My Project

LUCAS VIEIRA DE ALMEIDA
Version 1.0

Table of Contents

Table of contents

File Index

File List

Here is a list of all files with brief descriptions:

File Documentation

códigos/array3D.c File Reference

`#include <stdio.h>`

Functions

- int **main** (int argc, char *argv[])

Function Documentation

int main (int *argc*, char * *argv*[])

```
3                                   {
4
5
6      int dim1 = 10, dim2 = 20, dim3 = 5;
7      int array3D[dim1][dim2][dim3];
8
9
10      int value = 1;
11      for (int i = 0; i < dim1; i++) {
12      for (int j = 0; j < dim2; j++) {
13      for (int k = 0; k < dim3; k++) {
14
15      array3D[i][j][k] = value++;
16
17      }
18    }
19 }
20
21      for (int i = 0; i < dim1; i++) {
22      for (int j = 0; j < dim2; j++) {
23      for (int k = 0; k < dim3; k++) {
24
25      printf("%d ", array3D[i][j][k]);
26
27      }
28      printf("\n");
29
30      }
31      printf("\n");
32
33      }
34
35     system("pause");
36
37      return 0;
38 }
```

códigos/arrayunidimensional.c File Reference

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

Macros

- #define **TAM** 10000

Functions

- void **bubbleSort** (int array[], int n)
- int **calcularMedia** (int array[], int n)
- int **calcularMediana** (int array[], int n)
- int **calcularDesvioPadrao** (int array[], int n, int media)
- void **encontrarValoresRepetidos** (int array[], int n)
- int * **removerRepetidos** (int array[], int n, int *novoTamanho)
- int **main** (int argc, char *argv[])

Macro Definition Documentation

#define TAM 10000

Function Documentation

void bubbleSort (int *array*[], int *n*)

```
8                                       {
9     for (int i = 0; i < n - 1; i++) {
10     for (int j = 0; j < n - i - 1; j++) {
11
12     if (array[j] > array[j + 1]) {
13
14     int temp = array[j];
15     array[j] = array[j + 1];
16     array[j + 1] = temp;
17             }
18         }
19     }
20 }
```

int calcularDesvioPadrao (int *array*[], int *n*, int *media*)

```
40                                                              {
41     int somaQuadrados = 0;
42     for (int i = 0; i < n; i++) {
43         somaQuadrados += pow(array[i] - media, 2);
44     }
45     return sqrt(somaQuadrados / n);
46 }
```

int calcularMedia (int *array*[], int *n*)

```
22                                          {
23     int soma = 0;
24     for (int i = 0; i < n; i++) {
25         soma += array[i];
26     }
27     return soma / n;
28 }
```

int calcularMediana (int *array*[], int *n*)

```
30                                              {
31      if (n % 2 == 0) {
32
33          return (array[n / 2 - 1] + array[n / 2]) / 2.0;
34      } else {
35
36          return array[n / 2];
37      }
38  }
```

void encontrarValoresRepetidos (int *array*[], int *n*)

```
48                                                      {
49      int valorAtual = array[0];
50      int contagem = 1;
51
52      for (int i = 1; i < n; i++) {
53          if (array[i] == valorAtual) {
54              contagem++;
55          } else {
56              printf("Valor: %d, Quantidade: %d\n", valorAtual, contagem);
57              valorAtual = array[i];
58              contagem = 1;
59          }
60      }
61
62
63      printf("Valor: %d, Quantidade: %d\n", valorAtual, contagem);
64  }
```

