



## Programação Estruturada

Rodrigo Adur  
rodrigoadurti@gmail.com



### Apresentação



- **Professor Rodrigo Adur**
  - **Formação**
    - Bacharelado em Sistemas de Informações (FILC)
    - Especialista em Sistemas de Informações (NCE/UFRJ)
  - **Experiência Profissional**
    - Desenvolvimento de Sistemas
      - Desenvolvimento de biblioteca de componentes
      - Desenvolvimento de framework
    - Analista de Sistemas do SERPRO
      - Arquiteto
      - Líder técnico
      - Programador

Programação Estruturada – Rodrigo Adur

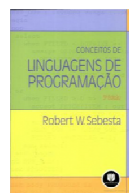
### Estrutura do Módulo



- **Módulo Teórico**
  - **Parte I**
    - Vinculação
    - Verificação de tipos
    - Escopo
  - **Parte II**
    - Tipos de dados
    - Expressões
    - Estruturas de controle
  - **Parte III**
    - Modularização
    - Funções e Procedimentos
    - Coesão e Acoplamento

Programação Estruturada – Rodrigo Adur

### Bibliografia



Conceitos de Linguagem de Programação  
Robert Sebesta



Linguagens de Programação  
Flavio VAREJO

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de Tipos e Escopo

### Vinculação, Verificação de tipos e Escopo



- **Programação Estruturada**
  - A programação estruturada pode ser vista como um subconjunto do **paradigma imperativo**.
  - A programação estruturada fundamenta-se no princípio básico de que um programa deve possuir um único ponto de entrada e um único ponto de saída, sendo desenvolvido por um número muito limitado de estruturas de controle.
  - Tem como proposta a divisão dos programas em um conjunto de subprogramas menores, cada um com seu objetivo específico e bem definido, mais fáceis de implementar e de testar.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Identificadores

- Um identificador é uma cadeia de caracteres definida pelos programadores para identificar alguma entidade de um programa.
- Algumas linguagens de programação são *case sensitive*, ou seja, fazem distinção entre letras maiúsculas e minúsculas.
- Não é permitido usar palavras reservadas como identificadores.
- Como boa prática, os identificadores devem ser formados por nomes significativos que reflitam informações a respeito da entidade referenciada.
  - Exemplo de identificadores: *media, idade, salario*.

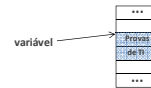
Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Variáveis

- Uma variável de programa é uma abstração de uma célula ou de um conjunto de células de memória do computador.



- Uma variável pode ser caracterizada pelos seguintes atributos: nome, endereço, valor, tipo, tempo de vida e escopo.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Variáveis

- Nome**
  - Normalmente uma variável tem um nome definido pelo programador.
- Endereço**
  - O endereço de uma variável é o mesmo da memória à qual ela está associada.
  - Algumas linguagens de programação permitem que o endereço seja acessado livremente pelo programador.
  - Dizemos que variáveis são sinônimas quando múltiplos identificadores fazem referência ao mesmo endereço.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Variáveis

- Tipo**
  - O tipo de uma variável determina o conjunto de valores que ela pode ter e o conjunto de operações definidas para os valores do tipo.
- Valor**
  - O valor da variável é o conteúdo das células de memória associadas a variável.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Variáveis

- Tempo de vida**
  - O tempo de vida de uma variável corresponde ao período em que ela existe.
- Escopo**
  - O escopo determina a região do programa onde a variável é visível.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Vinculação

- Uma vinculação é uma associação entre entidades de programação como, por exemplo, entre uma variável e seu valor.
- Existem diferentes tipos de vinculações que podem ocorrer em momentos distintos. O momento em que uma vinculação desenvolve-se é chamado de tempo de vinculação.
- Uma vinculação é estática se ocorrer antes do tempo de execução e permanecer inalterada ao longo da execução do programa.
- Uma vinculação é dinâmica se ela ocorrer durante a execução ou puder ser modificada no decorrer da execução de um programa.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculação de Tipos

- Antes que uma variável possa ser referenciada em um programa, ela deve ser vinculada a um tipo de dado.

### ➤ Vinculação estática de tipos

- Na vinculação estática, a declaração de uma variável produz uma vinculação entre um identificador e um tipo.
- Uma declaração de tipos é explícita quando uma instrução em um programa lista nomes de variáveis e especifica que elas são de um tipo particular.

```
int qtdItens = 40;
```

*Código: Exemplo na linguagem C.*

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculação de Tipos

#### ➤ Vinculação estática de tipos

- Uma declaração de tipos é implícita quando usamos um meio de associar variáveis a tipos por convenções padrão em vez de por instruções de declaração.

```
iQtdItens = 40;
```

*Código: Exemplo na linguagem FORTRAN.*

- Tanto as declarações explícitas como as implícitas criam vinculações estáticas a tipos.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculação de Tipos

#### ➤ Vinculação dinâmica de tipos

- Na vinculação dinâmica de tipos, o tipo não é especificado por uma instrução de declaração.
- A variável é vinculada ao tipo do valor que lhe é atribuído em uma instrução de atribuição.

```
qtdItens = 40;
```

*Código: Exemplo na linguagem lua.*

- A principal vantagem dessa abordagem é proporcionar flexibilidade de programação.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculação de Tipos

#### ➤ Vinculação dinâmica de tipos

##### ➤ Desvantagens:

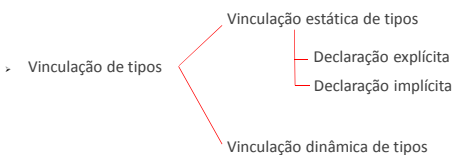
- A capacidade de detecção de erros do compilador é reduzida quando comparada a vinculação de tipos estática.
- O custo para implementar a vinculação dinâmica de atributos é consideravelmente mais alto.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculação de Tipos



Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Vinculações de Armazenamento e Tempo de vida

- Nós chamamos de alocação o processo que toma, de um pool de memória disponível, as células de memória à qual uma variável é vinculada.
- Por outro lado chamamos de desalocação o processo de devolver células de memória desvinculadas de uma variável ao pool de memória disponível.
- O tempo de vida de uma variável é o tempo durante a qual esta é vinculada a uma localização de memória específica.



Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Verificação de Tipos

- A verificação de tipos é uma atividade que assegura que os operandos de um operador sejam de tipos compatíveis.
- Se um operando impróprio for aplicado a um determinado operador, ocorre um erro de tipo.
- Se a vinculação de variáveis a tipos for estática, a verificação de tipos poderá ser feita estaticamente.
- A vinculação dinâmica de tipos requer a verificação dinâmica de tipos.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Tipificação Forte

- Numa linguagem de programação fortemente tipificada erros de tipo sempre são detectados.
- Isso exige que todos os tipos de todos os operandos possam ser determinados durante a compilação ou em tempo de execução.
- A importância da tipificação forte reside em sua capacidade de detectar todos os usos equivocados de variáveis que resultam em erros de tipo.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Escopo de Variáveis

- O escopo de uma variável de programa é a faixa de instruções na qual a variável é visível.
- **Bloco de comandos**
  - Um bloco de comandos é um subprograma ou um trecho de código delimitado por marcadores.
  - Um bloco de comandos delimita o escopo de qualquer variável que ele possa conter.
  - Os blocos de comandos podem ser classificados como: bloco monolítico, blocos não aninhados e blocos aninhados.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Escopo de Variáveis

#### Bloco Monolítico

- Na estrutura monolítica, todo o programa é composto por um único bloco.
- Todas as variáveis tem como escopo de visibilidade o programa inteiro.
- Essa estrutura de blocos não é apropriada para programas grandes.

```
Int x = 10;
Int y = 20;
-
-
-
```

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Escopo de Variáveis

#### Blocos Não-aninhados

- Nessa estrutura, o escopo de visibilidade das variáveis é o bloco onde foram criadas. Essas são denominadas variáveis locais.
- Variáveis criadas fora do bloco são chamadas de variáveis globais, uma vez que seu escopo de visibilidade é todo o programa.

```
Int x = 10;
Int y = 20;
Int z = 30;
Int w = 40;
```

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### Escopo de Variáveis

#### Blocos Aninhados

- Nessa estrutura, qualquer bloco pode ser aninhado dentro de outro bloco e inserido em qualquer lugar convenientemente.
- Variáveis podem ser declaradas dentro de qualquer bloco.

```
Int x = 10;
Int y = 20;
Int z = 20;
Int y = 20;
Int z = 20;
```

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Escopo de Variáveis

#### ➤ Escopo Estático

- Nesse método o escopo de uma variável é definido em tempo de compilação.
- Quando uma referência a uma variável é encontrada pelo compilador, os atributos da variável são determinados localizando-se as instruções nas quais ela é declarada.
- A procura pela declaração da variável inicia-se no bloco onde a referência ocorre. Se nenhuma declaração for encontrada, então a mesma continua na próxima unidade envolvente (pai estático) e assim por diante.

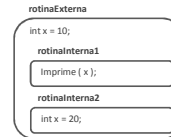
Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Escopo de Variáveis

#### ➤ Escopo Estático



**Cenário 1:** O que seria exibido caso a rotinaExterna fizesse uma chamada a rotinaInterna1?

A rotinaInterna1 imprimiria: 10.

