Modularização

Ricardo Frohlich da Silva

Modularização

- Os algoritmos geralmente são desenvolvidos para solucionar problemas complexos.
- A solução destes problemas geram algoritmos extensos e complexos que podem ser divididos em problemas menores.
- A solução dos problemas menores requer algoritmos mais simples.
- A combinação destas soluções menores leva à solução do problema inicial (complexo).

Modularização

- A divisão de um problema complexo em subproblemas pode ser feita de maneira sucessiva, de forma a simplificar sua solução pela obtenção de soluções parciais.

- Estas soluções parciais podem ser usadas em diferentes pontos do algoritmo inicial ou por outros algoritmos (ser reutilizada).

Modularização

- Para desenvolver um algoritmo de forma que possa ser modularizado, é preciso:
  - Identificar o programa geral e quais tarefas ele deverá executar;
  - Definir um programa principal, que integra as tarefas a serem executadas;
  - Programar a soluções para cada uma das tarefas identificadas.

Funções

- Formas de modularizar a programação são por meio do uso de sub-rotinas conhecidas como funções.
- Funções são os blocos de construção da linguagem C# e o local onde parte da atividade do programa ocorre.
- Todo programa em C# é composto por pelo menos uma função: *main* (até mesmo os que estão em nível superior, porém é abstraído esta informação)

Funções

- Declaração de uma função:

```
static tipo nome_função(tipo1 nome1, tipo2 nome2, ..., tipo3 nome3){
    Corpo da função; //algoritmo
    return ;
}
```

Funções

- *Static* define que esta função é estática e não necessita da utilização de conceitos de orientação a objetos (vamos ver mais sobre isso)

- *tipo* define o tipo de valor que o comando *return* da função irá devolver. Se nenhum tipo é definido, o compilador assume que um valor inteiro será devolvido. O uso de *void* é indicado para funções que não retornam valores, pois impede o uso acidental destas em expressões.

- *nome_função* identifica o bloco de comandos, de forma que possa ser referência em qualquer outra parte do programa.

- *lista_parâmetros* é uma lista tipo nome separada por vírgulas: tipo1 nome1, tipo2 nome2, ..., tipo3 nome3 . Uma função pode não ter parâmetros.

- *corpo da função* identifica os comandos a serem executados. É onde está implementado o algoritmo da função.

- *return* indica o dado que será retornado pela função, para a função chamadora.

Funções

- Mas antes de aprofundar, vamos fazer alguns exemplos simples para compreender como funciona a criação e a chamada de uma função:

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Seja bem vindo a um exemplo com funções!");
        apresentaMensagem();
    }

    static void apresentaMensagem()
    {
        Console.WriteLine("Estou na função!");
    }
}
```

Funções

- Jogo rápido:
  - Fazer uma função que leia dois números inteiros e apresente a soma:

Funções

- Jogo rápido:
  - Fazer uma função que leia dois números inteiros e apresente a soma:

```
static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Jogo rápido:
  - Fazer uma função que leia dois números inteiros e apresente a soma:
    - E a chamada na main?

Funções

- Jogo rápido:
  - Fazer uma função que leia dois números inteiros e apresente a soma:
    - E a chamada na main?

```
static void Main(string[] args)
{
    ler2NumSoma();
}
```

Funções

- Jogo rápido:
  - E teria como fazer em duas funções?
  - Uma que efetua a leitura e outro que efetua o cálculo?
    - Sim, mas com o que vimos até agora, precisaremos compreender sobre a variável global.

Funções

- Dúvida: As variáveis selecionadas podem ser usadas na main?
- Façam um teste!

```
static void Main(string[] args)
{
    ler2NumSoma();
}

static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Dúvida: As variáveis selecionadas podem ser usadas na main?
- Façam um teste!
  - Não, né?

```
static void Main(string[] args)
{
    ler2NumSoma();
}

static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Dúvida: As variáveis selecionadas podem ser usadas na função ler2NumSoma?
- Façam um teste!

```
static void Main(string[] args)
{
    int x, y;
    ler2NumSoma();
}

static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Dúvida: As variáveis selecionadas podem ser usadas na função ler2NumSoma?
- Façam um teste!
  - Não, né?

