

1. Introdução. 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 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 *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 $m = \frac{1}{n} \sum a[i]$; 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

$$\frac{1}{n} \sum (a[i] - m)^2,$$

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. Geração 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_dp`. No arquivo `mdp.w`, a ordem para gerar um arquivo `mdp.h` é indicada por “@(`mdp.h`>=”.

(`mdp.h` 8) ≡

```
extern void média_dp(int[], int, float *, float *);
```


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* ≡ Λ)” 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* ≥ 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 *arg;
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 macro `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 desprezível 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.

a: 4, 18.

aleatório: 10, 12, 15.

arg: 9, 10.

arg: 13, 14, 16, 17, 18.

arquivo: 10, 12, 14.

d: 7, 21.

ed: 4, 6.

em: 4, 5.

fclose: 14.

fopen: 14.

fprintf: 11, 13, 14, 17, 19, 20, 22.

fscanf: 17.

i: 7, 19.

m: 7, 21.

main: 9.

MAX: 17, 18, 19.

média_dp: 4, 8, 9, 20.

n: 4, 18.

nomearg: 2, 10, 12, 14.

numargs: 9, 10.

rand: 19, 23.

RAND_MAX: 19, 23.

sqrt: 6, 23.

sscanf: 10.

stderr: 11, 14, 17, 19, 20, 23.

stdin: 13, 23.

stdout: 13, 19, 22, 23.

strcmp: 10, 23.

teclado: 10, 12, 13.

v: 7.

< 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 *arg* 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

	Seção	Pág.
Introdução	1	1
O que o programa faz	2	1
Média e desvio padrão	4	2
Geração do header file mdp.h	8	3
Estrutura geral do programa	9	4
Linha de comando	10	4
Entrada	13	5
Cálculo da média e do desvio padrão	20	6
Saída	22	6
Inclusão de header files	23	7
Índice	25	8