Exercícios de Algoritmos II

Prof. Eduardo Rosalém Marcelino

Última atualização: Maio/2018

Livros que temos em nosso acervo com bons exercícios de programação, além de conteúdo:

- Anita Lopes; Guto Garcia. Introdução à Programação. 500 algoritmos resolvidos. Elsevier, 2002
- Sandra Puga; Gerson Rissetti. Lógica de programação e estruturas de dados com aplicações em Java. Pearson Prentice Hall, 2009
- Ana Fernanda Gomes Ascencio. Lógica de Programação com Pascal. Makron Books, 1999.
- Victorine Viviane Mizrahi. Treinamento em Linguagem C. Pearson Prentice Hall, 2008.
- MANZANO, J. A. N. G & OLIVEIRA, J. F. Algoritmos: Lógica para Desenvolvimento de Programação de Computadores. 14 ed. São Paulo:Érica, 2002.

MÉTODOS

1. Primeiro Nome

Faça um programa que tenha um método que receba um nome completo ou não. Retorne então apenas o primeiro nome. No programa principal vc deverá solicitar o nome de 2 pessoas e executar este método.

```
static  string RetornaPrimeiroNome(string Nomecompleto)
{
   return ???
}
```

2. Par Impar

Faça um programa que tenha uma método que receba um número e devolva um String dizendo se o número é Par ou Impar.

3. Numero Primo

Faça um programa que tenha uma método que receba um número e devolva um String dizendo se o número é PRIMO.

4. Vogais

Faça um programa que tenha uma método void que recebe um nome e o objetivo do método é escrever no vídeo apenas as vogais deste nome.

5. Nome Invertido

Faça um programa que solicite um texto qualquer e então um método chamado irá devolver o texto recebido como parâmetro invertido. ex: nariz r= ziran

6. Três Métodos

Faça um programa que tenha 3 métodos :

UltimaLetra -> recebe um texto e exibe a última letra dele.

TrocaCaracter -> recebe um texto, o caractere original e o novo caractere.

Ele substituirá todo os caracteres originais pelo novo caractere e deverá devolver o novo texto.

Iniciais -> recebe um texto e retorna suas iniciais.
Faça o programa principal de forma que se possa testar todos estes métodos

7. Verifica Espaço

Faça um método cujo objetivo é verificar se existe espaco (' ') em um texto informado via parâmetro. Ele deverá retornar True se houver espaço e False se não houver. Faça o programa principal para testar este método.

8. Valida E-mail

Faça um método cujo objetivo é validar um e-mail informado via parâmetro. Ela deverá retornar TRUE se o e-mail for válido ou FALSE caso contrário. Um e-mail válido é aquele que possui um @ (arroba) e um . (ponto) em seu conteúdo, mas não inicia ou termina com eles.

9. Tabuada

Faça um método void que imprima a tabuada de um número. Faça o programa principal para testá-la.

10. Menor valor

Faça um método que receba um vetor de inteiros e ele deverá devolver qual o menor valor no vetor.

11. Soma

Faça um método que receba um vetor de inteiros e ele deverá devolver a soma dos elementos do vetor.

12. Media

Faça um método que receba um vetor de inteiros e ele deverá devolver a média dos elementos do vetor.

Depuração:

13. Opção válida

O programa a seguir deveria solicitar uma opção e apenas os números 1, 2 e 3 deveriam ser aceitos, porém mesmo digitando um desses valores o programa continua sem deixar sair da validação.

```
static void Main(string[] args)
{
    int opcao;
    do
    {
        Console.WriteLine("Digite uma das 3 opções: \n1-> Calcular, " +
            "\n2-> Listar e \n3-> Sair");
        opcao = Convert.ToInt32(Console.ReadLine());
    }
    while (opcao != 1 || opcao != 2 || opcao != 3);

    Console.WriteLine("Você escolheu a opção: {0}", opcao);
    Console.ReadLine();
}
```

