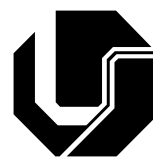




Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Experimental de Metrologia



Programação Procedimental

Lista 5 - Laboratório

Dr. Marcelo Barros de Almeida

Matheus Souza Da Costa

12021EEL021

Lista de ilustrações

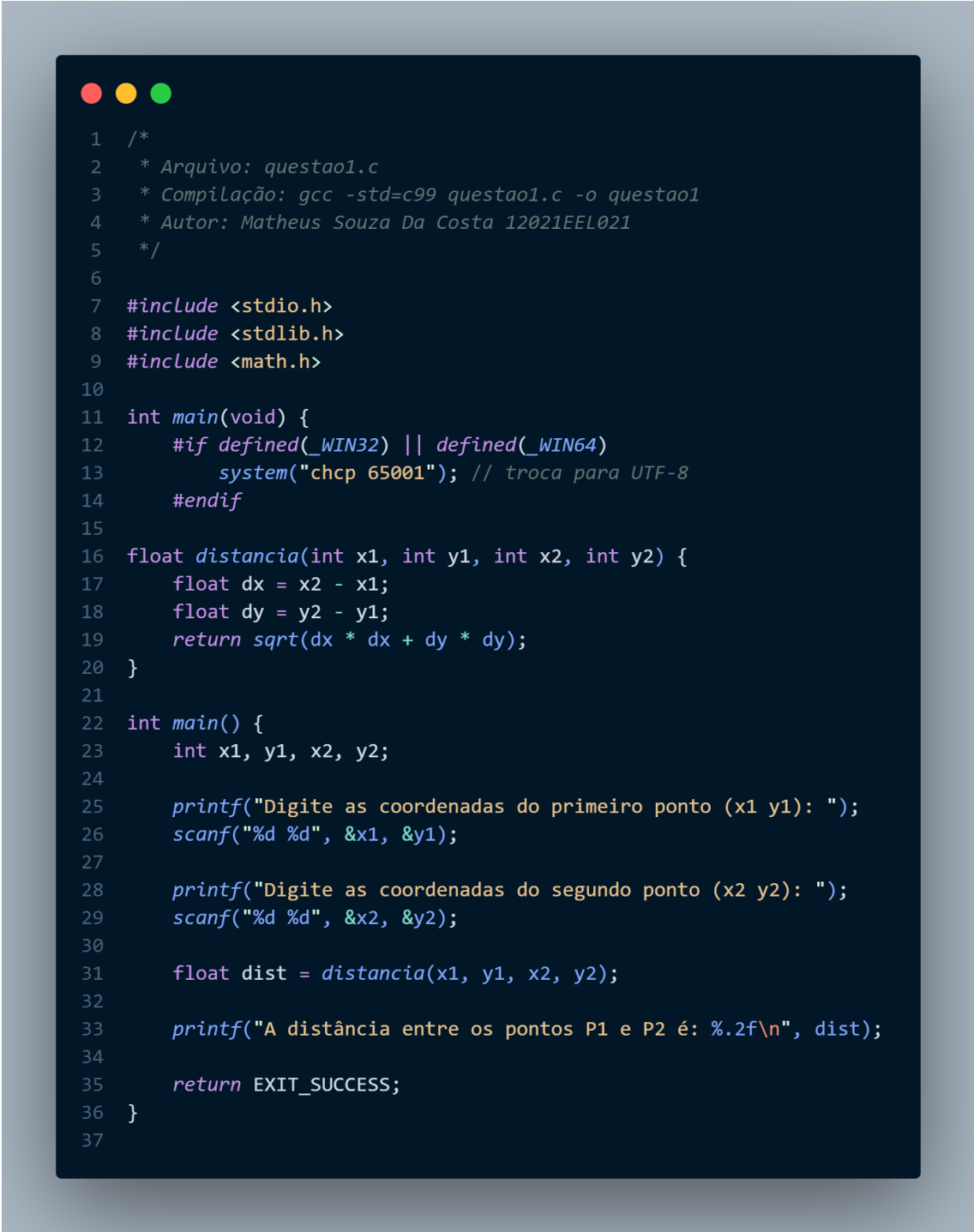
Figura 1 – Questão 1	4
Figura 2 – Questão 2	7
Figura 3 – Questão 3	8
Figura 4 – Questão 4	9
Figura 5 – Questão 5	10
Figura 6 – Questão 6	11
Figura 7 – Questão 7	12
Figura 8 – Questão 8	13
Figura 9 – Questão 10	14