**Cenário 2:** O que seria exibido caso a rotinaInterna2 fizesse uma chamada a rotinaInterna1?

A rotinaInterna1 imprimiria: 10.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Escopo de Variáveis

#### ➤ Escopo Dinâmico

- Nesse método o escopo de uma variável é definido em tempo de execução.
- O ambiente de vinculação é determinado em função das sequências das chamadas dos módulos do programa.
- A procura pela variável inicia-se no bloco onde a referência ocorre. Se nenhuma declaração for encontrada, então a busca continua pelo procedimento chamador (pai dinâmico) e assim por diante.

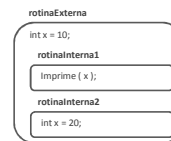
Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Escopo de Variáveis

#### ➤ Escopo Dinâmico



**Cenário 1:** O que seria exibido caso a rotinaExterna fizesse uma chamada a rotinaInterna1?

A rotinaInterna1 imprimiria: 10.

**Cenário 2:** O que seria exibido caso a rotinaInterna2 fizesse uma chamada a rotinaInterna1?

A rotinaInterna1 imprimiria: 20.

Programação Estruturada – Rodrigo Adur

## Vinculação, Verificação de tipos e Escopo



### ➤ Constantes

- Uma constante nomeada é uma variável vinculada a um valor que necessita ser armazenado durante uma computação e não deve ser modificado ao longo do programa.

```
const float PI = 3.14159;
```

Código: Exemplo na linguagem C++.

- As constantes nomeadas são úteis para aumentar a legibilidade e a confiabilidade dos programas.

- Constantes podem ser usadas diretamente em uma expressão:

```
double media = (nota1 + nota2 + nota3) / 3;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Questão 01



[2011 - CESPE - EBC - Analista de Sistemas]

A respeito de programação estruturada, julgue os itens seguintes.

Um programa que possui somente um ponto de entrada e somente um ponto de saída pode ser considerado estruturado.

Certo    Errado

Programação Estruturada – Rodrigo Adur

## Questão 01



[2011 - CESPE - EBC - Analista de Sistemas]

A respeito de programação estruturada, julgue os itens seguintes.

Um programa que possui somente um ponto de entrada e somente um ponto de saída pode ser considerado estruturado.

➡ Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 02



[2013 - CESGRANRIO - FINEP – Desenvolvimento de Sistemas]

Uma linguagem de programação não exige que as variáveis tenham seu tipo definido. Porém, sempre detecta erros de tipo, determinando o tipo de todos os operandos em tempo de execução. Isso a caracteriza como uma linguagem

- (A) sem tipos
- (B) fracamente tipificada
- (C) quase fortemente tipificada
- (D) fortemente tipificada
- (E) de tipos estáticos

Programação Estruturada – Rodrigo Adur

## Questão 02



[2013 - CESGRANRIO - FINEP – Desenvolvimento de Sistemas]

Uma linguagem de programação não exige que as variáveis tenham seu tipo definido. Porém, sempre detecta erros de tipo, determinando o tipo de todos os operandos em tempo de execução. Isso a caracteriza como uma linguagem

- (A) sem tipos
- (B) fracamente tipificada
- (C) quase fortemente tipificada
- ➡ (D) fortemente tipificada
- (E) de tipos estáticos

Programação Estruturada – Rodrigo Adur

## Questão 03



[2010 - FGV - DETRAN-RN - Programador de computador]

Assinale a alternativa que tipifica o item "1.23" na seguinte fórmula:

**RESULTADO = 1.23 \* ENTRADA**

- (A) Variável.
- (B) Vetor.
- (C) Operador.
- (D) Constante.
- (E) Ponteiro.

Programação Estruturada – Rodrigo Adur

## Questão 03



[2010 - FGV - DETRAN-RN - Programador de computador]

Assinale a alternativa que tipifica o item "1.23" na seguinte fórmula:

**RESULTADO = 1.23 \* ENTRADA**

- (A) Variável.
- (B) Vetor.
- (C) Operador.
- ➡ (D) Constante.
- (E) Ponteiro.

Programação Estruturada – Rodrigo Adur

## Gabarito



01 – Certo

02 – D

03 – D

Programação Estruturada – Rodrigo Adur

## Tipos de dados, Expressões e Estruturas de controle

### Tipos de dados



#### ➤ Conceitos Gerais

- As linguagens de programação utilizam os conceitos de tipos de dados para permitir a representação de valores em programas.
- Um tipo de dado define um conjunto de valores válidos que ele pode representar.
- Cada linguagem adota um conjunto próprio de valores e tipos para permitir a representação de dados.

Programação Estruturada – Rodrigo Adur

### Tipos de dados



#### ➤ Tipos de Dados Primitivos

- Os tipos primitivos são a base de todo sistema de tipos de uma linguagem, pois a partir deles é que todos os demais tipos podem ser construídos.
- Nós contamos com os seguintes tipos primitivos:
  - Tipos numéricos
    - Inteiro
    - Ponto-flutuante
  - Tipo booleano
  - Tipo de caractere

Programação Estruturada – Rodrigo Adur

### Tipos de dados



#### ➤ Tipos de Dados Primitivos

##### ➤ Inteiro

- Um tipo inteiro corresponde a um intervalo do conjunto dos números inteiros.
- Em geral existem vários tipos inteiros numa mesma linguagem de programação.

```
int idade = 30;
unsigned contador = 1000;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

### Tipos de dados



#### ➤ Tipos de Dados Primitivos

##### ➤ Ponto-flutuante

- Os tipos de dados com ponto-flutuante modelam os número reais.
- Valores de ponto flutuante são representados com uma notação que combina representações binárias de frações e expoentes.
- A maioria das linguagens inclui dois tipos de ponto-flutuante chamados de *float* e *double*.

Programação Estruturada – Rodrigo Adur

### Tipos de dados



#### ➤ Tipos de Dados Primitivos

##### ➤ Ponto-flutuante

- O *float* é normalmente armazenado em 32 bits de memória.
- O *double* é fornecido para situações em que se necessitam de partes fracionárias maiores. Ele geralmente ocupa o dobro de bits do float.

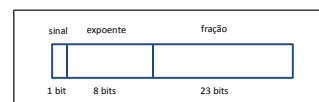


Figura: Padrão IEEE 754 para ponto-flutuante de precisão simples.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipos de Dados Primitivos

#### Ponto-flutuante

```
float salario = 5500.00;
double totalSalarios = 0.0;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipos de Dados Primitivos

#### Booleanos

- Esse tipo de dado representa apenas dois valores: um correspondente a *verdadeiro* e outro a *falso*.
- Esses tipos são geralmente usados como resultados de expressões condicionais ou como variáveis identificadoras de estado.

```
boolean executouComSucesso = true;
```

Código: Exemplo na linguagem Java.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipos de Dados Primitivos

#### Caractere

- Os dados de caracteres são armazenados nos computadores como codificações numéricas.
- A tabela de codificação mais comumente usada é a ASCII (American Standard Code for Information Interchange).

Char	Dec
A	65
B	66
C	67
...	...

Figura: Tabela ASCII

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipos de Dados Primitivos

#### Caractere

```
char a1 = 'A';
char a2 = 65;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipo String de Caracteres

- O tipo string de caracteres é aquele cujos valores consistem em sequências de caracteres.
- Formas de se implementar strings:
  - Estática** - o tamanho da string é definido em tempo de compilação e não é modificado durante a execução.
  - Dinâmica limitada** - o tamanho máximo da string é definido em tempo de compilação, mas pode variar durante a execução até o máximo especificado.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipo String de Caracteres

- Formas de se implementar strings:
  - Dinâmica** - o tamanho da string pode variar livremente durante a execução.

```
char *s = malloc( 15 * sizeof(char));
s[0] = 'C';
s[1] = 'O';
s[2] = 'N';
s[3] = 'C';
s[4] = 'U';
s[5] = 'R';
s[6] = 'S';
s[7] = 'O';
s[8] = 'S';
-> s[9] = '\0';
s[10] = 'O';
s[11] = 'E';
s[12] = 'T';
s[13] = 'I';
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur



## Tipos de dados



### ➤ Tipo Enumeração

- Um tipo enumeração é aquele em que todos os valores possíveis são enumerados na definição.

```
enum DIAS_DA_SEMANA {SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};
```

Código: Exemplo na linguagem C.

- Os tipos enumerados apresentam vantagem tanto em termos de legibilidade como de confiabilidade.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### ➤ Tipo Enumeração

- Cada elemento da enumeração é associado a um valor numérico que representa seu índice dentro do conjunto.

```
enum DIAS_DA_SEMANA {SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};
                        0      1      2      3      4      5      6
```

Código: Exemplo na linguagem C.

```
if ( diaAtual == DOMINGO ) {
    printf("Folga!");
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### ➤ Tipo Array

- Um array é um agregado homogêneo de elementos de dados cujo elemento individual é identificado por sua posição no agregado em relação ao primeiro.
- Os vetores possuem tamanho fixo e suas posições são contíguas na memória.
- Os arrays podem ser unidimensionais ou multidimensionais.



Programação Estruturada – Rodrigo Adur

## Tipos de dados



### ➤ Tipo Array

```
int valores[] = {10, 20, 30, 100};
int terceiroElemento = valores[2];
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### ➤ Tipo Registro

- Um registro é um agregado possivelmente heterogêneo de elementos cujos elementos individuais são identificados por nomes.
- Os campos dos registro são nomeados e as referências a eles são feitas usando esses identificadores.

```
struct Aluno {
    char* nome;
    int idade;
    int matricula;
};
```

Código: Exemplo na linguagem C.