14. Testador de letras maiúsculas

O programa a seguir deveria solicitar um texto ao usuário e, após analisa-lo, informar se há ao menos uma letra maiúscula ou se não há nenhuma letra em maiúsculo. O problema é que ele está exibindo várias mensagens na tela sem ainda ter terminado de procurar e, em determinado momento, ele dá um erro.

```
static void Main(string[] args)
```

```
    {
        Console.WriteLine("Digite um texto");
        string texto = Console.ReadLine();

        for (int n=0; n<=texto.Length; n++)
        {
            if (texto[n] == texto.ToUpper()[n])
            {
                Console.WriteLine("Existe letra maiúscula");
            }
            else
            {
                Console.WriteLine("Não existe letra maiúscula");
            }
        }

        Console.ReadLine();
    }
```

15. Gerador de números aleatórios

O objetivo do programa a seguir é gerar 6 números aleatórios. Os números devem ser entre 1 e 6 e não podem se repetir. O problema é que ele entra em um loop infinito. O que há de errado?

```
    static void Main(string[] args)
    {
        Random gerador = new Random();
        int[] vetor = new int[6];

        for (int n = 0; n < vetor.Length; n++)
        {
            bool existe;
            do
            {
                existe = false;
                vetor[n] = gerador.Next(1, 6);

                for (int p = 0; p < n; p++)
                {
                    if (vetor[p] == vetor[n])
                    {
                        existe = true;
                        break;
                    }
                }
            }
            while (existe);
        }

        for (int n = 0; n < vetor.Length; n++)
        {
            Console.WriteLine(vetor[n]);
        }
        Console.ReadLine();
    }
}
```

16. Editor de texto

O objetivo do programa a seguir é ir acumulando os textos digitados cada vez que o usuário pressiona a tecla Enter. Caso o texto digitado seja "sair" o programa deveria salvar todo o texto digitado em um arquivo chamado "dados.txt" e fechar o programa. Caso o programa seja aberto novamente e caso o arquivo "dados.txt" exista, seu conteúdo deveria ser exibido antes que novos textos sejam solicitados novamente. O programa apresenta uma série de problemas. Ele logo de cara já exibe um erro se o arquivo não existir. Ele também não está saindo ao digitar "sair" além não acumular as informações digitadas. Tentei consertar, mas só piorei as coisas e agora ele não está mais nem compilando... :(

```
static void Main(string[] args)
{
    string dadosNoDisco = File.ReadAllText("dados.txt");
    Console.WriteLine(dadosNoDisco);

    string conteudo;
    string palavra;
    do
    {
        palavra = Console.ReadLine();
        if (palavra != "SAIR")
            conteudo = Environment.NewLine + palavra;
    }
    while (palavra != "SAIR");

    File.WriteAllText("dados.txt", conteudo);
}
```

CONTROLE DE EXCEÇÃO

17. Solicite o código, o salário e a data de nascimento de um aluno. Caso ele digite valores incorretos, solicite-os novamente. Utilize o controle de exceção. Ao final, mostre os valores.

18. Aplique o controle de exceção nos exercícios Número Primo, Par Impar, Tabuada, Menor Valor. Em todos eles, aplique a validação de não permitir números <=0.

ESTRUTURAS HETEROGÊNEAS

(Utilize controle de exceção e métodos nas soluções quando necessário ou sempre que possível)

19. Faça um programa para gravar os seguintes dados de uma disciplina:
nome : string; Não pode estar vazio
nota de aprovação : int ; Entre 1 e 10

Ao final exiba os dados no formato nome – nota de aprovação

20. Sistema Acadêmico

Faça um programa que solicite os seguintes dados de um aluno: nome, telefone, idade, valor da mensalidade.
Ao final do programa, exiba os dados cadastrados.
Validações:

Nome deve ser preenchido, telefone deve ter exatamente 9 dígitos (contando o "-")
O primeiro dígito do celular não pode iniciar com 7,8 ou 9
Idade deve estar entre 0 e 150

mensalidade deve ser >= 0.

