


Conceitos Básicos



- ❑ Nesta aula são introduzidos alguns conceitos básicos sobre programação de computadores
- ❑ Serão abordados alguns conceitos sobre sistemas computacionais e algoritmos

José Augusto Baranauskas
 Departamento de Física e Matemática – FFCLRP-USP
 Sala 222 – Bloco P2 – Fone (16) 3602-4361

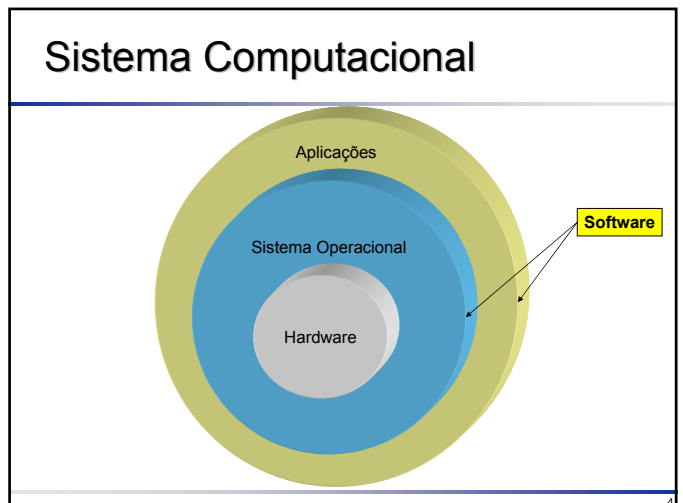
E-mail: augusto@ffclrp.usp.br
 URL: <http://www.fmrp.usp.br/augusto>

Sistema Computacional

- ❑ Componentes Físicos: dispositivos mecânicos, magnéticos, elétricos ou eletrônicos
- ❑ Componentes Lógicos: Programas, métodos e procedimentos, regras e documentação

Sistema Computacional

- ❑ Sistema Operacional
 - Responsável pelo processamento dos programas
 - Assegura que recursos físicos estejam disponíveis quando necessários
 - Assegura que recursos de software sejam fornecidos quando exigidos
 - Cria um ambiente onde os usuários podem preparar e executar seus programas sem se preocuparem com detalhes de hardware
 - Diversos usuários pode usar simultaneamente o sistema;
 - É função do sistema operacional coordenar o compartilhamento dos recursos exigidos pelos usuários



Sistemas de Numeração

- ❑ Há duas formas básicas de armazenamento de dados
 - Analógica: os dados assumem valores contínuos
 - Exemplo: músicas transmitidas por uma estação de rádio, músicas armazenadas em discos de vinil
 - Digital: os dados assumem (poucos) valores discretos
 - Exemplo: músicas em CD
- ❑ Os computadores atuais utilizam o sistema de numeração binário (base 2) para armazenar e processar dados

Sistemas de Numeração

- ❑ Assim como o sistema de numeração decimal (base 10), o sistema binário (base 2) é um sistema **posicional**
- ❑ Isto significa que o valor de um dígito é dados pela sua posição no número
- ❑ No sistema decimal, o número 3453 tem a seguinte interpretação
 - $3 \times 1000 (10^3) = 3000$
 - $+ 4 \times 100 (10^2) = 400$
 - $+ 5 \times 10 (10^1) = 50$
 - $+ 3 \times 1 (10^0) = 3$
 - -----
 - 3453
- ❑ Note que o dígito **3** tem diferentes interpretações em suas duas ocorrências dentro do número

Sistemas de Numeração

- Assim, a interpretação de um número composto por 5 dígitos **abcde** no sistema de numeração com base **n** ($n > 0$) é:
 - $a \times n^4 + b \times n^3 + c \times n^2 + d \times n^1 + e \times n^0$
- Em geral, a interpretação de um número composto por **k** dígitos $d_{k-1}d_{k-2}...d_1d_0$ na base **n** é:
 - $d_{k-1} \times n^{k-1} + d_{k-2} \times n^{k-2} + \dots + d_1 \times n^1 + d_0 \times n^0$
- Portanto, em um sistema posicional de numeração a posição de um dígito no número especifica o expoente da base do sistema de numeração para se obter a contribuição daquele dígito no resultado final

