.

12. Como se vê na seção ⟨ Processamento da linha de comando 10 ⟩, a preferência do usuário é registrada em três variáveis booleanas: a variável *teclado* vale 1 se e só se o usuário tiver escolhido a opção `-t`; as variáveis *arquivo* e *aleatório* são definidas analogamente. No caso de *arquivo* ≡ 1, o nome do arquivo será guardado em *nomearq*; espero que esse nome não tenha mais que 80 caracteres ...

```
⟨ Variáveis locais da main 12 ⟩ ≡
  int teclado, arquivo, aleatório;
  char nomearq[81];
```

Veja também as seções 18 e 21.

Este código é usado na seção 9.

13. Entrada.

```
⟨ Leitura ou geração dos dados 13 ⟩ ≡
  if (teclado) {
    fprintf(stdout, "\nDigite␣os␣números");
    fprintf(stdout, "\n(para␣terminar,␣digite␣caracter␣não-numérico):\n");
    arq = stdin;
    ⟨ Leitura de arq 17 ⟩
  }
```

Veja também as seções 14 e 15.

Este código é usado na seção 9.

14.

```
⟨ Leitura ou geração dos dados 13 ⟩ +≡
  else
    if (arquivo) {
      arq = fopen(nomearq, "r");
      if (arq ≡ Λ) {
        fprintf(stderr, "\nNão␣encontrei␣arquivo␣%s\n", nomearq);
        return;
      }
      ⟨ Leitura de arq 17 ⟩;
      fclose(arq);
    }
```

15.

```
⟨ Leitura ou geração dos dados 13 ⟩ +≡
  else if (aleatório) ⟨ Gera números aleatórios 19 ⟩
```

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

17. Quer *arq* esteja em um disco magnético, quer seja o teclado, o algoritmo de leitura é o mesmo. A leitura termina ao chegar ao fim do arquivo ou ao encontrar algo incompatível com o formato `%d`. (Por um instante, tive dúvidas: "**if** ($n >$ `MAX`)" ou "**if** ($n \geq$ `MAX`)"?)

```
⟨ Leitura de arq 17 ⟩ ≡
  n = 0;
  while (fscanf(arq, "%d", &a[n]) ≡ 1) {
    ++n;
    if (n > MAX) {
      fprintf(stderr, "\nExcedeu␣limite␣de␣%d␣números\n", MAX);
      return;
    }
  }
```

Este código é usado nas seções 13 e 14.

18. Nosso programa está preparado para lidar com no máximo `MAX` números (mas o vetor a terá espaço para `MAX` + 1 elementos, para evitar problemas no caso em que o usuário tenta fornecer mais que `MAX` números).

```
#define MAX 1000
```

⟨ Variáveis locais da *main* 12 ⟩ +≡

```
FILE *arq;
int n, a[MAX + 1];
```

19. A função de biblioteca *rand* gera aleatoriamente (ou melhor, pseudo-aleatoriamente) um número entre 0 e `RAND_MAX`. No meu computador, `RAND_MAX` vale $2^{15} - 1 \equiv 32767$.

⟨ Gera números aleatórios 19 ⟩ ≡

```
{
  int i;
  if (n > MAX) {
    fprintf(stderr, "\nExcedeu limite de %d números\n", MAX);
    return;
  }
  for (i = 0; i < n; ++i) {
    a[i] = rand( );
    fprintf(stdout, "\n%d", a[i]);
  }
  fprintf(stdout, "\n");
}
```

Este código é usado na seção 15.

20. Cálculo da média e do desvio padrão. Meus dados estão em $a[0 \,..\, n-1]$. Só faz sentido calcular média e desvio padrão se o vetor não estiver vazio, ou seja, se $n > 0$.

⟨ Cálculo da média e desvio padrão 20 ⟩ ≡

```
if (n ≡ 0) {
  fprintf(stderr, "\nNão há dados!\n");
  return;
}
média_e_dp(a, n, &m, &d);
```

Este código é usado na seção 9.

21. Não podemos nos esquecer de declarar as variáveis m e d.

⟨ Variáveis locais da *main* 12 ⟩ +≡

```
float m, d;
```

22. Saída. Os resultados serão impressos na tela do monitor:

⟨ Impressão dos resultados 22 ⟩ ≡

```
fprintf(stdout, "\n%d números dados", n);
fprintf(stdout, "\nmédia = %f", m);
fprintf(stdout, "\ndesvio padrão = %f", d);
fprintf(stdout, "\n");
```

Este código é usado na seção 9.

23. Inclusão de header files. Será necessário incluir quatro header files em nosso programa: o `stdio.h`, que contém os protótipos das funções de entrada e saída e as definições de *stdin*, *stdout* e *stderr*; o `stdlib.h`, que contém o protótipo da função de biblioteca *rand* e a definição da constante `RAND_MAX`; o `string.h`, que contém o protótipo da função de biblioteca *strcmp*; e `math.h`, que contém o protótipo da função *sqrt*.

Espero não me esquecer de usar a opção `-lm` ao compilar o programa C, para que a biblioteca que contém *sqrt* seja carregada.

⟨ Inclusão de arquivos externos 23 ⟩ ≡

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

Este código é usado na seção 9.

24. Agora que terminei, estou vendo que o programa ficou muito fragmentado (grande número de pequenas seções) e excessivamente comentado. Mas vou deixar as coisas assim mesmo; afinal, isso é apenas uma ilustração despretensiosa do `CWEB`!

25. Índice. 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.

⟨ Cálculo da média e desvio padrão 20 ⟩ Usado na seção 9.
⟨ Faça $*ed$ igual ao desvio padrão 6 ⟩ Usado na seção 4.
⟨ Faça $*em$ igual à média 5 ⟩ Usado na seção 4.
⟨ Gera números aleatórios 19 ⟩ Usado na seção 15.
⟨ Impressão dos resultados 22 ⟩ Usado na seção 9.
⟨ Inclusão de arquivos externos 23 ⟩ Usado na seção 9.
⟨ Instruções de uso 11 ⟩ Usado na seção 10.
⟨ Leitura de *arq* 17 ⟩ Usado nas seções 13 e 14.
⟨ Leitura ou geração dos dados 13, 14, 15 ⟩ Usado na seção 9.
⟨ Média e desvio padrão 4 ⟩ Usado na seção 9.
⟨ Processamento da linha de comando 10 ⟩ Citado na seção 12. Usado na seção 9.
⟨ Variáveis auxiliares 7 ⟩ Usado na seção 4.
⟨ Variáveis locais da *main* 12, 18, 21 ⟩ Usado na seção 9.
⟨ `mdp.h` 8 ⟩

MDP

Paulo Feofiloff
IME–USP, São Paulo, 6 de março de 2001

Este documento foi gerado pelo sistema `CWEB` de D. E. Knuth e S. Levy.