21. Funcionários

Faça um programa que armazene até 30 funcionários, guardando o nome e o salário.
Ao final, exiba: O maior salário, o menor salário, a soma dos salários e a média salarial.

22. Caixa

Faça um programa para armazenar o caixa de uma empresa.

O caixa é registrado diariamente informando se:

* DATA (DD/MM/AA) datetime
* Total de Entradas (>=0) double
* Total de Saídas (>= 0) double

Permita a digitação de um caixa de até 30 dias.

O usuário poderá parar a qualquer momento.
Ao termino da digitação, exiba um menu com as seguintes opções:

1-) Exibir o caixa no formato : DATA - Entradas - Saídas
2-) Exibir o Total geral de entradas
3-) Exibir o Total geral de saídas
4-) Exibir o saldo : Total de entradas - Total de Saídas
5-) Exibir o caixa por mês. Solicite o mês e exiba somente (MM) os dados pertinentes no mesmo formato da opção 1.
7-) Sair.

23. Automóvel

Crie uma estrutura (MODELO) para guardar o modelo de um automóvel:
string marca (obrigatório)
string tipo (obrigatório)

Crie outra estrutura (CARRO) para guardar os dados de um determinado automóvel:
string placa (obrigatório 3 letras, um hífen, 4 números: EX: XXX-9999)
int ano de fabricação >= 2000
MODELO modelo

Crie um vetor para cadastrar até 10 carros. Pergunte se o usuário deseja parar o cadastro após cadastrar um carro.
Ao final, exiba os dados em vídeo.

24. Faça um método que receba um vetor de inteiros e ele deverá devolver:

- a média dos elementos do vetor.
- a soma
- o maior valor
- o menor valor

Faça duas versões do método: uma que devolve uma estrutura e outra que devolve por parâmetro de referência.

25. Crie um método que devolva 3 informações sobre um nome:

- A quantidade de vogais do nome
- O primeiro nome do nome informado
- Uma versão do nome com apenas a primeira letra de cada palavra em maiúsculo

26. Faça seu próprio Int.TryParse.

O seu método deverá receber um número no formato string e uma variável de saída. Retorne true se o valor informado for um inteiro válido e false caso contrário, além de devolver por referência o resultado da conversão.

27. Faça um método que valide um CPF.

O método deverá dizer se o CPF informado é valido ou não. Seu método deverá ser capaz de aceitar CPFs com ou sem formatação (ponto e traço). Devolva em um parâmetro de referência uma versão do CPF formatado com ponto e traço.

No Brasil existe o CPF (Cadastro de Pessoas Físicas) que serve para identificar cada indivíduo no país. O número do CPF é composto de 11 dígitos, sendo os dois últimos os dígitos de verificação. A fórmula para verificar a validade do número do CPF é simples e é explicada abaixo:

Vamos tomar como exemplo o número de CPF **123.456.789-09**

● Calculando o 1º Dígito Verificador
Primeiro calculamos a soma da multiplicação dos 9 primeiros dígitos por **10, 9, 8, ... , 3, 2,** respectivamente. Ou seja:

Soma = (1*10) + (2*9) + ... + (8*3) + (9*2)

Depois calculamos o resto da divisão de soma por 11:
Resultado = Soma **módulo** 11.

Agora analisamos **Resultado**:

- Se **Resultado** for igual a **0 ou 1**, então o 1º dígito verificador deve ser **0**;
- Caso contrário, o 1º dígito verificador é calculado com a fórmula: **11** - **Resultado**.

● Calculando o 2º Dígito Verificador
Primeiro calculamos a soma da multiplicação dos 9 primeiros dígitos por **11, 10, 9, ... , 4, 3,** respectivamente e em seguida somamos com **(Digito1*2)**, sendo que **Digito1** é o valor encontrado para o 1º dígito verificador. Ou seja:

Soma = (1*11) + (2*10) + ... + (8*4) + (9*3) + (Digito1*2)

Depois calculamos o resto da divisão de soma por 11:
Resultado = Soma **módulo** 11.

