Avaliação 3

DIM0321

2015.1

Por favor, leia as seguintes instruções:

- Esta é uma prova individual. É expressamente proibido consultar qualquer tipo de material didático e dispositivos eletrônicos. Mantenha em cima da mesa apenas lápis, caneta e borracha. Suas respostas finais devem ser escritas utilizando **unicamente caneta**.
- Os telefones celulares – de qualquer modelo – devem estar desligados e guardados (não podem ficar em cima da mesa).
- Você terá 100 minutos para responder todas as questões. A prova vale 10 pontos. O valor em pontos de cada questão é dado no enunciado.
- Esta prova contará para 70% da sua nota final da primeira unidade. A prova tem 8 páginas e 4 exercícios independentes. Eles podem ser tratados em qualquer ordem.
- Ao terminar a prova, você deve entregá-la ao professor e deixar a sala de aula.

1 Cadeia de caracteres (2.5)

Escreva a função `strncpy` com a declaração abaixo.

```
1 unsigned int strncpy(char *dst, const char *src, unsigned int siz);
```

- A função `strncpy` copia a string `src` para a string `dst` de tamanho `siz`. No máximo `siz - 1` caracteres serão copiados. O caractere nulo sempre termina (salvo se `siz` for 0).
- A função retorna o tamanho de `src`. Se o valor de retorno for maior que `siz`, a cópia foi truncada.

```
1  #include <stdio.h>
2
3  unsigned int
4  strncpy2 (char *dst, const char *src, unsigned int siz)
5  {
6      char *d = dst;
7      const char *s = src;
8      unsigned int n = siz;
9
10     /* Copy as many bytes as will fit */
11     if (n != 0) {
12         while (--n != 0) && ((*d++ = *s++) != '\0');
13     }
14
15     /* Not enough room in dst, add NUL and traverse rest of src */
16     if (n == 0) {
17         if (siz != 0)
18             *d = '\0';          /* NUL-terminate dst */
19         while (*s++) ;
20     }
21
22     return (s - src - 1); /* count does not include NUL */
23 }
24
25 int main() {
26     char *foo = "foo";
27     char *foobar = "foobar";
28     char dst[5];
29     unsigned int n;
30
31     // Tests
32     n = strncpy2(dst, foo, 5);
33     printf ("Size of src = %u, dst = %s\n", n, dst);
34     n = strncpy2(dst, foobar, 5);
35     printf ("Size of src = %u, dst = %s\n", n, dst);
36     return 1;
37 }
```

2 Estruturas / registros (2)

Usando a seguinte definição do tipo `struct data`, representando uma data do calendário.

```
1 struct data {
2     int dia;
3     int mes;
4     int ano;
5 };
```

1. Escrever uma função de comparação

```
1 int comparar_data(struct data d1, struct data d2);
```

Essa função retorne

- 0 se as duas datas são iguais;
- um número positivo se $d_1 > d_2$
- um número negativo senão.

```
1 int comparar_data(struct data d1, struct data d2)
2 {
3     if (d1.ano != d2.ano) return d1.ano - d2.ano;
4     if (d1.mes != d2.mes) return d1.mes - d2.mes;
5     if (d1.dia != d2.dia) return d1.dia - d2.dia;
6     return 0;
7 }
```

2. Escreve uma função para incrementar uma data `d` de um dia.

```
1 void prox_dia(struct data *d);
```

Dicas

- janeiro, março, maio, julho, agosto, outubro e dezembro contém 31 dias
- abril, junho, setembro e novembro contém 30 dias
- fevereiro contém 29 dias nos anos bissextos e 28 dias nos anos não bissextos
- Um ano é bissexto de acordo com as seguintes regras:
 - São bissextos todos os anos múltiplos de 400, p.ex: 1600, 2000, 2400, 2800...
 - São bissextos todos os múltiplos de 4 e não múltiplos de 100, p.ex: 1996, 2004, 2008, 2012, 2016...
 - Não são bissextos todos os demais anos.

```
1 void prox_dia(struct data *d)
2 {
3     if (d->dia < 28) { d->dia++;
4     } else // d->dia >= 28
5     {
6         if (d->mes != 2 && d->dia < 30) d->dia++;
```

```

7      switch (d->mes) {
8      case 1:
9      case 3:
10     case 5:
11     case 7:
12     case 8:
13     case 10:
14     case 12:
15         if (d->dia == 30) d->dia++;
16         else {
17             d->dia = 1;
18             if (d->mes != 12) {
19                 d->mes++;
20             }
21             else {
22                 d->ano++;
23                 d->mes = 1;
24             }
25         }
26         break;
27     case 4:
28     case 6:
29     case 9:
30     case 11:
31         d->dia = 1;
32         d->mes++;
33         break;
34     case 2:
35         if (d->ano % 4 == 0 &&
36             ((d->ano % 100 != 0) || (d->ano % 400 != 0))) {
37             if (d->dia == 28) {
38                 d->dia++;
39             } else {
40                 d->dia = 1;
41                 d->mes = 3;
42             }
43         } else {
44             d->dia = 1;
45             d->mes = 3;
46         }
47         break;
48     default:
49         /* NOT REACHED */
50         break;
51     }
52 }
53 }

```

3 Quadrado mágico (3.5)

Uma matriz *quadrada* de dimensão $n \times n$ é um **quadrado mágico** de ordem n se todos os inteiros $1, 2, 3, 4, \dots, n^2$ aparecem na matriz e se, em cada linha, cada coluna, e cada diagonal, a soma dos elementos resulta em um mesmo valor.