int main (int *argc*, char * *argv*[])

```
88                                      {
89      int myArray[TAM];
90
91      srand(time(NULL));
92
93      printf("%-10s %13s\n", "Elemento", "Valor");
94
95      for (int i = 0; i < TAM; i++) {
96          myArray[i] = rand() % 1001;
97          printf("%7d %13d\n", i + 1, myArray[i]);
98      }
99
100      bubbleSort(myArray, TAM);
101
102      printf("\nArray Ordenado:\n");
103      printf("%-10s %13s\n", "Elemento", "Valor");
104      for (int i = 0; i < TAM; i++) {
105          printf("%7d %13d\n", i + 1, myArray[i]);
106      }
107
108      printf("\n3 Menores Números:\n");
109      for (int i = 0; i < 3; i++) {
110      printf("%d\n", myArray[i]);
111     }
112
113      printf("\n3 Maiores Números:\n");
114      for (int i = TAM - 1; i >= TAM - 3; i--) {
115      printf("%d\n", myArray[i]);
116     }
117
118    double media = calcularMedia(myArray, TAM);
119    double mediana = calcularMediana(myArray, TAM);
120    double desvioPadrao = calcularDesvioPadrao(myArray, TAM, media);
121
122    printf("\nMédia: %f\n", media);
123    printf("Mediana: %f\n", mediana);
124    printf("Desvio Padrão: %f\n", desvioPadrao);
125
126    printf("\nValores Repetidos:\n");
127    encontrarValoresRepetidos(myArray, TAM);
128
129    int novoTamanho;
130    int* novoArray = removerRepetidos(myArray, TAM, &novoTamanho);
```

```
131
132   printf("\nNovo Array sem Repetições:\n");
133   printf("%-10s %13s\n", "Elemento", "Valor");
134   for (int i = 0; i < novoTamanho; i++) {
135
136   printf("%7d %13d\n", i, novoArray[i]);
137   }
138
139   media = calcularMedia(novoArray, novoTamanho);
140   mediana = calcularMediana(novoArray, novoTamanho);
141   desvioPadrao = calcularDesvioPadrao(novoArray, novoTamanho, media);
142
143   printf("\nMédia (Novo Array): %f\n", media);
144   printf("Mediana (Novo Array): %f\n", mediana);
145   printf("Desvio Padrão (Novo Array): %f\n", desvioPadrao);
146
147   free(novoArray);
148
149 system("pause");
150
151     return 0;
152 }
```

int * removerRepetidos (int *array*[], int *n*, int * *novoTamanho*)

```
66                                                                  {
67     int* novoArray = (int*)malloc(n * sizeof(int));
68     if (!novoArray) {
69
70         exit(EXIT_FAILURE);
71     }
72
73     int valorAtual = array[0];
74     novoArray[0] = valorAtual;
75     *novoTamanho = 1;
76
77     for (int i = 1; i < n; i++) {
78         if (array[i] != valorAtual) {
79             valorAtual = array[i];
80             novoArray[*novoTamanho] = valorAtual;
81             (*novoTamanho)++;
82         }
83     }
84
85     return novoArray;
86 }
```

códigos/calculoimc.c File Reference

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

Functions

- int **main** ()

Function Documentation

int main ()

```
4           {
5   float h, w, imc, hC, iW; // hC = altura em cm; iW = peso ideal
6   char sex;
7
8   printf("Insira sua altura (m): ");
9   scanf("%f", &h);
10
11   printf("Insira seu peso (kg): ");
12   scanf("%f", &w);
13
14   printf("Marque seu sexo (F/M): ");
15   scanf(" %c", &sex);
16
17   imc = w / pow(h, 2);
18
19   if (sex == 'F') {
20
21       printf("\nÍndice de Massa Corporal: %f", imc);
22       hC = h * 100;
23       iW = hC - 100 - ((hC - 150) / 2);
24     if (imc < 19.0) {
25       printf("\nFaixa: Abaixo do peso");
26     } else if (imc >= 19.0 && imc < 23.9) {
27       printf("\nFaixa: Peso normal");
28     } else if (imc >= 24.0 && imc < 28.9) {
29       printf("\nFaixa: Sobrepeso");
30     } else if (imc >= 29.0 && imc < 33.9) {
31       printf("\nFaixa: Obesidade I");
32     } else if (imc >= 34.0 && imc < 38.9) {
33       printf("\nFaixa: Obesidade II");
34     } else if (imc >= 39.0) {
35       printf("\nFaixa: Obesidade III");
36     }
37       printf("\nPeso Ideal: %f", iW);
38   } else if (sex == 'M') {
39
40      printf("\nÍndice de Massa Corporal: %f", imc);
41       hC = h * 100;
42       iW = hC - 100 - ((hC - 150) / 4);
43     if (imc < 20.0) {
44       printf("\nFaixa: Abaixo do peso");
45     } else if (imc >= 20.0 && imc < 24.9) {
46       printf("\nFaixa: Peso normal");
47     } else if (imc >= 25.0 && imc  < 29.9) {
48       printf("\nFaixa: Sobrepeso");
49     } else if (imc >= 30.0 && imc < 34.9) {
50       printf("\nFaixa: Obesidade I");
51     } else if (imc >= 35.0 && imc < 39.9) {
52       printf("\nFaixa: Obesidade II");
53     } else if (imc >= 40.0) {
54       printf("\nFaixa: Obesidade III");
55     }
56      printf("\nPeso Ideal: %f", iW);
57   }
58
59   system("pause");
60
61   return 0;
```

```
62 }
```

códigos/lista1q12.c File Reference

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

Functions

- int **main** ()