```
struct Aluno aluno1;
aluno1.nome = "Maria";
aluno1.idade = 29;
aluno1.matricula = 100;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### ➤ Tipo Ponteiro

- O ponteiro surgiu com a necessidade de se alocar memória de acordo com as demandas dinâmicas do programa.
- Uma variável de ponteiro pode ser interpretada de duas maneiras:
  - Como uma referência ao conteúdo da célula de memória à qual a variável está vinculada (endereço).
  - Como uma referência ao valor da célula de memória cujo endereço está sendo referenciado pelo ponteiro.
- Principais operações com ponteiros:
  - Alocação
  - Desalocação
  - Derreferenciamento

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipo Ponteiro

- Uma operação de alocação armazena dinamicamente um espaço de memória acessado via ponteiro e retorna o endereço da célula inicial alocada.

```
int* ptrInteriro = (int) malloc(sizeof(int));
```

Código: Exemplo na linguagem C.

- Uma operação de desalocação devolve a memória alocada de forma que a mesma fique disponível novamente para uso.

```
free(ptrInteriro);
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Tipos de dados



### Tipo Ponteiro

- A operação de derreferenciamento retorna o conteúdo do que é apontado pelo ponteiro.

```
int a = 10;
int *p = &a;
*p = *p + 11;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Conceitos Gerais

- As expressões são o meio fundamental de especificar computações em uma linguagem de programação.
- Expressões são caracterizadas pelo uso de operadores, pelos tipos de operandos e pelo tipo de resultado que produzem.
- Para entender a avaliação de expressões é necessário estar familiarizado com as ordens de avaliação de operadores e de operandos.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Aritméticas

- Nas linguagens de programação, as expressões aritméticas consistem em operadores, operandos, parênteses e chamadas a função.
- Os operadores podem ser unários, significando que tem um único operando, ou binários, significando que têm dois operandos.
- A finalidade de uma expressão aritmética é especificar uma computação aritmética.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Aritméticas

#### Precedência dos operadores

- O valor de uma expressão depende da ordem de avaliação dos operadores na expressão.
- As regras de precedência de operadores definem a ordem em que os operadores de diferentes níveis de precedência são avaliados.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Aritméticas

#### Precedência dos operadores

Operadores	Descrição
()	Parênteses
* / %	Multiplicação, divisão e resto da divisão
+ -	Adição e Subtração binária

+ Maior prioridade

- Menor prioridade

Figura: Alguns operadores da linguagem C.

#### Parênteses

- O uso de parênteses na expressão altera a regra de precedência.

```
int valor = 3 + 5 * 3;  → 18
int valor = (3 + 5) * 3; → 24
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Condicionais

- O operador ternário é usado para formar expressões condicionais.

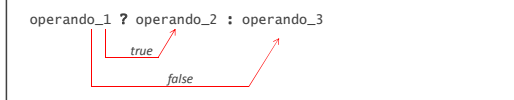


Figura: Lógica do operador ternário.

```
char* statusAluno = media >= 7.0 ? "Aprovado" : "Reprovado";
```

Operando\_1      Operando\_2      Operando\_3

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Relacionais

- Uma expressão relacional tem dois operandos e um operador relacional.
- O resultado de uma expressão relacional é um valor booleano.

Operadores	Descrição
Igual	==
Diferente	!=
Maior que	>
Menor que	<
Maior que ou igual	>=
Menor que ou igual	<=

Figura: Operadores da linguagem C.

```
media >= 7.0
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Binárias

- As expressões binárias são responsáveis por realizar operações sobre conjuntos de bits binários.

Operador	Descrição
&	Conjunção binária
	Disjunção inclusiva
^	Disjunção exclusiva
<<	Deslocamento à esquerda
>>	Deslocamento à direita

Figura: Alguns operadores da linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Binárias

- Abaixo a tabela verdade de algumas operações:

P	Q	P & Q	P   Q	P ^ Q
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Figura: Tabela verdade.

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Binárias

#### Conjunção Binária (AND)

```
int x = 10;
int y = 2;
printf( "x & y: %d", x & y );
// x & y: 2
```

Código: Exemplo na linguagem C.

10 = 1010<sub>(2)</sub>  
2 = 0010<sub>(2)</sub>  
2 = 0010<sub>(2)</sub>

#### Disjunção Inclusiva (OR)

```
int x = 10;
int y = 2;
printf( "x | y: %d", x | y );
// x | y: 10
```

Código: Exemplo na linguagem C.

10 = 1010<sub>(2)</sub>  
2 = 0010<sub>(2)</sub>  
10 = 1010<sub>(2)</sub>

Programação Estruturada – Rodrigo Adur

## Expressões



### Expressões Binárias

#### Disjunção Exclusiva (XOR)

```
int x = 10;
int y = 2;
printf( "x ^ y: %d", x ^ y );
// x ^ y: 8
```

Código: Exemplo na linguagem C.

10 = 1010<sub>(2)</sub>  
2 = 0010<sub>(2)</sub>  
8 = 1000<sub>(2)</sub>

#### Deslocamento à Esquerda

```
int x = 10;
int y = 2;
printf( "x << y: %d", x << y );
// x << y: 40
```

Código: Exemplo na linguagem C.

32 16 8 4 2 1  
1 0 1 0 0 0  
1 0 1 0 0 0  
40 = 10100<sub>(2)</sub>

Programação Estruturada – Rodrigo Adur

## Expressões

- Expressões Binárias
- Deslocamento à Direita
 

<pre>int x = 10; int y = 2; printf( "x &gt;&gt; y: %d", x &gt;&gt; y ); // x &gt;&gt; y: 2</pre>	<pre>32 16 8 4 2 1 1 0 1 0(2) &gt;&gt; 2 ----- 1 0(2) = 2</pre>
--	---

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões

- Avaliação curto circuito
  - Avaliações em curto-circuito são muito usadas para avaliar as expressões booleanas binárias de conjunção ou disjunção.
  - Uma avaliação curto-circuito de uma expressão se caracteriza por determinar seu resultado sem avaliar todos os operandos e operadores.

```
if ( media >= 7 && qtdFaltas <= 6 ) {
    statusAluno = "Aprovado";
}
```

Código: Exemplo na linguagem C.

```
if ( media < 7 || qtdFaltas > 6 ) {
    statusAluno = "Reprovado";
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões

- Instruções de Atribuição
  - A instrução de atribuição oferece o mecanismo por meio do qual o usuário pode mudar dinamicamente as vinculações de valores a variáveis.
  - Atribuição Simples
    - Tipo mais comum de atribuição, no qual o resultado de uma expressão é atribuído a uma variável.

```
int mediaNotas = (n1 + n2 + n3) / 3;
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões

- Instruções de Atribuição
  - Alvos Múltiplos
    - Permitir a atribuição do valor da expressão a mais de uma variável.

```
resultadoParcial = total = 200;
```

Código: Exemplo na linguagem C.
  - Alvos Condicionais
 

```
char* statusAluno = media >= 7.0 ? "Aprovado" : "Reprovado";
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões

- Instruções de Atribuição
- Operadores de Atribuição Compostos
  - Um operador de atribuição composto é um método abreviado de especificar uma forma de atribuição que é utilizada com muita frequência.

Operadores		
+=	-=	*=
/=	%=	=
^=	<=<=	>=>=

Figura: Alguns operadores da linguagem C

<pre>a = a + 10; a += 10;</pre>	<pre>a = a - 10; a -= 10;</pre>	<pre>a = a * 2; a *= 2;</pre>
---------------------------------	---------------------------------	-------------------------------

Código: Exemplos na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões

- Instruções de Atribuição
- Operadores de Atribuição Unários
  - São operadores unários especiais usados para fazer incrementos e decrementos.

Operadores	
++	--

Figura: Operadores da linguagem C

<pre>a = a + 1; a ++;</pre>	<pre>a = a - 1; a --;</pre>
-----------------------------	-----------------------------

Código: Exemplos na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Instruções de Atribuição

#### Operadores de Atribuição Unários

- Esses operadores podem ser usados de forma pré-fixada ou pós-fixada.

```
int valor1 = 3;
int valor2 = ++valor1 + 10;
```

Diagrama de execução:

- [1º] valor1 = 4
- [2º] resolve a expressão
- [3º] valor2 = 14 (atribuição)

Código: Exemplo na linguagem C.

```
int valor1 = 3;
int valor2 = valor1++ + 10;
```

Diagrama de execução:

- [1º] resolve a expressão
- [2º] valor2 = 13 (atribuição)
- [3º] valor1 = 4

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Expressões



### Precedência dos operadores

Operadores	
()	+ Maior prioridade
++ --	
* / %	
+ -	
<< >>	
< <= > >=	- Menor prioridade
== !=	
&&	
?:	
= += -=	

Figura: Operadores em C

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções Sequenciais

- Determina que as instruções são executadas em uma sequência predeterminada.

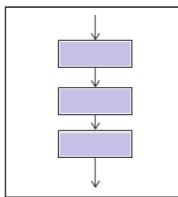


Figura: Sequência.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

- Uma instrução de seleção permite a especificação de caminhos alternativos para o fluxo de controle do programa.

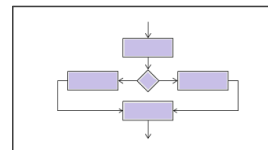


Figura: Seleção.

- A maioria das linguagens de programação fornece três tipos de instruções de seleção: Seleção de Caminho Condicionado, Seleção de Caminho Duplo e Seleção de Caminhos Múltiplos.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

#### Seleção de Caminho Condicionado

- Esse comando permite que um trecho de programa seja executado se determinada condição é satisfeita.

#### Sintaxe:

```
if ( expressão ) {
    instrução;
}
```

Código: Sintaxe na linguagem C.

#### Exemplo

```
if ( salario > mediaSalarial ) {
    salario = salario + gratificacaoFuncao;
    salario = salario + bonus;
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

#### Seleção de Caminho Duplo

- Esse comando permite uma escolha entre dois trechos alternativos de programa.