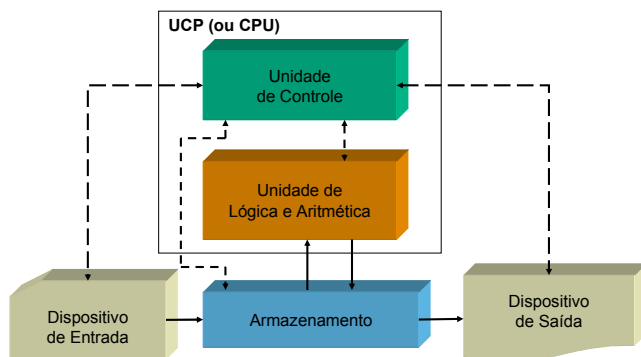
7

Sistemas de Numeração

- Observe que um sistema de numeração base **n**, os dígitos permitidos d_i encontram-se entre **zero** e **n-1**, ou seja ($0 \leq d_i < n$)
- Base 10 (decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Base 2 (binária): 0, 1
- Base 8 (octal): 0, 1, 2, 3, 4, 5, 6, 7
- Base 16 (hexadecimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Por exemplo, o número 110101 na base 2 tem valor decimal igual a 53
 - 1×32 (2^5) = 32
 - $+ 1 \times 16$ (2^4) = 16
 - $+ 0 \times 8$ (2^3) = 0
 - $+ 1 \times 4$ (2^2) = 4
 - $+ 0 \times 2$ (2^1) = 0
 - $+ 1 \times 1$ (2^0) = 1
 - -----
 - 53

8

Componentes de um Computador



9

Unidade Central de Processamento

- A CPU é o "cérebro" do computador;
- Comanda o fluxo de dados no sistema e realiza operações com esses dados
- ULA
 - responsável pelos cálculos
- UC
 - Transfere dados dos dispositivos de entrada, transfere dados da memória principal para a secundária e vice-versa, envia resultados para os dispositivos de saída

10

Memória Principal

- Os programas em execução devem residir juntamente com os dados necessários na memória principal
- É mais cara (que memória secundária) mas mais rápida
- A memória principal é dividida em unidades pequenas e de mesmo tamanho, chamadas **palavras**, sendo que cada palavra de memória tem um único **endereço**
- Cada palavra é capaz de armazenar essencialmente uma única informação, por exemplo, o resultado de uma operação numérica
- O tamanho da palavra de memória (número de bits) determina o maior e o menor número que pode ser armazenado

11

Memória Principal

- Em terminologia de memória, diz-se que qualquer dispositivo capaz de armazenar um dígito binário (apenas dois estados possíveis) define um **bit** de informação
 - bit = binary digit = dígito binário
- Por definição 1 byte = 8 bits
- O tamanho das palavras é normalmente expresso em bits
 - palavras de 8 bits (1 byte), 16 bits (2 bytes), 32 bits (4 bytes), ...

12

Memória Principal

- Pode facilmente observar que o maior valor decimal que se pode representar com n bits é $2^n - 1$
- Na prática, a representação da informação é mais complicada pois diversos tipos de informação são armazenados:
 - Para informações não numéricas (caracteres) a representação é efetuada através de um sistema de codificação no qual, por convenção, certos conjuntos de bits representam certos caracteres
 - ❖ Exemplos: ASCII e EBCDIC
 - Para números inteiros o problema do sinal é tratado normalmente reservando um bit da palavra (geralmente o mais à esquerda) para o **bit de sinal**
 - ❖ 0 indica número positivo
 - ❖ 1 indica número negativo
 - ❖ Se a palavra tem k bits, tem-se $k-1$ bits para o módulo do número
 - A representação de números reais é consideravelmente mais complicada

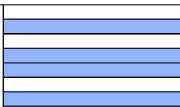


13

Memória Principal

- Para ilustrar como a informação é armazenada, suponha que a memória seja composta por uma série de chaves elétricas de duas posições
- Por convenção, uma chave na posição “ligada” representa o dígito binário 1 e uma chave na posição “desligada” o dígito binário 0
- Na figura seguinte, um retângulo hachurado representa uma chave ligada e um retângulo branco representa uma chave desligada e o tamanho da palavra de memória usado foi de 8 bits