Function Documentation

int main ()

```
4            {
5   double cateto1, cateto2, hipotenusa, perimetro, area;
6
7   printf("Informe medida do cateto oposto:");
8   scanf("%lf", &cateto1);
9
10   printf("Informe medida do cateto adjascente:");
11   scanf("%lf", &cateto2);
12
13   hipotenusa = sqrt(pow(cateto1,2) + pow(cateto2,2));
14   perimetro = cateto1 + cateto2 + hipotenusa;
15   area = cateto2 * cateto1 / 2;
16
17   printf("A hipotenusa é: %lf\n", hipotenusa);
18   printf("O perimetro é: %lf\n", perimetro);
19   printf("A area é: %lf\n", area);
20
21   system("pause");
22
23   return 0;
24 }
```

códigos/lista1q13.c File Reference

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

Functions

- int **main** ()

Function Documentation

int main ()

```
4            {
5   double a, b, c;
6
7   printf("Digite o coeficiente a do polinômio (ax^2 + bx + c): ");
8   scanf("%lf", &a);
9   printf("\nDigite o coeficiente b do polinômio (ax^2 + bx + c): ");
10   scanf("%lf", &b);
11   printf("\nDigite o coeficiente c do polinômio (ax^2 + bx + c): ");
12   scanf("%lf", &c);
13
14   double delta = pow(b,2) - 4 * a * c;
15
16   if (a == 0) {
17
18   printf("Isso não é um polinômio de segundo grau.\n");
19
20   } else if (delta > 0) {
21
22   double x1 = (-b + sqrt(delta)) / (2 * a);
23   double x2 = (-b - sqrt(delta)) / (2 * a);
24
25   printf("O polinômio tem duas raízes reais: x1 = %.2lf e x2 = %.2lf\n", x1, x2);
26
27   } else if (delta == 0) {
28
29   double x1 = -b / (2 * a);
30
31   printf("O polinômio tem uma raiz real: x = %.2lf\n", x1);
32
33   } else {
34
35   printf("O polinômio não tem raízes reais.\n");
36
37   }
38
39   double x;
40   printf("Digite um valor para x: ");
41   scanf("%lf", &x);
42
43   double px = (a * pow(x,2)) + (b * x) + c;
44
45   printf("p(x) = %.2lf\n", px);
46
47   system("pause");
48
49   return 0;
50
51 }
```

códigos/lista1q2.c File Reference

```
#include <stdio.h>
#include <ctype.h>
```

Functions

- int **main** ()

Function Documentation

int main ()

```
4            {
5
6   char ch1, ch2, ch3;
7
8   printf("\n\nPara caractere ch1\n\n");
9
10   printf("Digite um caractere: ");
11   scanf("%c", &ch1);
12
13  ch1 >= 'A' && ch1 <= 'Z' ? printf("O caractere digitado é uma letra maiuscula:
%c\n",ch1) :
14  ch1 >= 'a' && ch1 <= 'z' ? printf("O caractere digitado é uma letra minuscula:
%c\n",ch1) :
15  ch1 >= '0' && ch1 <= '9' ? printf("O caractere digitado é um numero: %c\n",ch1)
: printf("O caractere digitado é um símbolo: %c", ch1);
16
17   printf("\n\nPara caractere ch2\n\n");
18
19   ch2 = 81;
20
21   printf("Caractere em char: %c\n ", ch2);
22
23   printf("Caractere em decimal: %d\n ", ch2);
24
25   printf("Caractere em octal: %o\n ", ch2);
26
27   printf("Caractere em hexadecimal: %X\n ", ch2);
28
29
30   printf("\n\nPara caractere ch3\n\n");
31
32   if (ch2 >= 'A' && ch2 <= 'Z') {
33     ch3 = ch2 + ('a' - 'A');
34
35     printf("Caractere em char: %c\n ", ch3);
36
37     printf("Caractere em decimal: %d\n ", ch3);
38
39     printf("Caractere em octal: %o\n ", ch3);
40
41     printf("Caractere em hexadecimal: %X\n ", ch3);
42
43   } else {
44     printf("O caractere atribuído a ch2 não é uma letra maiúscula\n");
45   }
46
47   system("pause");
48
49  return 0;
50 }
```