- Em nenhuma circunstância os dois blocos de comandos serão executados.

#### Sintaxe:

```
if ( expressão ) {
    instrucoes_if;
} else {
    instrucoes_else;
}
```

Código: Sintaxe na linguagem C.

#### Exemplo

```
if ( vendas > mediaVendas ) {
    salario = salario + gratificacaoFuncao;
    salario = salario + bonus;
} else {
    salario = salario + (gratificacaoFuncao * 0.8);
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

#### Aninhando Seletores

- Em muitas linguagens de programação ocorre um problema de ambiguidade sintática na escrita de comandos condicionais duplos aninhados.

```
if ( media < 7 ) {
    printf( "Não Aprovado" );
    if ( media > 3 ) {
        printf( "Recuperação" );
    }
} else {
    printf( "Aprovado" );
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

```
if ( media < 7 )
    printf( "Não Aprovado" );
if ( media > 3 )
    printf( "Recuperação" );
else
    printf( "Aprovado" );
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

#### Seleção de Caminhos Múltiplos

- Esse comando permite uma escolha entre várias alternativas de execução do programa.

#### Sintaxe:

```
switch ( expressão ) {
    case expressão_constante_1 : instruções_case_1;
    case expressão_constante_2 : instruções_case_2;
    ...
    [default : instruções_default]
}
```

Código: Sintaxe na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções de Seleção

#### Seleção de Caminhos Múltiplos

```
switch ( operacao ) {
    case '+':
        printf( "Realiza a soma" );
        break;
    case '-':
        printf( "Realiza a subtração" );
        break;
    case '/':
        printf( "Realiza a divisão" );
        break;
    case '*':
    case 'x':
        printf( "Realiza a multiplicação" );
        break;
    default:
        printf( "Operação Inesperada" );
}
```

Código: Exemplo na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções Iterativas

- Uma instrução iterativa faz com que uma instrução ou uma coleção de instruções seja executada zero, uma ou mais vezes.

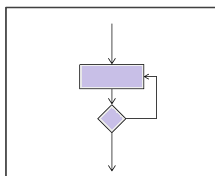


Figura: Iteração.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



### Instruções Iterativas

#### Laços controlados por contador

- Esse comando é geralmente usado quando o número de iterações é determinado previamente.

#### Sintaxe:

```
for ( expressão_1; expressão_2; expressão_3 )
```

Inicialização
Controle de laço
Termínio de cada iteração

Código: Sintaxe na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados por contador

- A primeira expressão serve para a inicialização, sendo avaliada somente uma vez, quando se inicia a execução da instrução for.
    - A segunda expressão é o controle do laço, avaliada antes de cada iteração.
    - A terceira expressão é executada depois de cada iteração do corpo do laço.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados por contador

- Cada uma das expressões pode conter instruções múltiplas que deverão estar separadas por vírgulas.

```
for ( i = 1, j = 10; i < j; i++, j-- ){
    printf( "i: %d | j: %d \n", i, j );
}
```

Código: Exemplo na linguagem C – exibe os valores de i e j enquanto i &lt; j.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados logicamente

- Esses tipos de laços são usados quando o número de iterações não é definido previamente.
    - Algumas linguagens incluem tanto laços pré-testes como pós-testes controlados logicamente.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados logicamente (pré-teste)

- Na versão pré-teste, a instrução é executada contanto que a expressão seja avaliada como verdadeira .

- Sintaxe:

```
while ( expressão ) {
    instruções;
}
```

Código: Sintaxe na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados logicamente (pré-teste)

```
int i = 1;
while ( i <= 10 ) {
    int resto = i % 2;
    if ( resto == 0 ) {
        printf( "%d \n", i );
    }
    i++;
}
```

Código: Exemplo na linguagem C – exibe os números pares compreendidos entre 1 e 10 (inclusive).

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas

- Laços controlados logicamente (pós-teste)

- Na versão pós-teste, o corpo do laço é executado até que a expressão seja avaliada como falsa.

- Sintaxe:

```
do {
    instruções;
} while ( expressão );
```

Código: Sintaxe na linguagem C.

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas
  - Laços controlados logicamente (pós-teste)

```
int i = 1;
do {
    int resto = i % 2;
    if ( resto == 0 ) {
        printf( "%d \n", i );
    }
    i++;
} while ( i <= 10 );
```

Código: Exemplo na linguagem C – exibe os números pares compreendidos entre 1 e 10 (inclusive).

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas
  - Mecanismos de controle de laços
    - O mecanismo de controle de laço mais comum é o break.
    - O comando break possibilita colocar o ponto de parada do comando iterativo em qualquer local do seu corpo.

```
int i = 1;
for ( ; ; ) {
    if ( i > 10 ) {
        break;
    }

    int resto = i % 2;
    if ( resto == 0 ) {
        printf( "%d \n", i );
    }
    i++;
}
```

Código: Exemplo na linguagem C – exibe os números pares compreendidos entre 1 e 10 (inclusive).

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Instruções Iterativas
  - Mecanismos de controle de laços
    - O mecanismo de controle de laço continue possibilita pular instruções da iteração corrente sem finalizar o loop.
    - O fluxo de execução continua no início da próxima iteração do loop.

```
int i;
for ( i = 1 ; i <= 10; i++ ) {
    int resto = i % 2;
    if ( resto == 1 ) {
        continue;
    }
    printf( "%d \n", i );
}
```

Código: Exemplo na linguagem C – exibe os números pares compreendidos entre 1 e 10 (inclusive).

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Desvio Incondicional
  - Uma instrução de desvio incondicional transfere o controle de execução para um lugar especificado no programa.
  - O desvio incondicional é a instrução mais poderosa para controlar o fluxo de execução de instruções de um programa.

```
int i = 1;
int resto;

rotuloInicial:
resto = i % 2;
if ( resto == 0 ) {
    printf( "%d \n", i );
}
if ( i++ < 10 ) {
    goto rotuloInicial;
}
```

Código: Exemplo na linguagem C – exibe os números pares compreendidos entre 1 e 10 (inclusive).

Programação Estruturada – Rodrigo Adur

## Estruturas de Controle



- Desvio Incondicional
  - O goto tem um poder formidável e uma grande flexibilidade, mas esse próprio poder torna seu uso perigoso.
  - O uso do goto pode resultar em:
    - Baixa legibilidade;
    - Baixa manutenibilidade.

Programação Estruturada – Rodrigo Adur

## Questão 04



## [2010 – CESGRANRIO - Petrobras - Engenharia de Software]

Abaixo são exibidas expressões na linguagem Java, nas quais *a* e *b* são variáveis do tipo *boolean*. Qual, dentre as expressões que, ao ser avaliada, resulta em um valor diferente das demais?

- (A) (!a | b)^true
- (B) a ^ b
- (C) (a | b) & ! (a & b)
- (D) (!a | !b) & (a | b)
- (E) (a | (b^false)) & ( (a^true) | !b)

Programação Estruturada – Rodrigo Adur



## Questão 04



- (A)  $(!a \mid b)^{\wedge}true$   
 (B)  $a \wedge b$   
 (C)  $(a \mid b) \& !(a \& b)$   
 (D)  $(!a \mid !b) \& (a \mid b)$   
 (E)  $(a \mid (b^{\wedge}false)) \& ((a^{\wedge}true) \mid !b)$

a	b	$a \wedge b$	$!a$	$(!a \mid b)$	$(!a \mid b)^{\wedge}TRUE$	$a \mid b$	$a \& b$	$!(a \& b)$	$(a \mid b) \& !(a \& b)$
V	V	F	F	V	F	V	V	F	F
V	F	V	F	F	V	V	F	V	V
F	V	V	V	V	F	V	F	V	V
F	F	F	V	V	F	F	V	V	F

(B)

(A)

(C)

Programação Estruturada – Rodrigo Adur

## Questão 04



## [2010 – CESGRANRIO - Petrobras - Engenharia de Software]

Abaixo são exibidas expressões na linguagem Java, nas quais  $a$  e  $b$  são variáveis do tipo *boolean*. Qual, dentre as expressões que, ao ser avaliada, resulta em um valor diferente das demais?

- (A)  $(!a \mid b)^{\wedge}true$   
 (B)  $a \wedge b$   
 (C)  $(a \mid b) \& !(a \& b)$   
 (D)  $(!a \mid !b) \& (a \mid b)$   
 (E)  $(a \mid (b^{\wedge}false)) \& ((a^{\wedge}true) \mid !b)$

Programação Estruturada – Rodrigo Adur

## Questão 05



## [2013 – FCC - TRT - 9ª REGIÃO - Tecnologia da Informação]