Agora analisamos **Resultado**:

- Se **Resultado** for igual a **0 ou 1**, então o 2º dígito verificador deve ser **0**;
- Caso contrário, o 2º dígito verificador é calculado com a fórmula: **11** - **Resultado**.

No nosso exemplo (**123.456.789-09**) o número é válido.

28. Faça um programa que leia um arquivo texto chamado dados.txt. Jogue esses dados em uma estrutura. Exiba as seguintes informações acerca do arquivo. Obs: A primeira linha do arquivo indica o cabeçalho e deve ser ignorada.

Conteúdo do arquivo (salve os dados em um arquivo chamado dados.txt)

```
Código; nome;  preco;  categoria;
1;      Papel higiênico;1;      Higiene
2;      Arroz;  2;      Grãos
3;      Feijão; 3;      Grãos
4;      Refrigerante;   1;      Bebida
5;      Suco;   2;      Bebida
6;      Salgadinho;     3;      Guloseima
7;      Sorvete;1;      Guloseima
8;      Chocolate;      2;      Guloseima
9;      Batata; 3;      Legumes
10;     Beterraba;      1;      Legumes
11;     Sabão;  2;      Higiene
12;     Cenoura;        3;      Legumes
13;     Milho;  1;      Grãos
```

Após ler e processar o arquivo, exiba um menu com as seguintes opções:

1 – Exibir todos os dados do arquivo no formato:
    Código: XXXX
    Descrição: XXXX
    Preço: XXXX
    Categoria: XXXX
2 – Exibir as categorias dos produtos, sem repetir.
3 – Exibir as categorias e a soma dos produtos.
4 – Solicitar uma categoria e exibir os produtos cadastrados naquela categoria, no mesmo formato que solicitado na opção 1.
5 – Exiba os produtos da seguinte Forma:
Exiba a letra A e então mostre todos os produtos que começam com esta letra.
Depois exiba a letra B e faça o mesmo.
Faça isso exibindo até a letra Z.

EM DUPLA

Postar nos 2 integrantes!!!

Execute o programa anexo. Ele vai gerar um arquivo com 100.000.000 milhões de números.

Objetivo: Dizer quais os números distintos foram gerados e a quantidade que foi gerada de cada número, Ordenados em ordem crescente.

Escreva no arquivo:

numero - qtde de vezes que ele aparece

salvar em um arquivo texto a resposta
Usar exclusivamente estruturas de dados da aula
Não pode usar nada pronto do C#

Você pode utilizar qualquer estrutura de dados ou algoritmos que aprendeu para realizar o processo e armazenar os números.

Quanto tempo seu algoritmo demorou para processar?

Para calcular o tempo que demorou para processar o algoritmo, faça assim:

```
DateTime inicio = DateTime.Now; // guarda a hora inicial

do
{
//coloque o seu algoritmo de gerar números aqui...
//salvar os dados em .txt
}
while (true);

TimeSpan tempo = DateTime.Now.Subtract(inicio); // calcula o total de segundos que demorou o processo

Console.WriteLine(tempo.TotalSeconds);
```

Considere no seu tempo o tempo gasto para salvar o arq. texto!

Com base num vetor de inteiros, onde cada posição do vetor guarda o RA de um aluno, deseja-se fazer uma sistema que crie duplas de forma aleatória.
Caso o número de alunos seja impar, poderá haver 1 trio.

Faça um método que receba um vetor de inteiros que representam RAs de alunos e devolva um novo vetor de string contendo as duplas.

O retorno deve ser os códigos dos integrantes, separados por vírgula.

ex:

```
string[] MontaDuplas ( int[] alunos)
{

}
```

Exemplo da resposta para uma lista impar de alunos.

```
[0] = "7,21";
[1] = "1,3";
[2] = "4,12";
[3] = "14,5,9";
```