14

Memória Principal

	Endereços	Acionamento de chaves	Dígitos binários Equivalentes	Valores decimais Equivalentes
Palavra (8 bits)	0		01111111	91
	1		01111111	110
	2		10000000	1

15

Tamanho de Memória

- O tamanho de memória é expresso em número de bytes e seus múltiplos
- Em computação, os prefixos K (quilo), M (mega), G (giga), T (tera), P (peta), E (exa), Z (zeta), Y (yota) ... são múltiplos de $1024 (2^{10})$ e não de $1000 (10^3)$
 - 1 Kbyte = $1024 \text{ bytes} \cong 10^3 \text{ bytes}$
 - 1 Mbyte = $1024 \text{ Kbytes} = 1024^2 \text{ bytes} = 1.048.576 \text{ bytes} \cong 10^6 \text{ bytes}$
 - 1 Gbyte = $1024 \text{ Mbytes} = 1024^3 \text{ bytes} = 1.073.741.824 \text{ bytes} \cong 10^9 \text{ bytes}$
 - 1 Tbyte = $1024 \text{ Gbytes} = 1024^4 \text{ bytes} = 1.099.511.627.776 \text{ bytes} \cong 10^{12} \text{ bytes}$
 - 1 Pbyte = $1024 \text{ Tbytes} = 1024^5 \text{ bytes} = 1.125.899.906.842.624 \text{ bytes} \cong 10^{15} \text{ bytes}$
 - ...

16

Memória Principal

- ROM (Read Only Memory)
 - Armazena as informações básicas para a início e operação do computador
 - É permanente, não podendo ser apagada ou alterada
- RAM (Random Access Memory)
 - Espaço de trabalho
 - Volátil: os dados mantidos enquanto a máquina estiver ligada, ou seja, a RAM é mantida apenas se tiver energia disponível

17

Memória Secundária, Auxiliar ou de Massa

- São mais lentas que a memória principal mas são muito maiores e mais baratas e podem armazenar grandes quantidades de dados por longos períodos de tempo
- Os tipos de mídia mais utilizados são:
 - *Floppy Disks* (disquetes);
 - *Hard Disks* (discos rígidos ou winchester);
 - Discos de mídia removível (*Zip disks*, *Memory Sticks* ou similares);
 - Discos ópticos (CD-ROM, CD-R, CD-RW, DVD e similares).

18

Componentes de um Computador

- ❑ Dispositivos de Entrada e Saída (I/O: Input/Output)
 - Fornecem o meio pelo qual os dados são transmitidos ao computador e o resultado é apresentado
 - Os principais dispositivos de entrada são teclado, mouse, leitora de código de barras, scanner, microfone
 - O principal dispositivo de saída é o Monitor. Além do monitor existe a impressora, caixas de som, *plotters*
 - Existem também dispositivos de entrada e saída ao mesmo tempo, como os Fax-Modem, que recebem e transmitem dados

19

Linguagens de Programação

- ❑ Linguagem de Máquina
 - Um programa escrito em linguagem de máquina consiste de uma série de números binários pois é a linguagem compreendida pela CPU
 - Exemplo: 011101010100...
- ❑ Linguagem de Montagem
 - Mnemônicos são utilizados para representar as instruções
 - Exemplo:

❖ Mnemônico	operando	significado
LOAD	10	carrega 10 no acumulador
ADD	5	soma 5 ao conteúdo do acumulador
 - Programas **montadores** são utilizados para traduzir um programa escrito em linguagem de montagem para linguagem de máquina

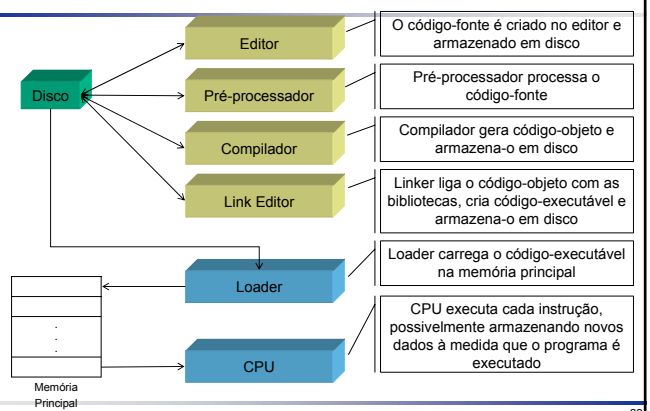
20

Linguagens de Programação

- ❑ Linguagem de Alto Nível
 - A linguagem utilizada é mais próxima da linguagem humana (e não da máquina)
 - Exemplo: `if (delta > 0) x = a/delta;`
 - Programas **compiladores** são utilizados para traduzir um programa escrito em linguagem de alto nível (código-fonte) para código-objeto; o código-objeto então é *link*-editado com bibliotecas para produzir o programa em linguagem de máquina