```

início
real: n1, n2;
imprima("Digite a primeira nota: ");
leia(n1);
imprima("Digite a segunda nota: ");
leia(n2);
se [ ]
então
  media ← (n1+n2)/2;
  imprima ("A media das notas é ", media);
senão
  imprima("Alguma nota fornecida é inválida.");
fim se;
fim.

```

Considerando que uma nota válida deve possuir valores entre 0 e 10 (inclusive), a lacuna que corresponde à condição do comando *SE* é corretamente preenchida por:

- (A)  $n1 \geq 0 \text{ OU } n1 \leq 10 \text{ OU } n2 \geq 0 \text{ OU } n2 \leq 10$   
 (B)  $(n1 \geq 0 \text{ E } n1 \leq 10) \text{ OU } (n2 \geq 0 \text{ E } n2 \leq 10)$   
 (C)  $(n1 \geq 0 \text{ OU } n1 \leq 10) \text{ E } (n2 \geq 0 \text{ OU } n2 \leq 10)$   
 (D)  $n1 \geq 0 \text{ E } n1 \leq 10 \text{ E } n2 \geq 0 \text{ E } n2 \leq 10$   
 (E)  $n1 \geq 0 \text{ E } n1 \leq 10 \text{ E } n2 \geq 0 \text{ E } n2 \leq 10$

Programação Estruturada – Rodrigo Adur

## Questão 05



## [2013 – FCC - TRT - 9ª REGIÃO - Tecnologia da Informação]

```

início
real: n1, n2;
imprima("Digite a primeira nota: ");
leia(n1);
imprima("Digite a segunda nota: ");
leia(n2);
se [ ]
então
  media ← (n1+n2)/2;
  imprima ("A media das notas é ", media);
senão
  imprima("Alguma nota fornecida é inválida.");
fim se;
fim.

```

Considerando que uma nota válida deve possuir valores entre 0 e 10 (inclusive), a lacuna que corresponde à condição do comando *SE* é corretamente preenchida por:

- (A)  $n1 \geq 0 \text{ OU } n1 \leq 10 \text{ OU } n2 \geq 0 \text{ OU } n2 \leq 10$   
 (B)  $(n1 \geq 0 \text{ E } n1 \leq 10) \text{ OU } (n2 \geq 0 \text{ E } n2 \leq 10)$   
 (C)  $(n1 \geq 0 \text{ OU } n1 \leq 10) \text{ E } (n2 \geq 0 \text{ OU } n2 \leq 10)$   
 → (D)  $n1 \geq 0 \text{ E } n1 \leq 10 \text{ E } n2 \geq 0 \text{ E } n2 \leq 10$   
 (E)  $n1 > 0 \text{ E } n1 < 10 \text{ E } n2 > 0 \text{ E } n2 < 10$

Programação Estruturada – Rodrigo Adur

## Questão 06



## [2013 – FCC - TRT - 9ª REGIÃO - Tecnologia da Informação]

Considere o algoritmo em pseudolinguagem:

```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
  se (sexo = 'm' E sexo = 'M')
  então imprima ("Você é do sexo masculino. ");
  senão imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim se;
fim.

```

Programação Estruturada – Rodrigo Adur

## Questão 06



## [2013 – FCC - TRT - 9ª REGIÃO - Tecnologia da Informação]

Sobre o algoritmo acima é correto afirmar que:

- (A) a lógica do algoritmo está comprometida pela falta de um comando de repetição.  
 (B) em vez de utilizar comandos de decisão se aninhados, deveria ter sido usado um único comando de seleção múltipla, por isso a lógica ficou comprometida.  
 (C) se for digitada uma letra maiúscula 'F' ou minúscula 'f' será impresso Você é do sexo feminino.  
 (D) somente se a letra digitada para o sexo for diferente de 'F', 'f', 'M', 'm' é que a frase Você digitou um valor de sexo inválido será impressa.  
 (E) não importa o valor digitado no sexo, pois sempre será impresso Você digitou um valor de sexo inválido.

Programação Estruturada – Rodrigo Adur

## Questão 06



```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
se (sexo = 'm' E sexo = 'M')
então imprima ("Você é do sexo masculino. ");
senão
imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim.

```

(A) a lógica do algoritmo está comprometida pela falta de um comando de repetição.

Programação Estruturada – Rodrigo Adur

## Questão 06



```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
se (sexo = 'm' E sexo = 'M')
então imprima ("Você é do sexo masculino. ");
senão
imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim.

```

(B) em vez de utilizar comandos de decisão *se* aninhados, deveria ter sido usado um único comando de seleção múltipla, por isso a lógica ficou comprometida.

Programação Estruturada – Rodrigo Adur

## Questão 06



```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
se (sexo = 'm' E sexo = 'M')
então imprima ("Você é do sexo masculino. ");
senão
imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim.

```

(C) se for digitada uma letra maiúscula 'F' ou minúscula 'f' será impresso Você é do sexo feminino.

Programação Estruturada – Rodrigo Adur

## Questão 06



```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
se (sexo = 'm' E sexo = 'M')
então imprima ("Você é do sexo masculino. ");
senão
imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim.

```

(D) somente se a letra digitada para o sexo for diferente de 'F', 'f', 'M', 'm' é que a frase Você digitou um valor de sexo inválido será impressa.

Programação Estruturada – Rodrigo Adur

## Questão 06



```

início
caracter: nome, sexo;

imprima ("Qual é o seu nome? ");
leia (nome);
imprima ("Qual é o seu sexo? (f/m) ");
leia (sexo);

se (sexo = 'f' E sexo = 'F')
então imprima ("Você é do sexo feminino. ");
senão
se (sexo = 'm' E sexo = 'M')
então imprima ("Você é do sexo masculino. ");
senão
imprima ("Você digitou um valor de sexo invalido ");
fim se;
fim.

```

(E) não importa o valor digitado no sexo, pois sempre será impresso Você digitou um valor de sexo inválido.

Programação Estruturada – Rodrigo Adur

## Questão 06



[2013 – FCC – TRT - 9ª REGIÃO - Tecnologia da Informação]

Sobre o algoritmo acima é correto afirmar que:

- (A) a lógica do algoritmo está comprometida pela falta de um comando de repetição.
- (B) em vez de utilizar comandos de decisão *se* aninhados, deveria ter sido usado um único comando de seleção múltipla, por isso a lógica ficou comprometida.
- (C) se for digitada uma letra maiúscula 'F' ou minúscula 'f' será impresso Você é do sexo feminino.
- (D) somente se a letra digitada para o sexo for diferente de 'F', 'f', 'M', 'm' é que a frase Você digitou um valor de sexo inválido será impressa.
- ➔ (E) não importa o valor digitado no sexo, pois sempre será impresso Você digitou um valor de sexo inválido.

Programação Estruturada – Rodrigo Adur

**Questão 07**

[2009 – ESAF - ANA - Tecnologia da Informação (Desenvolvimento)]

Na programação estruturada, são necessários apenas três blocos de formas de controle para implementar algoritmos. São eles:

- (A) seleção, repetição e aninhamento.
- (B) empilhamento, aninhamento e operação.
- (C) sequência, aninhamento e seleção.
- (D) sequência, seleção e repetição.
- (E) função, operação e programa.

Programação Estruturada – Rodrigo Adur

**Questão 07**

[2009 – ESAF - ANA - Tecnologia da Informação (Desenvolvimento)]

Na programação estruturada, são necessários apenas três blocos de formas de controle para implementar algoritmos. São eles:

- (A) seleção, repetição e aninhamento.
- (B) empilhamento, aninhamento e operação.
- (C) sequência, aninhamento e seleção.
- ➔ (D) sequência, seleção e repetição.
- (E) função, operação e programa.

Programação Estruturada – Rodrigo Adur

**Questão 08**

[2011 – CESPE - TJ-ES - Técnico de Informática]

Julgue os itens de 29 a 40, relativos a fundamentos de computação e linguagens de programação e desenvolvimento para a Web.

Uma estrutura de repetição possibilita executar um bloco de comando, repetidas vezes, até que seja encontrada uma dada condição que conclua a repetição.

Certo      Errado

Programação Estruturada – Rodrigo Adur

**Questão 08**

[2011 – CESPE - TJ-ES - Técnico de Informática]

Julgue os itens de 29 a 40, relativos a fundamentos de computação e linguagens de programação e desenvolvimento para a Web.

Uma estrutura de repetição possibilita executar um bloco de comando, repetidas vezes, até que seja encontrada uma dada condição que conclua a repetição.

➔ Certo      Errado

Programação Estruturada – Rodrigo Adur

**Questão 09**

[2011 – CESPE – EBC - Analista]

A respeito de estruturas de controle de fluxo em algoritmos, julgue os próximos itens.

Nas estruturas de repetição, que são utilizadas quando se deseja repetir certo trecho de instruções, o número de repetições deve ser conhecido ou determinado previamente e precisa ser finito.

Certo      Errado

Programação Estruturada – Rodrigo Adur

**Questão 09**

[2011 – CESPE – EBC - Analista]

A respeito de estruturas de controle de fluxo em algoritmos, julgue os próximos itens.

Nas estruturas de repetição, que são utilizadas quando se deseja repetir certo trecho de instruções, o número de repetições deve ser conhecido ou determinado previamente e precisa ser finito.

Certo      ➔ Errado

Programação Estruturada – Rodrigo Adur

## Questão 10



[2011 – CESGRANRIO – EPE – Tecnologia da Informação]

Observe o fragmento de código abaixo:

```
x=3;
y=4;
z=5;

if ((x-1) > 2)
    y=y+1;
else
    y=y-1;
    z=x+y;
for (i=1; i<9; i++)
    y=y+1;
    z=z+y;
```

Programação Estruturada – Rodrigo Adur

## Questão 10



[2011 – CESGRANRIO – EPE – Tecnologia da Informação]

Ao final da execução desse código, qual o valor de z?

- (A) 12
- (B) 15
- (C) 16
- (D) 17
- (E) 20

Programação Estruturada – Rodrigo Adur

## Questão 10



```
x=3;
y=4; y=3 y = 11
z=5; z=6 z = 17

if ((x-1) > 2)
    y=y+1;
else
    y=y-1;
    z=x+y;
for (i=1; i<9; i++)
    y=y+1;
    z=z+y;
```

Programação Estruturada – Rodrigo Adur

## Questão 10



[2011 – CESGRANRIO – EPE – Tecnologia da Informação]

Ao final da execução desse código, qual o valor de z?

- (A) 12
- (B) 15
- (C) 16
- (D) 17
- (E) 20

Programação Estruturada – Rodrigo Adur

## Gabarito



- |        |             |
|--------|-------------|
| 04 – A | 08 – Certo  |
| 05 – D | 09 – Errado |
| 06 – E | 10 – D      |
| 07 – D |             |

Programação Estruturada – Rodrigo Adur

**Modularização**

## Modularização



### Conceitos Gerais

- Dividir para conquistar é uma técnica comumente usada para resolver problemas complexos.
- A modularização é a forma que implementamos a técnica dividir para conquistar nas linguagens de programação.
- As técnicas de modularização foram desenvolvidas com o objetivo de dar apoio ao desenvolvimento de sistemas grandes.

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

- A abstração consiste em focar os aspectos relevantes de um problema e ignorar os demais aspectos.
- As linguagens de programação fornecem alguns mecanismos de abstrações que podem ser classificados como:
  - Abstração de Processos – são abstrações sobre o fluxo de controle do programa.
  - Abstração de Dados – são abstrações sobre as estruturas de dados do programa.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

- Os subprogramas são o mecanismo mais comumente usado para representar abstrações de processos.
- Eles permitem segmentar um programa em vários blocos logicamente relacionados.
- Um subprograma é responsável por realizar uma determinada funcionalidade que deve possuir um propósito único e bem definido.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

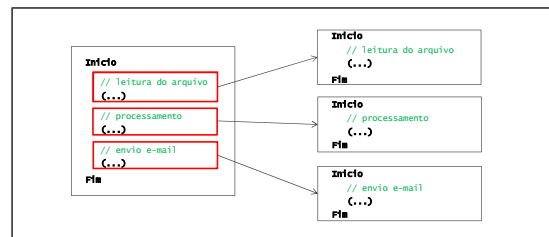


Figura: Representação de subprogramas.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

- Como consequência, podemos afirmar que a segmentação implica em:
  - Aumento da legibilidade dos programas.
  - Facilidade de depuração.
  - Reuso do código.
- O uso do subprograma torna-se oportuno quando alguma parte do programa desempenha uma tarefa específica ou quando algumas linhas de código aparecem repetidas em trechos distintos do programa.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

- Uma chamada a um subprograma é uma solicitação explícita para executar o programa. Cada subprograma tem um único ponto de entrada.
- A unidade chamadora é suspensa durante a execução do programa chamado.
- O controle sempre retorna ao chamador quando a execução do subprograma é finalizada.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

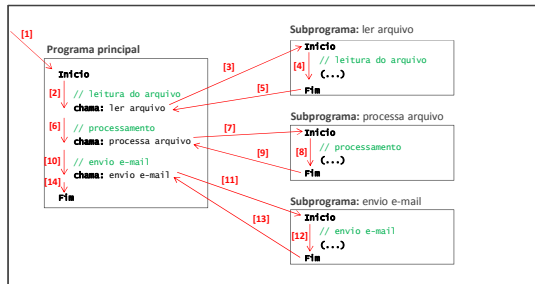


Figura: Fluxo de execução dos subprogramas.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

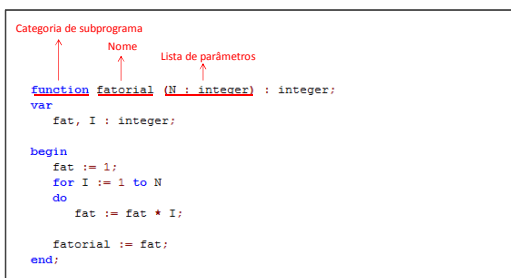
- Existem duas categorias de subprogramas: procedimentos e funções.
- Na primeira linha de definição de um subprograma temos o seu cabeçalho, que conterá as seguintes informações:
  - Definir a categoria do subprograma
  - Nomear o subprograma
  - Especificar uma lista de parâmetros

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas



Código: Exemplo na linguagem Pascal – subprograma da categoria função.

Programação Estruturada – Rodrigo Adur

## Modularização



### Subprogramas

- Parâmetros**
  - Os parâmetros facilitam o processo de aplicação de dados diferenciados entre chamadas distintas de um subprograma.
  - Os dados passados pelos parâmetros são acessados por nomes locais para o subprograma.
  - Os parâmetros:
    - No cabeçalho dos subprogramas são chamados de parâmetros formais.
    - Passados na chamada dos subprogramas são chamados de parâmetros reais.

Programação Estruturada – Rodrigo Adur

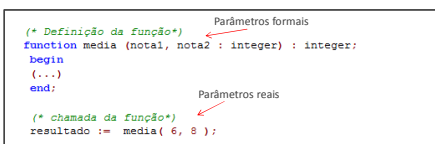
## Modularização



### Subprogramas

#### Correspondência entre parâmetros reais e parâmetros formais:

- Posicional** – a sequência na qual os parâmetros são escritos determina a correspondência entre os parâmetros formais e reais.



Código: Exemplo na linguagem Pascal – correspondência posicional.

Programação Estruturada – Rodrigo Adur

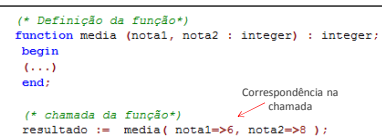
## Modularização



### Subprogramas

#### Correspondência entre parâmetros reais e parâmetros formais:

- Por palavra-chave** – envolvem a listagem explícita dos parâmetros reais e seus correspondentes formais.



Código: Exemplo na linguagem ADA – correspondência posicional.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Correspondência entre parâmetros reais e parâmetros formais:

- Em ambos os casos, os tipos dos parâmetros reais devem ser compatíveis com os dos parâmetros formais correspondentes.
- Algumas linguagens permitem a criação de subprogramas cujo número de parâmetros reais pode variar de uma chamada para outra.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Procedimentos

- Os procedimentos são coleções de instruções que abstraem um comando a ser executado.
- Quando um procedimento é chamado, ele pode produzir resultados na unidade de programa chamadora por dois métodos:
  - Parâmetros formais que permitam a transferência de dados para o chamador.
  - Atualização de variáveis, que não são parâmetros, que são visíveis tanto no procedimento quando na unidade chamadora.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Procedimentos

```

Procedure ordena(var array:IntArray; size:integer);
var
  i,j,temp: integer;

begin
  for i := size to 1 do
    for j := 1 to i-1 do
      if array[j] > array[j+1] then
        begin
          temp = array[j];
          array[j] = array[j+1];
          array[j+1] = temp;
        end;
      end;
    end;
  end;

```

Código: Exemplo na linguagem Pascal – ordenação de um conjunto de números.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Funções

- Uma função abstrai uma expressão a ser avaliada.
- Elas retornam um valor que é associado ao seu nome, dessa forma, é necessário declarar que tipo de dado será retornado.
- Em essência, as funções não deveriam causar nenhum efeito colateral, ou seja, ela não deveria modificar nenhum parâmetro e nem qualquer variável declarada fora dela.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Funções

```

function fatorial (N : integer) : integer;
var
  fat, I : integer;

begin
  fat := 1;
  for I := 1 to N do
    fat := fat * I;

  fatorial := fat;
end;

```

Código: Exemplo na linguagem Pascal – cálculo do fatorial de um número.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas

- Os clientes desses subprogramas se preocupam apenas com os dados que devem ser enviados e o efeitos da computação (no caso do procedimento) ou o valor retornado (no caso das funções).
- O implementador se preocupa com o algoritmo do subprograma, considerando os valores recebidos por parâmetro.
- Um subprograma pode utilizar constantes ou variáveis do módulo principal ou definir suas próprias constantes ou variáveis.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Os métodos de passagem de parâmetros são as maneiras pelas quais se transmitem parâmetros para os subprogramas.
    - Métodos de passagem de parâmetros:
      - Passagem por valor
      - Passagem por referência
    - Tanto as funções quanto os procedimentos permitem passagem de parâmetro por valor e por referência.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - **Passagem por valor**
      - Esse mecanismo envolve a criação de uma cópia do parâmetro real no ambiente local do subprograma.
      - O valor do parâmetro real é usado para inicializar o parâmetro formal correspondente.
      - Alterações feitas nos parâmetros formais não refletem nos parâmetros reais.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por valor

```
void exibeSucessor(int n){
    n = n + 1;
    printf( "%d\t", n );
}

→ int numero = 1;
  exibeSucessor( numero );
  printf( "Depois: %d\t", numero );
```

Código: Exemplo na linguagem C – passagem por valor.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por valor