Sumário

1	Questão	4
2	Questão	4
3	Questão	5
4	Questão	5
5	Questão	5
6	Questão	5
7	Questão	5
8	Questão	6
9	Questão	6
10	Questão	6

1 Questão

Resposta conforme imagem 1



```
1  /*
2   * Arquivo: questao1.c
3   * Compilação: gcc -std=c99 questao1.c -o questao1
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <math.h>
10
11 int main(void) {
12     #if defined(_WIN32) || defined(_WIN64)
13         system("chcp 65001"); // troca para UTF-8
14     #endif
15
16 float distancia(int x1, int y1, int x2, int y2) {
17     float dx = x2 - x1;
18     float dy = y2 - y1;
19     return sqrt(dx * dx + dy * dy);
20 }
21
22 int main() {
23     int x1, y1, x2, y2;
24
25     printf("Digite as coordenadas do primeiro ponto (x1 y1): ");
26     scanf("%d %d", &x1, &y1);
27
28     printf("Digite as coordenadas do segundo ponto (x2 y2): ");
29     scanf("%d %d", &x2, &y2);
30
31     float dist = distancia(x1, y1, x2, y2);
32
33     printf("A distância entre os pontos P1 e P2 é: %.2f\n", dist);
34
35     return EXIT_SUCCESS;
36 }
37
```

Figura 1 – Questão 1

2 Questão

Resposta conforme imagem 2

3 Questão

Resposta conforme imagem 3

4 Questão

Resposta conforme imagem 4

5 Questão

Resposta conforme imagem 5

6 Questão

Resposta conforme imagem 6

7 Questão

A função `imprime_alguma_coisa` é uma função recursiva que imprime a representação binária de um número inteiro não negativo n . Ela utiliza a recursão para converter o número decimal em seu equivalente binário e, em seguida, imprime os dígitos binários.

Aqui está como a função funciona:

A função `imprime_alguma_coisa` recebe um parâmetro n , que é o número inteiro a ser convertido em binário.

A função verifica se n não é igual a zero ($n \neq 0$). Se n for zero, a recursão para.

Dentro do bloco `if`, a função chama a si mesma com $n / 2$. Isso divide o número n por 2, efetivamente deslocando todos os seus dígitos binários para a direita. Isso é feito para processar o próximo dígito binário.

Em seguida, a função imprime o dígito binário mais à direita de n , que é obtido calculando $n \% 2$. O `printf` é usado para imprimir o caractere correspondente ao dígito binário. O uso de `'0' + n \% 2` converte o valor binário (0 ou 1) em seu caractere correspondente na tabela ASCII ('0' ou '1').

O processo se repete até que n seja igual a zero, o que ocorre quando todos os dígitos binários tenham sido impressos.

Portanto, a função `imprime_alguma_coisa` imprime a representação binária de n da esquerda para a direita, começando com o dígito binário mais significativo até o menos significativo. Por exemplo, se você chamar `imprime_alguma_coisa(5)`, ela imprimirá "101" porque o binário de 5 é 101. Resposta conforme imagem 7

8 Questão

Resposta conforme imagem 8

9 Questão

A versão iterativa do fatorial geralmente faz menos operações computacionais do que a versão recursiva. Isso ocorre porque na versão iterativa, você usa um loop para calcular o fatorial, acumulando o resultado à medida que avança. Na versão recursiva, você faz várias chamadas de função e multiplica os resultados retornados por essas chamadas.

Em relação à memória, a versão iterativa geralmente consome menos memória do que a versão recursiva, pois não precisa manter uma pilha de chamadas de função. Cada vez que uma função recursiva é chamada, informações sobre a função atual, como argumentos e endereço de retorno, são empilhadas na pilha de chamadas. Em casos de números muito grandes, a versão recursiva pode resultar em um estouro de pilha (stack overflow) devido ao uso excessivo de memória.

Portanto, em termos de eficiência de tempo e uso de memória, a versão iterativa do fatorial é geralmente preferida para números grandes. No entanto, a versão recursiva é mais simples de implementar e compreender em muitos casos. Portanto, a escolha entre as duas depende das necessidades específicas do seu programa e da eficiência desejada.

10 Questão

Resposta conforme imagem 9

```

1  /*
2   * Arquivo: questao2.c
3   * Compilação: gcc -std=c99 questao2.c -o questao2
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int ehPrimo(int i) {
11     if (i <= 1) {
12         return 0; // Números menores ou iguais a 1 não são primos
13     }
14     for (int j = 2; j * j <= i; j++) {
15         if (i % j == 0) {
16             return 0; // Encontrou um divisor, não é primo
17         }
18     }
19     return 1; // Se nenhum divisor foi encontrado, é primo
20 }
21
22 int main(void) {
23     #if defined(_WIN32) || defined(_WIN64)
24         system("chcp 65001"); // troca para UTF-8
25     #endif
26
27     int n;
28
29     printf("Digite um valor inteiro n: ");
30     scanf("%d", &n);
31
32     printf("Números primos de 1 até %d:\n", n);
33     for (int i = 1; i <= n; i++) {
34         if (ehPrimo(i)) {
35             printf("%d ", i);
36         }
37     }
38
39     printf("\n");
40
41     return EXIT_SUCCESS;
42 }
43