A seguinte matriz é um quadrado mágico:

$$\begin{pmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{pmatrix}$$

1. Escrever uma função que recebe um inteiro $i \leq n$ e uma matriz $\text{mat } n \times n$ e calcula a soma dos elementos da linha i da matriz.

```
1 | int soma_lin(int n, int i, int mat[n][n]);
```

Nas próximas questões, suponha a existência de uma função

```
1 | int soma_col(int n, int j, int mat[n][n]);
```

que realiza a soma de uma dada coluna j duma matriz de dimensão $n \times n$.

2. Escrever uma função que verifica se todos os inteiros do intervalo $[1, n^2]$ ocorrem numa dada matriz $n \times n$

```
1 | bool verific(int n, int mat[n][n]);
```

3. Utilizar essas funções para escrever uma função para verificar se uma dada matriz é um quadrado mágico de ordem $n \leq 100$, de acordo com a seguinte declaração.

```
1 | bool quadrado_magico(int n, int mat[n][n]);
```

```
1 | #include <assert.h>
2 | #include <stdio.h>
3 | #include <stdbool.h>
4 | #include <stdlib.h>
5 |
6 | int soma_lin(int n, int i, int mat[n][n])
7 | {
8 |     int sum = 0, j;
9 |     for (j = 0; j < n; j++) sum += mat[i][j];
10 |    return sum;
11 | }
12 |
13 | int soma_col(int n, int i, int mat[n][n])
14 | {
15 |     int sum = 0, j;
16 |     for (j = 0; j < n; j++) sum += mat[j][i];
17 |    return sum;
18 | }
19 |
```

```

20
21 bool verific(int n, int m[n][n])
22 {
23     bool v[n * n];
24     int i, j;
25
26     for (i = 0; i < n * n; i++) v[i] = false;
27
28     for (i = 0; i < n; i++)
29         for (j = 0; j < n; j++) {
30             if ((m[i][j] > n * n) || (m[i][j] < 1)) {
31                 printf("Verif failed because of m[%d][%d] = %d\n", i, j, m[i][j]);
32                 return false;
33             }
34             v[m[i][j] - 1] = true;
35         }
36
37     for (i = 0; i < n * n; i++) printf("%d", v[i]);
38     printf("\n");
39     for (i = 0; i < n * n; i++)
40         if (!v[i]) return false;
41     return true;
42 }
43
44 bool quadrado_magico(int n, int m[n][n])
45 {
46     int sum, sum_diag;
47     int i;
48
49     if (!verific(n, m)) return false;
50
51     sum = soma_lin(n, 0, m);
52     for (i = 1; i < n; i++)
53         if (soma_lin(n, i, m) != sum) return false;
54
55     for (i = 0; i < n; i++)
56         if (soma_col(n, i, m) != sum) return false;
57
58     sum_diag = m[0][0];
59     for (i = 1; i < n; i++) sum_diag += m[i][i];
60     if (sum_diag != sum) return false;
61
62     sum_diag = m[0][n - 1];
63     for (i = 1; i < n; i++) sum_diag += m[i][n - i - 1];
64     if (sum_diag != sum) return false;
65
66     return true;
67 }

```

```

68
69
70 int main(int argc, char *argv[])
71 {
72     assert(argc > 1);
73     const char *fname = argv[1];
74     FILE *f;
75
76     if ((f = fopen(fname, "r")) != NULL) {
77         int n;
78         fscanf(f, "%d", &n);
79         assert(n > 1);
80         {
81             int m[n][n];
82             for (int i = 0; i < n; i++)
83                 for (int j = 0; j < n; j++)
84                     fscanf(f, "%d", &m[i][j]);
85             for (int i = 0; i < n; i++) {
86                 for (int j = 0; j < n; j++)
87                     printf(" %d", m[i][j]);
88                 printf("\n");
89             }
90             fclose(f);
91             printf("QM: %d\n", quadrado_magico(n, m));
92         }
93         return 0;
94     }
95     return 1;
96 }

```

4 Linhas num arquivo (2)

Escreva uma função para contar o número de linhas num arquivo cujo nome é fornecido em parâmetro. Essa função retorna um valor de -1 em caso de problema.

```

1 | int count_lines(char *filename);
2
3 | #include <stdio.h>
4
5 | int count_lines(char *filename)
6 | {
7 |     int n = 0;
8 |     FILE *f;
9 |
10 |     if ((f = fopen(filename, "r")) != NULL) {
11 |         int c;
12 |         while ((c = fgetc(f)) != EOF)
13 |             if (c == '\n') n++;
14 |     }
15 |     return n;
16 | }

```

```
12     return n;
13 }
14
15     return -1;
16
17 }
18
19 int main(int argc, char *argv[])
20 {
21     printf("Nlines = %d\n", count_lines(argv[1]));
22     return 0;
23 }
```