21

Ambiente de Programação



22

Ambiente de Programação C/C++

- ❑ O código-fonte é editado e armazenado em um arquivo com extensões:
 - Código-fonte em C: extensão `.c` (letra "c" minúscula)
 - Código-fonte em C++: extensões `.cpp`, `.cxx` ou `.C` (letra "C" maiúscula)
- ❑ No código-fonte, todo comando C++ termina com `;`
- ❑ Em seguida, o programador executa o comando para compilar o código-fonte
 - O compilador traduz o programa C++ para linguagem de máquina (código-objeto), com extensão `.obj` ou `.o`
 - Em um sistema C++, o pré-processador é executado automaticamente antes do compilador

23

Ambiente de Programação C/C++

- ❑ Em um sistema C/C++, o pré-processador é executado automaticamente antes do compilador
 - O pré-processador obedece comandos especiais chamados de *diretivas do pré-processador*, que indicam que manipulações devem ser realizadas no programa antes da compilação, tais como a inclusão de outros arquivos no código-fonte a ser compilado e a substituição de textos
 - Todas diretivas começam com `#`
 - Diretivas do pré-processador não são comandos C/C++, assim elas **não** terminam com `;`

24

Ambiente de Programação C/C++

- ❑ Diretivas mais utilizadas em C
 - `#include <stdio.h>`
funções de entrada e saída
 - `#include <math.h>`
funções matemáticas
- ❑ Diretivas mais utilizadas em C++:
 - `#include <iostream>`
funções de entrada e saída
 - `#include <iomanip>`
funções de formatação de entrada e saída
 - `#include <cmath>`
funções matemáticas

25

Ambiente de Programação C/C++

- ❑ A próxima fase é chamada de link-edição (ou edição de ligações)
 - Programas C/C++ tipicamente contém chamadas a funções definidas em outros locais, tais como as bibliotecas padrões ou bibliotecas de um projeto particular
 - O código-objeto produzido contém "buracos" devido a essas chamadas
 - O *linker* liga o código-objeto com o código dessas chamadas para produzir o código executável (sem "buracos")
 - ❖ Ambientes DOS/Windows: extensão `.exe`
 - ❖ Ambientes Unix: arquivo `a.out`

26

Ambiente de Programação C/C++

- ❑ Antes que um programa possa ser executado, ele deve ser colocado na memória principal
 - Esta tarefa é realizada pelo *loader* que transfere o código-executável do disco para a memória
- ❑ Finalmente, o computador, sob o controle da CPU, executa o programa, instrução por instrução
 - Para carregar e executar um programa, basta digitar o nome do código-executável e pressionar *Enter*