```

Figura 2 – Questão 2

```

1  /*
2  * Arquivo: questao3.c
3  * Compilação: gcc -std=c99 questao3.c -o questao3
4  * Autor: Matheus Souza Da Costa 12021EEL021
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 // Função para calcular a soma dos divisores próprios de um número
11 int somaDivisores(int num) {
12     int soma = 0;
13     for (int i = 1; i <= num / 2; i++) {
14         if (num % i == 0) {
15             soma += i;
16         }
17     }
18     return soma;
19 }
20
21 // Função para verificar se dois números são amigos
22 int amigos(int a, int b) {
23     int soma_a = somaDivisores(a);
24     int soma_b = somaDivisores(b);
25
26     // Dois números são amigos se a soma dos divisores próprios de um deles é igual ao outro
27     return (soma_a == b && soma_b == a);
28 }
29
30 int main(void) {
31     #if defined(_WIN32) || defined(_WIN64)
32         system("chcp 65001"); // troca para UTF-8
33     #endif
34
35     int num1, num2;
36
37     printf("Digite dois números inteiros positivos: ");
38     scanf("%d %d", &num1, &num2);
39
40     if (amigos(num1, num2)) {
41         printf("%d e %d são números amigos.\n", num1, num2);
42     } else {
43         printf("%d e %d não são números amigos.\n", num1, num2);
44     }
45
46     return EXIT_SUCCESS;
47 }
48

```

Figura 3 – Questão 3


```

1  /*
2   * Arquivo: questao4.c
3   * Compilação: gcc -std=c99 questao4.c -o questao4
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 // Função para inverter um vetor de inteiros
11 void inverte(int A[], int n) {
12     int inicio = 0;
13     int fim = n - 1;
14
15     while (inicio < fim) {
16         // Troca os elementos nas posições inicio e fim
17         int temp = A[inicio];
18         A[inicio] = A[fim];
19         A[fim] = temp;
20
21         // Move os índices para a próxima posição e a anterior
22         inicio++;
23         fim--;
24     }
25 }
26
27 int main(void) {
28     #if defined(_WIN32) || defined(_WIN64)
29         system("chcp 65001"); // troca para UTF-8
30     #endif
31
32     int n;
33
34     printf("Digite o tamanho do vetor: ");
35     scanf("%d", &n);
36
37     int vetor[n];
38
39     printf("Digite os elementos do vetor:\n");
40     for (int i = 0; i < n; i++) {
41         scanf("%d", &vetor[i]);
42     }
43
44     printf("Vetor original:\n");
45     for (int i = 0; i < n; i++) {
46         printf("%d ", vetor[i]);
47     }
48
49     inverte(vetor, n);
50
51     printf("\nVetor invertido:\n");
52     for (int i = 0; i < n; i++) {
53         printf("%d ", vetor[i]);
54     }
55
56     return EXIT_SUCCESS;
57 }
58

```

Figura 4 – Questão 4

```

1  /*
2   * Arquivo: questao5.c
3   * Compilação: gcc -std=c99 questao5.c -o questao5
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 // Função para verificar se B está contido em A
11 int compara(int A[], int tamA, int B[], int tamB) {
12     // Para cada elemento de B
13     for (int i = 0; i < tamB; i++) {
14         int encontrado = 0;
15
16         // Verifica se o elemento está em A
17         for (int j = 0; j < tamA; j++) {
18             if (B[i] == A[j]) {
19                 encontrado = 1;
20                 break;
21             }
22         }
23
24         // Se o elemento não foi encontrado em A, retorna 0
25         if (!encontrado) {
26             return 0;
27         }
28     }
29
30     // Se todos os elementos de B foram encontrados em A, retorna 1
31     return 1;
32 }
33
34 int main(void) {
35
36     #if defined(_WIN32) || defined(_WIN64)
37         system("chcp 65001"); // troca para UTF-8
38     #endif
39
40     int tamA, tamB;
41
42     printf("Digite o tamanho do vetor A: ");
43     scanf("%d", &tamA);
44
45     int vetorA[tamA];
46
47     printf("Digite os elementos do vetor A:\n");
48     for (int i = 0; i < tamA; i++) {
49         scanf("%d", &vetorA[i]);
50     }
51
52     printf("Digite o tamanho do vetor B: ");
53     scanf("%d", &tamB);
54
55     int vetorB[tamB];
56
57     printf("Digite os elementos do vetor B:\n");
58     for (int i = 0; i < tamB; i++) {
59         scanf("%d", &vetorB[i]);
60     }
61
62     int resultado = compara(vetorA, tamA, vetorB, tamB);
63
64     if (resultado) {
65         printf("B está contido em A.\n");
66     } else {
67         printf("B NÃO está contido em A.\n");
68     }
69
70     return EXIT_SUCCESS;
71 }
72