```
static void Main(string[] args)
{
    int x, y;
    ler2NumSoma();
}

static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Dúvida: As variáveis selecionadas podem ser usadas na main?
- Façam um teste!
  - Não, né?
  - Não podem pois elas estão dentro do escopo da função, ou seja, são **variáveis locais!**

```
static void Main(string[] args)
{
    int x, y;
    ler2NumSoma();
}

static void ler2NumSoma()
{
    int a, b, soma;
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
    soma = a + b;
    Console.WriteLine("A soma = "+soma);
}
```

Funções

- Variáveis locais:
  - São declaradas dentro de um bloco de código específico, como um método, uma função, um loop, ou qualquer outro escopo delimitado por chaves {}.
  - Elas existem apenas dentro desse escopo onde foram declaradas e não podem ser acessadas fora dele.
  - São úteis quando você deseja armazenar temporariamente valores para cálculos ou operações dentro de um método ou função.
  - São descartadas automaticamente quando o controle de fluxo sai do escopo em que foram declaradas, liberando memória.

Funções

- Variáveis Globais:
  - São declaradas fora de qualquer método ou função e, portanto, têm um escopo global.
  - Podem ser acessadas de qualquer lugar no programa, em qualquer método ou função.
  - Têm uma vida útil mais longa, pois existem durante toda a execução do programa.
  - *No entanto, o uso excessivo de variáveis globais pode dificultar a manutenção e depuração do código, pois podem ser modificadas de qualquer lugar.*

Funções

```
internal class Program
{
    static int a, b, soma;
    static void Main(string[] args)
    {
        ler2NumSoma();
    }

    static void ler2NumSoma()
    {

        Console.WriteLine("Digite um valor para A: ");
        a = int.Parse(Console.ReadLine());
        Console.WriteLine("Digite um valor para B: ");
        b = int.Parse(Console.ReadLine());
        soma = a + b;
        Console.WriteLine("A soma = "+soma);
    }
}
```

```
internal class Program
{
    static int a, b, soma;
    static void Main(string[] args)
    {
        ler2NumSoma();
    }

    static void ler2NumSoma()
    {

        Console.WriteLine("Digite um valor para A: ");
        a = int.Parse(Console.ReadLine());
        Console.WriteLine("Digite um valor para B: ");
        b = int.Parse(Console.ReadLine());
        soma = a + b;
        Console.WriteLine("A soma = "+soma);
    }
}
```

Funções

- Agora, vamos fazer duas funções:
  - Uma função para fazer a leitura de dois números
  - Outra para somar dois números

Funções

- Agora, vamos fazer duas funções:
  - Uma função para fazer a leitura de dois números

```
static int a, b;
static void Main(string[] args)
{
    ler2Num();
}

static void ler2Num()
{
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
}
```

Funções

```
static int a, b;
static void Main(string[] args)
{
    ler2Num();
}

static void ler2Num()
{
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
}
```

Funções

- Agora, vamos fazer duas funções:
  - Outra para somar os dois números:

```
static int a, b;
static void Main(string[] args)
{
    ler2Num();
}

static void somar()
{
    int soma;
    soma = a + b;
    Console.WriteLine("A soma = " + soma);
}
```

Funções

```
static int a, b;
static void Main(string[] args)
{
    ler2Num();
}

static void somar()
{
    int soma;
    soma = a + b;
    Console.WriteLine("A soma = " + soma);
}
```

```
static int a, b;
static void Main(string[] args)
{
    ler2Num();
}

static void somar()
{
    int soma;
    soma = a + b;
    Console.WriteLine("A soma = " + soma);
}

static void ler2Num()
{
    Console.WriteLine("Digite um valor para A: ");
    a = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = int.Parse(Console.ReadLine());
}
```

Atividade

- 1) Fazer uma calculadora das 4 operações (soma, subtração, multiplicação e divisão)
  - Para isso, precisaremos aceitar números com vírgula.
- 2) Fazer programa que tenha duas funções. i) para ler um vetor de 10 elementos (variável global) e ii) para verificar e apresentar a quantidade de elementos pares que este vetor possui.

