

DOM	SEG	TER	QUA	QUI	SEX	SÁB
DOM	LUN	MAR	MIÉ	JUE	VIE	SÁB

Discente → Paulo Henrique Diniz de Lima Alencar

101Q
24 02 2020
matriúla: 494837
Ciências da computação.

Questão 1 - Resolução:

Q1

```
int potencia (int base, int exp) {
    int pot = 1;
    for (int i = 0; i < exp; i++) {
        pot = pot * base;
    }
    return pot;
}
```

→ Responsável
por calcular as potências !

```
int fat (int k) {
    if (k == 0) return 1;
    return k * fat (k-1);
}
```

→ Responsável por calcular o
fatorial !

```
float q1 (float x, int n) {
    int aux = 1; exp = 2;
    float cosx = 1; <=
    for (int i = 0; i < n; i++) {
        aux = aux * (-1);
        cosx = cosx + aux * (potencia(x, exp) / (float) fat(exp));
        exp = exp + 2;
    }
    return cosx;
}
```

→ Responsável por
calcular uma aproximação
de cosx !

```
int main (void) {
```

// chamada da função q1 passando seus argumentos. Após isso, imprimir resultado.

```
return 0;
```

```
}
```



DOM	SEG	TER	QUA	QUI	SEX	SÁB
DOM	LUN	MAR	MIÉ	JUE	VIE	SÁB



Quesão 2 - Resolução:

* O que o algoritmo faz? Calcula o MDC (Máximo divisor comum) dos n termos que o usuário entrou.

* O último inteiro impresso pelo programa é o m , sua relação com os termos pedidos ao usuário é que m é o MDC dos n termos que o usuário digitou.

* Funcionamento:

O algoritmo recebe como entrada um inteiro n [quantidade de números pedidos ao usuário]. Em seguida realiza a leitura do primeiro número, isto é m . Após isso teremos um determinado processo de vai determinar o MDC dos números fornecidos pelo usuário. Claro que, durante esse processo o usuário vai entrar com uma quantidade n de números. Quando tudo terminar, ele vai imprimir $\text{printf}("%d \n", m)$; que corresponde o MDC dos números fornecidos pelo usuário.

Ex: Se $n = 3$, vai ler $\left\{ \begin{array}{l} 1^{\text{º}} \text{número} \rightarrow 4 \\ 2^{\text{º}} \text{número} \rightarrow 6 \\ 3^{\text{º}} \text{número} \rightarrow 12 \end{array} \right\}$ $\text{MDC}(4, 6, 12) = 2$

Vou fazer a exemplificação da 1^º execução:

// Entrada de dados $\rightarrow n = 3$ e $m = 4$

$i = 1;$

While ($i < n$) {

$1 < 3$ [V]

// Entrada de dados \rightarrow numero = 6

$i = i + 1;$ $i = 2$

$\text{div} = 1$ $1 <= 4$ E $1 <= 6$ [V]

While ($\text{div} \leq m \& \& \text{div} \leq \text{numero}$) {

if ($m \% \text{div} == 0 \& \& \text{numero \% div} == 0$) {

$4 \% 1 == 0$ E $6 \% 1 == 0$

[V]

[V]





$novo_m = 1$

DOM	SEG	TER	QUA	QUI	SEX	SÁB
DOM	LUN	MAR	MIÉ	JUE	VIE	SÁB

$novo_m = div$,

}

$div = div + 1$; $div = 2$

}

$m = novo_m$; (novo m é o menor que divide)

}

`printf ("%d", m);`

`return 0;` Vai corresponder ao MDC ($4, 6, 12$) = Quando todo processo terminar variável m será $= 2$ //

Teste de mesa :

n	m	i	div	$i < n$	$div \leq m \wedge div \leq numero$	$m \% div == 0 \wedge numero \% div == 0$	$novo_m$	numero
3	4	1	1	[V]	[V]	[V]	1	6
3	4	2	2		[V]	[V]	2	6
3	4	2	3		[V]	[F]	2	6
3	4	2	4	[V]	[F]		2	6
3	2	3	1		[V]	[V]	1	12
3	2	(3)	2		[V]	[V]	2	12
3	2	3	3		[F]		2	12
3	2	3	4	[F]			2	12