```
void exibeSucessor(int n){
    n = n + 1;
    printf( "%d\t", n );
}

→ int numero = 1;
  exibeSucessor( numero );
  printf( "Depois: %d\t", numero );
```



Código: Exemplo na linguagem C – passagem por valor.

Programação Estruturada – Rodrigo Adur

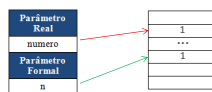
## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por valor

```
void exibeSucessor(int n){
    n = n + 1;
    printf( "%d\t", n );
}

→ int numero = 1;
  exibeSucessor( numero );
  printf( "Depois: %d\t", numero );
```



Código: Exemplo na linguagem C – passagem por valor.

Programação Estruturada – Rodrigo Adur

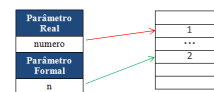
## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por valor

```
void exibeSucessor(int n){
    n = n + 1;
    printf( "%d\t", n );
}

→ int numero = 1;
  exibeSucessor( numero );
  printf( "Depois: %d\t", numero );
```



Código: Exemplo na linguagem C – passagem por valor.

Programação Estruturada – Rodrigo Adur



## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por valor

```
void exibeSucessor(int n){
    n = n + 1;
    printf( "%d\t", n );
}

int numero = 1;
exibeSucessor( numero );
→ printf( "Depois: %d\t", numero );
```

Código: Exemplo na linguagem C – passagem por valor.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem referência
      - Esse mecanismo envia para o ambiente local do subprograma uma referência para os parâmetros reais.
      - O parâmetro formal passa a referenciar as mesmas células de memória dos parâmetros reais.
      - Alterações feitas nos parâmetros formais refletem imediatamente nos parâmetros reais.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem referência
      - A vantagem da passagem por referência é que o próprio processo de passagem é eficiente, tanto em termos de tempo quanto de espaço.
      - O acesso aos parâmetros formais provavelmente serão mais lentos porque mais de um nível de endereçamento indireto é necessário quando valores de dados são transmitidos.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por referência

```
void exibeSucessor(int* n){
    *n = *n + 1;
    printf( "%d\t", *n );
}

→ int numero = 1;
exibeSucessor( &numero );
printf( "Depois: %d\t", numero );
```

Código: Exemplo na linguagem C – passagem por referência.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por referência

```
void exibeSucessor(int* n){
    *n = *n + 1;
    printf( "%d\t", *n );
}

→ int numero = 1;
exibeSucessor( &numero );
printf( "Depois: %d\t", numero );
```

Código: Exemplo na linguagem C – passagem por referência.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por referência

```
void exibeSucessor(int* n){
    *n = *n + 1;
    printf( "%d\t", *n );
}

→ int numero = 1;
exibeSucessor( &numero );
printf( "Depois: %d\t", numero );
```

Código: Exemplo na linguagem C – passagem por referência.

Programação Estruturada – Rodrigo Adur

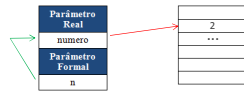
## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por referência

```
void exibeSucessor(int* n){
    *n = *n + 1;
    printf( "%d\t", *n );
}

int numero = 1;
exibeSucessor( &numero );
printf( "Depois: %d\t", numero );
```



Código: Exemplo na linguagem C – passagem por referência.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros
    - Passagem por referência

```
void exibeSucessor(int* n){
    *n = *n + 1;
    printf( "%d\t", *n );
}

int numero = 1;
exibeSucessor( &numero );
printf( "Depois: %d\t", numero );
```



Código: Exemplo na linguagem C – passagem por referência.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros das principais linguagens
    - Pascal
      - Por padrão, adota o método de passagem por valor.
      - Para especificar o método de passagem por referência, basta prefixar os parâmetros formais da palavra reservada var.
    - Linguagem C
      - Por padrão, adota o método de passagem por valor.
      - Obtém-se a semântica de passagem por referência usando-se ponteiros como parâmetros.

Programação Estruturada – Rodrigo Adur

## Modularização



- Subprogramas
  - Métodos de passagem de parâmetros das principais linguagens
    - Linguagem C++
      - Por padrão, adota o método de passagem por valor.
      - Para especificar o método de passagem por referência, basta prefixar os parâmetros formais de &.
    - Linguagem java
      - Somente adota o método de passagem por valor.

Programação Estruturada – Rodrigo Adur

## Modularização



- Recursividade
  - Uma função recursiva é aquela que possui uma ou mais chamadas para si mesma.
  - Toda função recursiva deve possuir um ou mais casos recursivos e pelo menos um caso base.
  - Durante o processamento de uma função recursiva, os dados vão para uma área da memória chamada stack memory.

Programação Estruturada – Rodrigo Adur

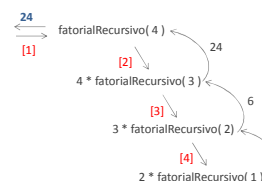
## Modularização



- Recursividade

```
int fatorialRecursivo( int num ) {
    /* caso base */
    if ( num == 0 || num == 1 ) {
        return 1;
    } else {
        /* caso recursivo */
        return num * fatorialRecursivo( num - 1 );
    }
}
```

Código: Exemplo na linguagem C – fatorial com recursividade.



Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Recursividade**