Passagem de parâmetros

- É possível enviar para a função, no momento de sua chamada, valores para serem utilizados em seu processamento.
  - Estes valores são chamados de argumentos.
- A especificação da quantidade e do tipo dos argumentos é feita na lista de parâmetros colocada após o nome da função, entre parênteses.
  - tipo nome_função(tipo1 nome1, tipo2 nome2, ..., tipo3 nome3){
    - Corpo da função; //algoritmo;
  - }

Passagem de parâmetros

- Os parâmetros da função são variáveis criadas quando a função é chamada e eliminadas no retorno à função chamadora (são variáveis locais, são visíveis apenas na função e existem enquanto a função estiver sendo executada).
    - Essas variáveis são inicializadas com os valores passados como argumentos.
- Os argumentos podem ser constantes, variáveis, expressões, ou inclusive outras funções que retornem valores.

Passagem de parâmetros

- No exemplo abaixo, há diferentes formas de passar valores à função:

```
static void Main(string[] args)
{
    f1(5);
    f2(7, 'c');
}

static void f1(int x)
{
    Console.WriteLine("Foi enviado "+x+" por parâmetro!");
}

static void f2(int x, char y)
{
    Console.WriteLine("Foi enviado "+x+ " e "+y+" por parâmetro!");
}
```

```
static void Main(string[] args)
{
    f1(5);
    f2(7, 'c');
}

static void f1(int x)
{
    Console.WriteLine("Foi enviado "+x+" por parâmetro!");
}

static void f2(int x, char y)
{
    Console.WriteLine("Foi enviado "+x+ " e "+y+" por parâmetro!");
}
```

Passagem de parâmetros

- Jogo rápido:
  - Vamos refazer utilizando parâmetros e sem variáveis globais o exercício 1 da atividade da aula passada?

Passagem de parâmetros

```
static void Main(string[] args)
{
    double a, b;
    Console.WriteLine("Digite um valor para A: ");
    a = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = double.Parse(Console.ReadLine());
    somar(a, b);
    subtrair(a, b);
    multiplicar(a, b);
    dividir(a, b);
}
```

Passagem de parâmetros

```
static void somar(double a, double b)
{
    double soma;
    soma = a + b;
    Console.WriteLine("A soma = " + soma);
}

static void subtrair(double a, double b)
{
    double sub;
    sub = a - b;
    Console.WriteLine("A subtração = " + sub);
}
```

Passagem de parâmetros

```
static void multiplicar(double a, double b)
{
    double multi;
    multi = a * b;
    Console.WriteLine("A multiplicação = " + multi);
}

static void dividir(double a, double b)
{
    double div;
    div = a / b;
    Console.WriteLine("A divisão = " + div);
}
```

Retorno de valores

- Uma função pode retornar à função chamadora quando termina sua execução.
- Neste retorno, a função pode devolver à função chamadora um valor (int, float, double, char ou algum tipo de dado heterogêneo).
- O retorno pode acontecer antes de se chegar ao finalizador da função (‘}’), se for usado o return em qualquer ponto da função.
- Quando a declaração return é usada, a função é terminada e o controle volta à função chamadora, independentemente da existência de outros comandos não executados.

Retorno de valores

```
static int f1()
{
    Console.WriteLine("Estou na F1!");
    return 1;
}
static double f2()
{
    Console.WriteLine("Estou na F2!");
    return 2.0;
}

static char f3()
{
    return 'a';
    Console.WriteLine("Estou na F3!");
}
```

```
static int f1()
{
    Console.WriteLine("Estou na F1!");
    return 1;
}
static double f2()
{
    Console.WriteLine("Estou na F2!");
    return 2.0;
}

static char f3()
{
    return 'a';
    Console.WriteLine("Estou na F3!");
}
```

Retorno de valores

```
static void Main(string[] args)
{
    f1();
    f2();
    f3();
}
```

Retorno de valores

- Mas e pra onde foram os valores do return??

Retorno de valores

```
static void Main(string[] args)
{
    int n1;
    double n2;
    char c;
    n1 = f1();
    Console.WriteLine("Retorno da F1 = " + n1);
    n2 = f2();
    Console.WriteLine("Retorno da F2 = " + n2);
    c = f3();
    Console.WriteLine("Retorno da F3 = " + c);
}
```

Retorno de valores

- Uma função pode gerar um (único) valor de retorno necessário para o funcionamento do resto do programa.
- O valor a ser retornado deve ser colocado após a declaração return.
- O tipo do dado retornado deve ser identificado antes do nome da função.