27

Exemplo: Programa em C++

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Bem vindo a linguagem C++\n";
    return 0;
}
```

Bem vindo a linguagem C++

28

Algoritmo

- ❑ Qualquer problema computacional pode ser solucionado executando uma série de ações em uma ordem específica
- ❑ Uma especificação para solucionar um problema em termos de
 - ações a serem executadas e
 - a ordem nas quais essas ações devem ser executadasé denominada um *algoritmo*
- ❑ Em outras palavras, um algoritmo é uma seqüência ordenada e sem ambigüidade de passos que levam à solução de um dado problema

29

Algoritmo: Exemplos

- ❑ Uma receita de cozinha
- ❑ No Natal, muitos pais passam horas seguindo algoritmos para montar os novos brinquedos de seus filhos
- ❑ Como outro exemplo, considere o algoritmo de levantar-para-trabalhar de um jovem executivo: (1) levantar da cama; (2) tirar pijamas; (3) tomar banho; (4) vestir-se; (5) tomar café; (6) ir para o trabalho

30

Algoritmo: Exemplos

- ❑ Suponha, entretanto, que os mesmos passos sejam executados em uma ordem ligeiramente diferente: (1) levantar da cama; (2) tirar pijamas; (3) vestir-se (4) tomar banho; (5) tomar café; (6) ir para o trabalho
- ❑ No segundo caso, o executivo chegará no trabalho ensopado
- ❑ Assim, a ordem na qual os passos são executados em um algoritmo é importante

31

Algoritmo: Trocar uma Lâmpada

1. Remova a lâmpada queimada
2. Coloque a nova lâmpada

32

Algoritmo: Trocar uma Lâmpada

1. Remova a lâmpada queimada
 - Posicione a escada debaixo da lâmpada queimada
 - Escolha uma nova lâmpada da mesma potência da queimada
 - Suba na escada até que a lâmpada possa ser alcançada
 - Gire a lâmpada queimada no sentido anti-horário até que se solte
2. Coloque a nova lâmpada

33

Algoritmo: Trocar uma Lâmpada

1. Remova a lâmpada queimada
 - Posicione a escada debaixo da lâmpada queimada
 - Escolha uma nova lâmpada da mesma potência da queimada
 - Suba na escada até que a lâmpada possa ser alcançada
 - Gire a lâmpada queimada no sentido anti-horário até que se solte
2. Coloque a nova lâmpada
 - Posicione a nova lâmpada no soquete
 - Gire-a no sentido horário até que ela se firme
 - Desça da escada

34

Algoritmo: Trocar uma Lâmpada

1. Remova a lâmpada queimada
 - Posicione a escada debaixo da lâmpada queimada
 - Escolha uma nova lâmpada da mesma potência da queimada
 - Suba na escada até que a lâmpada possa ser alcançada
 - Gire a lâmpada queimada no sentido anti-horário até que se solte
2. Coloque a nova lâmpada
 - Posicione a nova lâmpada no soquete
 - Gire-a no sentido horário até que ela se firme
 - Desça da escada

35

Algoritmo: Trocar uma Lâmpada

1. Remova a lâmpada queimada
 - Posicione a escada debaixo da lâmpada queimada
 - Escolha uma nova lâmpada da mesma potência da queimada
 - ❖ Selecione uma candidata à substituição
 - ❖ Se a potência não é a mesma da queimada, repita o processo até encontrar uma que sirva
 - Descarte a lâmpada selecionada
 - Selecione uma nova
 - Suba na escada até que a lâmpada possa ser alcançada
 - Gire a lâmpada queimada no sentido anti-horário até que se solte
2. Coloque a nova lâmpada
 - Posicione a nova lâmpada no soquete
 - Gire-a no sentido horário até que ela se firme
 - Desça da escada

36

Algoritmo: Trocar uma Lâmpada

- ❑ Posicione a escada debaixo da lâmpada queimada
- ❑ Escolha uma nova lâmpada da mesma potência da queimada
 - Selecione uma candidata à substituição
 - Se a potência não é a mesma da queimada, repita o processo até encontrar uma que sirva
 - ❖ Descarte a lâmpada selecionada
 - ❖ Selecione uma nova
- ❑ Repita até que a lâmpada possa ser alcançada
 - Suba um degrau da escada
- ❑ Repita até que a lâmpada fique livre do soquete
 - Gire a lâmpada queimada no sentido anti-horário
- ❑ Posicione a nova lâmpada no soquete
- ❑ Repita até que a nova lâmpada esteja firme no soquete
 - Gire a lâmpada no sentido horário
- ❑ Desça da escada

37

Algoritmo: Trocar uma Lâmpada

- ❑ Posicione a escada debaixo da lâmpada queimada
- ❑ Escolha uma nova lâmpada da mesma potência da queimada
 - Selecione uma candidata à substituição
 - Se a potência não é a mesma da queimada, repita o processo até encontrar uma que sirva
 - ❖ Descarte a lâmpada selecionada
 - ❖ Selecione
- ❑ Repita até que a lâmp
- Suba um degrau
- ❑ Repita até que a lâmp
- Gire a lâmpada
- ❑ Posicione a nova lâmp
- ❑ Repita até que a nov
- Gire a lâmpada
- ❑ Desça da escada

Você pode estar pensando: "Eu realizo essa atividade de maneira diferente". Isso pode acontecer, pois um mesmo problema pode ser solucionado de maneiras diferentes, porém gerando a mesma resposta. Assim, podem existir vários algoritmos para resolver um mesmo problema.

38

Algoritmo: Trocar uma Lâmpada

- ❑ Esse processo de aumentar o detalhamento de um algoritmo pode continuar quase que indefinidamente
- ❑ O algoritmo para trocar uma lâmpada mostra como os algoritmos podem ser expressos: operações simples e sem ambigüidade; a ordem na qual os passos devem ser seguidos é claramente expressa

39

Representação de Algoritmos

- ❑ Pseudo-código (ou português estruturado)
- ❑ Fluxograma
- ❑ Diagrama de Nassi-Shneiderman
- ❑ Diagrama de Booch
- ❑ Diagrama UML
- ❑ ...

Programas escritos em uma das representações não são executados diretamente em computadores; eles auxiliam os programadores a pensarem sobre um programa antes de tentar escrevê-lo em uma linguagem de programação como C, C++ ou Pascal.

40

Representação de Algoritmos

- ❑ Independentemente da representação utilizada, todo algoritmo pode ser escrito por meio do uso de três estruturas básicas de controle:
 - **seqüência**: um comando é executado após o comando anterior
 - **seleção**: comandos alternativos são executados dependendo do valor de uma condição (que assume valor ou verdadeiro ou falso) e
 - **repetição**: uma seqüência de comandos é executada zero, uma ou mais vezes
 - ❖ dependendo de um número pré-estabelecido de repetições (laço contado)
 - ❖ o laço é repetido enquanto uma condição é verdadeira (laço condicional)

41

Pseudo-código (PS)

- ❑ É uma linguagem artificial que auxilia os programadores no desenvolvimento de algoritmos
- ❑ É similar ao português; é conveniente e amigável, embora não seja uma linguagem de programação de computadores

42

Pseudo-código: Exemplo 1

- ❑ Faça um algoritmo para mostrar o resultado da multiplicação de dois números
- ❑ Algoritmo em pseudo-código