```
int fatorialIterativo( int num ) {
    int fatorial = 1;
    while (num > 0) {
        fatorial = fatorial * num;
        num--;
    }
    return fatorial;
}
```

*Código: Exemplo na linguagem C – fatorial com recursividade.*

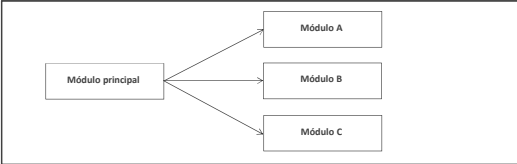
Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Conceitos Gerais**

- As técnicas de modularização implicam a divisão de um sistema em módulos.
- Um módulo bem projetado tem um único propósito e uma boa interface com os outros módulos.
- Um módulo inclui grupos de subprogramas e dados relacionados.



*Figura: Exemplo de Modularização*

Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Conceitos Gerais**

➤ Benefícios da modularização:

- Maior facilidade na escrita e depuração de módulos
- Reuso dos módulos
- Minimização do impacto das alterações
- Maior facilidade de isolamento de erros

Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Conceitos Gerais**

➤ **Coesão**

- Coesão é a medida da intensidade da associação funcional de um módulo.
- Um módulo altamente coeso realiza uma única tarefa, requerendo pouca ou nenhuma interação com procedimentos sendo realizados em outras partes de um programa.
- Quanto maior o nível da coesão, maior a qualidade do módulo.

Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Conceitos Gerais**

➤ **Coesão**

➤ Existem diversos níveis de coesão:

Níveis de Coesão
Funcional
Sequencial
Comunicacional
Procedural
Temporal
Lógica
Coincidental

➤ **Maior nível de coesão** (funcional)

➤ **Menor nível de coesão** (coincidental)

*Tabela: Níveis de Coesão*

Programação Estruturada – Rodrigo Adur

## Modularização

ITnerante

➤ **Conceitos Gerais**

➤ **Coesão**

- **Funcional** – todas as atividades internas estão relacionadas entre si e objetivam cumprir o objetivo do módulo. Maior nível de coesão!
- **Sequencial** – as atividades internas possuem uma relação onde a saída de uma é a entrada de outra.
- **Comunicacional** – as atividades são relacionadas por utilizarem as mesmas informações, ou seja, a mesma entrada ou a mesma saída.

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Coesão

- **Procedural** – as atividades internas estão juntas pois, em conjunto e executadas em determinada ordem, constituem um procedimento.
- **Temporal** – as atividades internas estão juntas pois devem ser executadas no mesmo tempo.

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Coesão

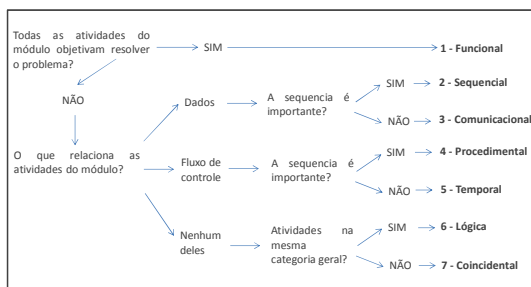
- **Lógica** – as atividades internas pertencem ao mesmo grupo lógico, onde a atividade a ser executada é selecionada pelo usuário do módulo.
- **Coincidental** – funções internas não possuem nenhum relacionamento (não pertencem ao mesmo grupo lógico). Menor nível de coesão!

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais



Esquema: Níveis de Coesão

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Acoplamento

- Acoplamento indica o grau de interdependência entre módulos de um programa.
- Módulos com alto acoplamento apresentam uma grande dependência entre si. Dessa forma, um módulo fica mais vulnerável às mudanças nos outros módulos.
- Quanto menor o nível do acoplamento, maior a qualidade dos módulos.

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Acoplamento

- Existem diversos níveis de acoplamento:

Níveis de Acoplamento	
De Dados	- Menor nível de acoplamento
De Imagem	
De Controle	
Comum	
De Conteúdo	+ Maior nível de acoplamento

Tabela: Níveis de Acoplamento

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Acoplamento

- **De Dados** – os módulos se comunicam apenas por meio de parâmetros trocados. Menor nível de acoplamento!
- **De Imagem** – os módulos se comunicam referenciando uma estrutura de dados em comum, passadas por parâmetros, onde ambos podem ler e gravar informações.
- **De Controle** – os módulos se comunicam de tal forma que parâmetros são passados para controlar o funcionamento interno do módulo chamado.

Programação Estruturada – Rodrigo Adur

## Modularização



### Conceitos Gerais

#### Acoplamento

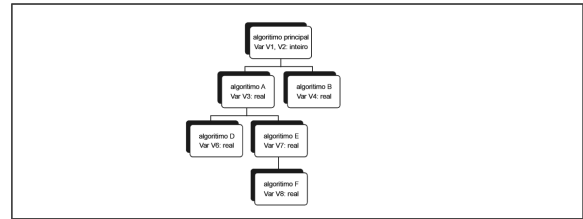
- **Comum** – os módulos se comunicam referenciando uma mesma área de dados global (não são passados por parâmetros).
- **De Conteúdo** – ocorre quando um módulo faz referência ou desvia a sequência de instruções para o interior de outro módulo.

Programação Estruturada – Rodrigo Adur

## Questão 11



[2011 – CESPE– EBC - Analista - Engenharia de Software]



Considerando a figura acima, que ilustra um diagrama representando a hierarquia para a resolução de um problema em módulos, julgue os itens a seguir.

As variáveis V6 e V8 podem ser utilizadas pelos respectivos algoritmos em que foram declaradas e pelo algoritmo A.

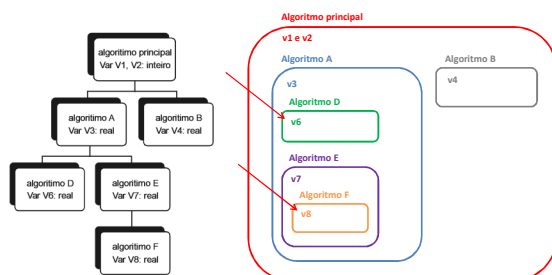
Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 11



[2011 – CESPE– EBC - Analista - Engenharia de Software]



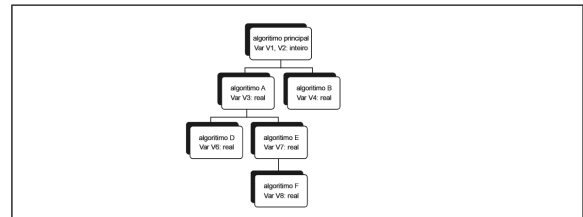
As variáveis V6 e V8 podem ser utilizadas pelos respectivos algoritmos em que foram declaradas e pelo algoritmo A.

Programação Estruturada – Rodrigo Adur

## Questão 11



[2011 – CESPE– EBC - Analista - Engenharia de Software]



Considerando a figura acima, que ilustra um diagrama representando a hierarquia para a resolução de um problema em módulos, julgue os itens a seguir.

As variáveis V6 e V8 podem ser utilizadas pelos respectivos algoritmos em que foram declaradas e pelo algoritmo A.

Certo      ➡ Errado

Programação Estruturada – Rodrigo Adur

## Questão 12



[2009 – CESPE– ANAC - Analista - Tecnologia da Informação]

Julgue os itens que se seguem, com relação a conceitos de construção de algoritmos.

Na passagem de parâmetro por valor, o parâmetro formal tem seu valor inicializado pelo valor do parâmetro real. Por esse motivo, o parâmetro real nunca é alterado. O seu valor se mantém inalterado depois que o subprograma termina a execução.

Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 12



[2009 – CESPE– ANAC - Analista - Tecnologia da Informação]

Julgue os itens que se seguem, com relação a conceitos de construção de algoritmos.

Na passagem de parâmetro por valor, o parâmetro formal tem seu valor inicializado pelo valor do parâmetro real. Por esse motivo, o parâmetro real nunca é alterado. O seu valor se mantém inalterado depois que o subprograma termina a execução.

➡ Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 13



[2010 – CESPE– Banco da Amazônia - Tecnologia da Informação]

Julgue os itens seguintes, relativos à lógica de programação e construção de algoritmos.

Na definição de uma função, a passagem de parâmetros por referência possibilita que o valor de uma variável passado como argumento seja alterado na função, e sua alteração mantenha-se mesmo após a execução da função.

Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 13



[2010 – CESPE– Banco da Amazônia - Tecnologia da Informação]

Julgue os itens seguintes, relativos à lógica de programação e construção de algoritmos.

Na definição de uma função, a passagem de parâmetros por referência possibilita que o valor de uma variável passado como argumento seja alterado na função, e sua alteração mantenha-se mesmo após a execução da função.

→ Certo      Errado

Programação Estruturada – Rodrigo Adur

## Questão 14



[2012 – CONSULPLAN – TSE - Programador de computador]

Observe o pseudocódigo referente a um programa de computador, em que ocorre passagens de parâmetros por valor de BB para MM e por referência de N1 para NP.

```

programa TSE;
variáveis
  N1, N2 : numérica;
  BB : lógica;
procedimento KEPLER(MM:lógica;NP:numérica);
início
  atribuir 38 a NP;
  se (NP par) então atribuir (NÃO MM) a MM;
fim_procedimento_KEPLER;
início
  atribuir FALSO a BB;
  atribuir 26 a N1; atribuir 17 a N2;
  KEPLER(BB,N1);
  se (NÃO BB) então atribuir (N1 / 2) a N2
  senão atribuir (N1 / 5) a N2;
  Escrever(N1,N2,BB);
fim_programa.
  
```

Programação Estruturada – Rodrigo Adur

## Questão 14



[2012 – CONSULPLAN – TSE - Programador de computador]

Ao final da execução, as variáveis N1, N2 e BB terão, respectivamente, os seguintes valores :

- A) 26, 13 e FALSO.
- B) 38, 19 e FALSO.
- C) 38, 19 e VERDADEIRO.
- D) 26, 13 e VERDADEIRO.

Programação Estruturada – Rodrigo Adur

## Questão 14



- Passagens de parâmetros por valor de BB para MM
- Passagem de parâmetros por referência de N1 para NP.

```

programa TSE;
variáveis
  N1, N2 : numérica;
  BB : lógica;
procedimento KEPLER(MM:lógica;NP:numérica);
início
  atribuir 38 a NP;
  se (NP par) então atribuir (NÃO MM) a MM;
fim_procedimento_KEPLER;
início
  atribuir FALSO a BB;
  atribuir 26 a N1; atribuir 17 a N2;
  KEPLER(BB,N1);
  se (NÃO BB) então atribuir (N1 / 2) a N2
  senão atribuir (N1 / 5) a N2;
  Escrever(N1,N2,BB);
fim_programa.
  
```

N1: 26 38  
N2: 17 19  
BB: FALSO

MM: FALSO VERDADEIRO  
NP:

Programação Estruturada – Rodrigo Adur

## Questão 14



[2012 – CONSULPLAN – TSE - Programador de computador]

Ao final da execução, as variáveis N1, N2 e BB terão, respectivamente, os seguintes valores :

- A) 26, 13 e FALSO.
- B) 38, 19 e FALSO.
- C) 38, 19 e VERDADEIRO.
- D) 26, 13 e VERDADEIRO.

N1: 38  
N2: 19  
BB: FALSO

Programação Estruturada – Rodrigo Adur

## Questão 15



[2010 – FCC – TCM-PA – Técnico de Informática]

Extensão natural do conceito de ocultação de informações, que diz: "um módulo deve executar uma única tarefa dentro do procedimento de software, exigindo pouca interação com procedimentos que são executados em outras partes de um programa", é o conceito de:

- A) coesão.
- B) enfileiramento.
- C) acoplamento.
- D) visibilidade.
- E) recursividade.

Programação Estruturada – Rodrigo Adur

## Questão 15



[2010 – FCC – TCM-PA – Técnico de Informática]

Extensão natural do conceito de ocultação de informações, que diz: "um módulo deve executar uma única tarefa dentro do procedimento de software, exigindo pouca interação com procedimentos que são executados em outras partes de um programa", é o conceito de:

- A) coesão.
- B) enfileiramento.
- C) acoplamento.
- D) visibilidade.
- E) recursividade.

Programação Estruturada – Rodrigo Adur

## Questão 16



[2009 – FCC – TRT - 7ª Região (CE) – Analista de Tecnologia da Informação]

No projeto de software, excetuando-se o acoplamento direto entre módulos, o seguinte no espectro (PRESSMAN) e o mais baixo desejado possível é o acoplamento

- A) por dados.
- B) por controle.
- C) por conteúdo.
- D) externo.
- E) comum.

Programação Estruturada – Rodrigo Adur

## Questão 16



[2009 – FCC – TRT - 7ª Região (CE) – Analista de Tecnologia da Informação]

No projeto de software, excetuando-se o acoplamento direto entre módulos, o seguinte no espectro (PRESSMAN) e o mais baixo desejado possível é o acoplamento

- A) por dados.
- B) por controle.
- C) por conteúdo.
- D) externo.
- E) comum.

Níveis de Acoplamento	
De Dados	- Menor nível de acoplamento
De Imagem	
De Controle	
Comum	
De Conteúdo	+ Maior nível de acoplamento

Programação Estruturada – Rodrigo Adur

## Gabarito



- 11 – Errado    14 – B
- 12 – Certo    15 – A
- 13 – Certo    16 – A

Programação Estruturada – Rodrigo Adur



TUDO PARA VOCÊ PASSAR

FIM

Rodrigo Adur  
rodrigoadurti@gmail.com