códigos/lista1q48.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Functions

- int **main** ()

Function Documentation

int main ()

```
5            {
6
7   printf("\n\n\nCRAPS\n\n\n");
8
9   srand(time(NULL));
10
11   int dado1 = (rand() % 6) + 1;
12   int dado2 = (rand() % 6) + 1;
13   int ponto, soma = dado1 + dado2, novaSoma;
14
15  printf("Primeiro lançamento: %d + %d = %d\n",  dado1, dado2, soma);
16
17   if ( soma == 7 || soma == 11) {
18     printf("%d, Você ganhou!:)", soma);
19
20   } else if ( soma == 2 || soma == 3 || soma == 12) {
21     printf("%d, Perdeu!:( A casa vence.", soma);
22
23   } else {
24     ponto = soma;
25     printf("Ponto é: %d\n", ponto);
26
27     while(1) {
28
29     dado1 = (rand() % 6) + 1;
30     dado2 = (rand() % 6) + 1;
31     novaSoma = dado1 + dado2;
32
33     printf("Lançamento: %d + %d = %d\n",  dado1, dado2, novaSoma);
34
35     if (novaSoma == ponto) {
36        printf("%d, Você ganhou!:)", novaSoma);
37        break;
38
39     } else if (novaSoma == 7) {
40       printf("%d, Perdeu!:( A casa vence.", novaSoma);
41       break;
42     }
43     }
44
45   }
46
47
48   return 0;
49 }
```

códigos/sbruto.c File Reference

`#include <stdio.h>`

Functions

- int **main** ()

Function Documentation

int main ()

```
3            {
4   float sbruto, sbrutomenos, sliquido, inss, irpf, vale, sminimo;
5
6   printf("\nValor do salário bruto:");
7   scanf("%f", &sbruto);
8
9   printf("\nValor do vale alimentação (insira 0 caso não receba):");
10   scanf("%f", &vale);
11
12   inss = sbruto * (9.0/100);
13   irpf = sbruto * (27.7/100);
14   sbrutomenos = sbruto - inss;
15   sliquido = sbrutomenos - irpf - vale;
16
17   if (sliquido < 1320){
18     sminimo = 1320 + vale;
19     printf("Salário líquido:R$%2.f", sminimo);
20   } else
21   if (sliquido > 1320){
22     printf("Salário líquido:R$%2.f", sliquido);
23   }
24
25   system("pause");
26
27   return 0;
28 }
```

códigos/torredehanoi.c File Reference

`#include <stdio.h>`

Functions

- void **TorredeHanoi** (int origem, int destino, int auxiliar, int **quantidade**)
- int **main** ()

Variables

- int **count** = 0
- int **quantidade** = 0

Function Documentation

int main ()

```
20            {
21
22   printf("Digite quantidade de discos: ");
23   scanf("%d", &quantidade);
24
25   TorredeHanoi(1,2,3,quantidade);
26   printf("Quantidade de discos:%d\n", quantidade);
27   printf("Número de movimentos: %d\n", count);
28
29   system("pause");
30
31   return 0;
32 }
```

void TorredeHanoi (int *origem*, int *destino*, int *auxiliar*, int *quantidade*)

```
6                                                                                {
7
8    if(quantidade == 1){
9      printf("Move de %d para %d\n",origem, destino);
10      count+=1;
11
12    }else{
13
14 TorredeHanoi(origem,auxiliar,destino,quantidade-1);
15 TorredeHanoi(origem,destino,auxiliar,1);
16 TorredeHanoi(auxiliar,destino,origem,quantidade-1);
17    }
18 }
```

Variable Documentation

int count = 0

int quantidade = 0

Index

INDEX