MARATONA DE PROGRAMAÇÃO - CÓDIGOS PRONTOS

Linguagem C

Funções:

> Número primo:

```c
#include <stdio.h>

int primo(int num)
{
    int c, d = 0;

    for(c = 1; c <= num; c++)
        if(num % c == 0) d++;

    if(d == 2) return 1;
    else return 0;
}

int main()
{
    int n = primo(7);

    if(n) printf("Sim");
    else printf("Não");

    return 0;
}

/* SAÍDA

Sim

*/
```

> Ordenar crescente:

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

```c
int *ordernar(int quant_num, int *num)
{
    int x, y, temp;

    for(x = 0; x < quant_num; x++)
    {
        for(y = 0; y < quant_num; y++)
        {
            if(num[y] > num[y + 1])
            {
                temp = num[y];
                num[y] = num[y + 1];
                num[y + 1] = temp;
            }
        }
    }

    for(x = 0; x < quant_num; x++) num[x] = num[x+1];

    num[quant_num] = 0;

    return num;
}

int main()
{
    int num[3];

    num[0] = 3;
    num[1] = 2;
    num[2] = 4;

    ordernar(3, num);

    printf("%d\n", num[0]);
    printf("%d\n", num[1]);
    printf("%d\n", num[2]);

    return 0;
}

/* SAÍDA

2
3
4

*/
```

> Ordernar decrescente:

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

```c
int *ordernar(int quant_num, int *num)
{
    int x, y, temp, desc[quant_num];

    for(x = 0; x < quant_num; x++)
    {
        for(y = 0; y < quant_num; y++)
        {
            if(num[y] > num[y + 1])
            {
                temp = num[y];
                num[y] = num[y + 1];
                num[y + 1] = temp;
            }
        }
    }

    for(x = quant_num - 1; x >= 0; x--) desc[x] = num[quant_num - x];

    for(x = 0; x < quant_num; x++) num[x] = desc[x];

    num[quant_num] = 0;

    return num;
}

int main()
{
    int num[3];

    num[0] = 3;
    num[1] = 2;
    num[2] = 4;

    ordernar(3, num);

    printf("%d\n", num[0]);
    printf("%d\n", num[1]);
    printf("%d\n", num[2]);

    return 0;
}

/* SAÍDA

4
3
2

*/
```

Math (Matemática):

> Arredondar:

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

int main()
{
    float n1 = 5.7;
    float n2 = 5.4;
    float N1, N2;

    // ARREDONDAR PARA BAIXO
    N1 = floor(n1);
    N2 = floor(n2);
    printf("%f\n", N1);
    printf("%f\n", N2);

    printf("\n===============\n\n");

    // ARREDONDAR PARA CIMA
    N1 = ceil(n1);
    N2 = ceil(n2);
    printf("%f\n", N1);
    printf("%f\n", N2);

    printf("\n===============\n\n");

    // ARREDONDAR PARA O MAI PRÓXIMO
    N1 = round(n1);
    N2 = round(n2);
    printf("%f\n", N1);
    printf("%f\n", N2);

    return 0;
}

/* SAÍDA

5.000000
5.000000

===============

6.000000
6.000000

===============

6.000000
5.000000
```

```
*/
```

> Potência:

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

int main()
{
    float r = pow(2, 3); // 2 elevado a 3

    printf("%.2f\n", r);

    return 0;
}

/* SAÍDA

8.00

*/
```

> Raiz quadrada:

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

int main()
{
    float r = sqrt(9); // Raiz quadrada de 9 = 3

    printf("%.2f\n", r);

    return 0;
}

/* SAÍDA

3.00

*/
```

Strings:

> Comparar partes de uma string:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char a[10] = "Curso de A";
    char b[10] = "Curso de B";
    int r;

    r = strncmp(a, b, 8); // Compara apenas os 8 primeiros caracteres

    printf("Retorno = %d\n", r);

    return 0;
}

/*
SE AS STRINGS SÃO IGUAIS = 0
SE A PRIMEIRA STRING É MENOR = -1
SE A SEGUNDA STRING É MAIOR = 1
*/

/* SAÍDA

Retorno = 0

*/
```

> Compara strings:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char s1[3] = "abc";
    char s2[3] = "abd";
    int r;

    r = strcmp(s1, s2);

    printf("Retorno = %d\n", r);

    return 0;
}

/*
SE AS STRINGS SÃO IGUAIS = 0
SE A PRIMEIRA STRING É MENOR = -1
SE A SEGUNDA STRING É MAIOR = 1
*/
```

```
/* SAÍDA

Retorno = -1

*/
```

> Concatenar partes de uma string:

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

int main()
{
    char a[10] = "Curso ";
    char b[10] = " de C na";

    strncat(a, b, 5);

    printf("%s\n", a);

    return 0;
}

/* SAÍDA

Curso de C

*/
```

> Concatenar:

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

int main()
{
    char a[5] = "Curso";
    char b[5] = " C";

    strcat(a, b);

    printf("%s\n", a);

    return 0;
}

/* SAÍDA
```

```
Curso C

*/
```

Copiar partes de uma string:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char a[10] = "1234567890";
    char b[5];

    strncpy(b, a, 5);

    printf("Valor = %s\n", b);

    return 0;
}

/* SAÍDA

Valor = 12345

*/
```

Copiar um string:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char nome[6];

    strcpy(nome, "Fulano");

    printf("Nome = %s\n", nome);

    return 0;
}

/* SAÍDA

Nome = Fulano

*/
```

> Ler uma string:

```c
#include <stdio.h>

int main()
{
    char nome[10];

    scanf("%s", nome);

    printf("Nome = %s\n", nome);

    return 0;
}

/* SAÍDA

Nome = nome

*/
```

> Tamanho da string:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char a[10] = "Curso";
    int t;

    t = strlen(a);

    printf("Tamanho = %d\n", t);

    return 0;
}

/* SAÍDA

Tamanho = 5

*/
```

Compilação no linux:

```
gcc -o exe arq.c
./exe < arq.txt # Arquivo com os valores das variáveis

gcc arq.c -o -lm exe # Para quando usar funções como pow, floor, etc...
./exe < arq.txt # Arquivo com os valores das variáveis
```