```

Figura 5 – Questão 5

```

1  /*
2  * Arquivo: questao6.c
3  * Compilação: gcc -std=c99 questao6.c -o questao6
4  * Autor: Matheus Souza Da Costa 12021EEL021
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 // Função para calcular a média aritmética de um vetor de números reais
11 float media(float F[], int n) {
12     if (n == 0) {
13         return 0.0; // Retorna 0 se o vetor estiver vazio para evitar divisão por zero
14     }
15
16     float soma = 0.0;
17
18     // Calcula a soma de todos os elementos no vetor
19     for (int i = 0; i < n; i++) {
20         soma += F[i];
21     }
22
23     // Calcula a média aritmética dividindo a soma pelo número de elementos
24     return soma / n;
25 }
26
27 int main(void) {
28     #if defined(_WIN32) || defined(_WIN64)
29         system("chcp 65001"); // troca para UTF-8
30     #endif
31
32     int n;
33
34     printf("Digite o tamanho do vetor: ");
35     scanf("%d", &n);
36
37     float vetor[n];
38
39     printf("Digite os elementos do vetor:\n");
40     for (int i = 0; i < n; i++) {
41         scanf("%f", &vetor[i]);
42     }
43
44     float resultado = media(vetor, n);
45
46     printf("A média aritmética dos elementos do vetor é: %.2f\n", resultado);
47
48
49     return EXIT_SUCCESS;
50 }
51
52

```

Figura 6 – Questão 6

```

1  /*
2  * Arquivo: questao7.c
3  * Compilação: gcc -std=c99 questao7.c -o questao7
4  * Autor: Matheus Souza Da Costa 12021EEL021
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 void imprime_alguma_coisa(int n){
11     if (n != 0) {
12         imprime_alguma_coisa(n / 2);
13         printf("%c", '0' + n % 2);
14     }
15 }
16
17 int main(void) {
18     #if defined(_WIN32) || defined(_WIN64)
19         system("chcp 65001"); // troca para UTF-8
20     #endif
21
22     int numero;
23
24     printf("Digite um número inteiro não negativo: ");
25     scanf("%d", &numero);
26
27     printf("A representação binária de %d é: ", numero);
28     if (numero == 0) {
29         printf("0"); // Caso especial: 0 em binário é apenas "0".
30     } else {
31         imprime_alguma_coisa(numero);
32     }
33     printf("\n");
34
35     return EXIT_SUCCESS;
36 }
37
38

```

Figura 7 – Questão 7

```

1  /*
2   * Arquivo: questao8.c
3   * Compilação: gcc -std=c99 questao8.c -o questao8
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 long long int fat(int n) {
11     if (n == 0 || n == 1) {
12         return 1;
13     } else {
14         return n * fat(n - 1);
15     }
16 }
17 int main(void) {
18     #if defined(_WIN32) || defined(_WIN64)
19         system("chcp 65001"); // troca para UTF-8
20     #endif
21
22     int numero;
23
24     printf("Digite um número inteiro não negativo: ");
25     scanf("%d", &numero);
26
27     if (numero < 0) {
28         printf("0 fatorial não está definido para números negativos.\n");
29     } else {
30         long long int resultado = fat(numero);
31         printf("0 fatorial de %d é %lld.\n", numero, resultado);
32     }
33
34
35     return EXIT_SUCCESS;
36 }
37

```

Figura 8 – Questão 8

```

1  /*
2   * Arquivo: questao10.c
3   * Compilação: gcc -std=c99 questao10.c -o questao10
4   * Autor: Matheus Souza Da Costa 12021EEL021
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 double pot(int a, int b) {
11     if (b == 0) {
12         return 1.0; // Qualquer número elevado a 0 é 1
13     } else if (b > 0) {
14         return a * pot(a, b - 1); // Caso positivo: a * a^(b-1)
15     } else {
16         return 1.0 / (a * pot(a, -b - 1)); // Caso negativo: 1 / (a * a^(-b-1))
17     }
18 }
19
20 int main(void) {
21     #if defined(_WIN32) || defined(_WIN64)
22         system("chcp 65001"); // troca para UTF-8
23     #endif
24
25     int base, expoente;
26
27     printf("Digite a base: ");
28     scanf("%d", &base);
29
30     printf("Digite o expoente: ");
31     scanf("%d", &expoente);
32
33     double resultado = pot(base, expoente);
34     printf("%d^%d = %lf\n", base, expoente, resultado);
35
36     return EXIT_SUCCESS;
37 }
38

```

Figura 9 – Questão 10