Retorno de valores

- Jogo rápido:
  - Agora, no exemplo anterior, da calculadora, vamos fazer que o valor da soma, subtração, multiplicação e divisão seja retornada a main?
    - Difícil?
      - Sim? Tenho uma dica no próximo slide!
      - Não? Bom trabalho, você tem 5 minutos!

Retorno de valores

```
static void Main(string[] args)
{
    double a, b, resultado;
    Console.WriteLine("Digite um valor para A: ");
    a = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = double.Parse(Console.ReadLine());
    resultado = somar(a, b);
    Console.WriteLine("A soma = " + resultado);

}

static double somar(double a, double b)
{
    double soma;
    soma = a + b;
    return soma;
}
```

```
static void Main(string[] args)
{
    double a, b, resultado;
    Console.WriteLine("Digite um valor para A: ");
    a = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = double.Parse(Console.ReadLine());
    resultado = somar(a, b);
    Console.WriteLine("A soma = " + resultado);

}

static double somar(double a, double b)
{
    double soma;
    soma = a + b;
    return soma;
}
```

Retorno de valores

```
static double somar(double a, double b)
{
    double soma;
    soma = a + b;
    return soma;
}

static double subtrair(double a, double b)
{
    double sub;
    sub = a - b;
    return sub;
}
```

Retorno de valores

```
static double multiplicar(double a, double b)
{
    double multi;
    multi = a * b;
    return multi;
}

static double dividir(double a, double b)
{
    double div;
    div = a / b;
    return div;
}
```

Retorno de valores

```
static void Main(string[] args)
{
    double a, b, resultado;
    Console.WriteLine("Digite um valor para A: ");
    a = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = double.Parse(Console.ReadLine());
    resultado = somar(a, b);
    Console.WriteLine("A soma = " + resultado);

    resultado = subtrair(a, b);
    Console.WriteLine("A subtração = " + resultado);

    resultado = multiplicar(a, b);
    Console.WriteLine("A multiplicação = " + resultado);

    resultado = dividir(a, b);
    Console.WriteLine("A divisão = " + resultado);

}
```

```
static void Main(string[] args)
{
    double a, b, resultado;
    Console.WriteLine("Digite um valor para A: ");
    a = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite um valor para B: ");
    b = double.Parse(Console.ReadLine());
    resultado = somar(a, b);
    Console.WriteLine("A soma = " + resultado);

    resultado = subtrair(a, b);
    Console.WriteLine("A subtração = " + resultado);

    resultado = multiplicar(a, b);
    Console.WriteLine("A multiplicação = " + resultado);

    resultado = dividir(a, b);
    Console.WriteLine("A divisão = " + resultado);

}
```

Retorno de valores

- Jogo rápido:
  - Faça no exemplo da calculadora com que tenha uma função que leia um valor double e retorne.

Retorno de valores

```
static void Main(string[] args)
{
    double a, b, resultado;
    a = lerDouble();
    b = lerDouble();
    resultado = somar(a, b);
    Console.WriteLine("A soma = " + resultado);

    resultado = subtrair(a, b);
    Console.WriteLine("A subtração = " + resultado);

    resultado = multiplicar(a, b);
    Console.WriteLine("A multiplicação = " + resultado);

    resultado = dividir(a, b);
    Console.WriteLine("A divisão = " + resultado);

}
```

Retorno de valores

```
private static double lerDouble()
{
    double n;
    Console.WriteLine("Digite um valor double: ");
    n = double.Parse(Console.ReadLine());
    return n;
}
```

Atividades

- 1) Ler um número inteiro e passar por parâmetro para uma função e 0, se o número é par, ou 1, se o número é ímpar.

- 2) Escrever um programa com uma função que receba por parâmetro dois números (n1 e n2) e retorna a soma dos números inteiros que existem entre n1 e n2 (inclusive ambos). Apresente o resultado na main.

- Desafio: Ler um vetor de inteiros com 10 elementos e passar por parâmetro para uma função que verifica e retorna o menor elemento do vetor.