$\rightarrow MDC(4, 6, 12) = 2$ //

DOM | SEG | TER | QUA | QUI | SEX | SÁB
DOM LUN MAR MIÉ JUE VIE SÁB



Questão 3 - Resolução:

2.5

...
int * q3 (int *x, int *y, int *z, int n) {

 int *m;

 int i = n - 1, j = n - 1, f = n - 1;

 m = (int *) malloc (sizeof (int) * 3 * n);

 for (int k = 0; k < 3 * n; k++) {

 if (i == -1 && j == -1) {

 m[k] = z[f];

 f--;

}

 else if (i == -1 && f == -1) {

 m[k] = y[j];

 j--;

}

 else if (f == -1 && j == -1) {

 m[k] = x[i];

 i--;

}

 else if (i == -1) { $i < j \&& i < f \&& i < j$ } fi

 if ($z[f] > y[j]$) {

 m[k] = z[f]; $i < j \&& i < f \&& i < j$ } fi

 f--;

}

 else {

 m[k] = y[j]; $i < j \&& i < f \&& i < j$ } fi

 j--;

}

}

 else if ($j == -1$) {

 i -- Associate ao x
x:

1	3	4
---	---	---

 j -- Associate ao y
y:

0	2	5
---	---	---

 f -- Associate ao z
z:

8	9	11
---	---	----

 m:

11	9	8	5	4	3	2	1	0
0	1	2	3	4	5	6	7	8

* Tabela de organização.



DOM DOM	SEG LUN	TER MAR	QUA MIÉ	QUI JUE	SEX VIE	SÁB SÁB
------------	------------	------------	------------	------------	------------	------------

if ($z[f] > x[i]$) {

 m[K] = z[f];

 f--;

} else {

 m[K] = x[i];

 i--;

}

else if ($f == -1$) {

 if ($y[j] > x[i]$) {

 m[K] = y[j];

 j--;

}

 else {

 m[K] = x[i];

 i--;

}

}

if ($i > -1 \&& j > -1 \&& f > -1$) {

 if ($x[i] \geq y[j] \&& x[i] \geq z[f]$) {

 m[K] = x[i];

 i--;

}

 else if ($y[j] \geq x[i] \&& y[j] \geq z[f]$) {

 m[K] = y[j];

 j--;

}

 else {

 m[K] = z[f];

 f--;



DOM	SEG	TER	QUA	QUI	SEX	SÁB
DOM	LUN	MAR	MIÉ	JUE	VIE	SÁB



: problema de busca em árvore binária de busca. N.

}

return m;

}

int main (void) {

...

int *m = q3 (x, y, z);

for (int i = 0; i < 3*n; i++) {

printf ("%d ", *(m+i));

};

return 0;

}

Questão 4 - Resolvida:

2,5

void q4 (int *a, int n) {

int cont;

for (int i = 0; i < n; i++) {

cont = 0;

for (int j = 0; j < n; j++) {

if (a[i] == a[j]) cont++;

if (cont > n/2) {

printf ("Elemento maioritário = %d\n", a[j]);

return;

}

}

}

printf ("Vetor não possui elemento maioritário");

}



DOM	SEG	TER	QUA	QUI	SEX	SÁB
DOM	LUN	MAR	MIÉ	JUE	VIE	SÁB

Ou poderia ser feita fazendo uso de uma variável sinalizadora:

```
void q4 (int *a, int n) {
```

```
    int cont, sinalizador = 0;
```

```
    for (int i = 0; i < n; i++) {
```

```
        cont = 0;
```

```
        for (int j = 0; j < n; j++) {
```

```
            if (a[i] == a[j]) cont++;
```

```
            if (cont > n / 2) {
```

```
                printf ("Elemento maioritário = %d \n", a[i]);
```

```
                sinalizador = -1;
```

```
                break;
```

```
}
```

```
    if (sinalizador == -1) break;
```

```
} if (sinalizador != -1) printf ("Vetor não possui elemento maioritário. \n");
```