```

Algoritmo Multiplicar
Início
  declare N1, N2, M : inteiro

  Escreva("Digite dois números")
  Leia(N1,N2)
  M ← N1 * N2
  Escreva(M)
Fim
    
```

43

Fluxograma

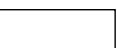
- ❑ Consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos pré-definidos, os passos a serem executados para a resolução do problema
- ❑ O entendimento de símbolos gráficos é mais fácil que o de textos
- ❑ Para algoritmos complexos, pode dificultar a programação de forma estruturada

44

Fluxograma: Principais Símbolos



- ❑ Indica o início e o fim do algoritmo
- ❑ Indica o sentido do fluxo de dados e é utilizado para conectar os demais símbolos entre si



- ❑ Cálculos e atribuições de valores



- ❑ Entrada de dados



- ❑ Saída de dados



- ❑ Tomada de decisão, com possibilidade de desvios

45

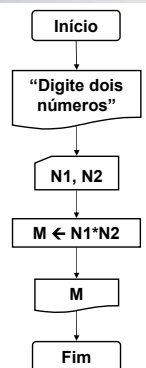
Fluxograma: Exemplo 1

- ❑ Faça um algoritmo para mostrar o resultado da multiplicação de dois números
- ❑ Algoritmo em pseudo-código

```

Algoritmo Multiplicar
Início
  declare N1, N2, M : inteiro

  Escreva("Digite dois números")
  Leia(N1,N2)
  M ← N1 * N2
  Escreva(M)
Fim
    
```



46

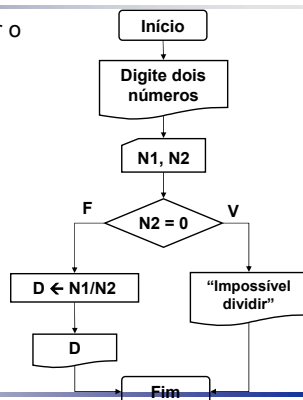
Fluxograma: Exemplo 2

- ❑ Faça um algoritmo para mostrar o resultado da divisão de dois números
- ❑ Algoritmo em pseudo-código

```

Algoritmo Dividir
Início
  declare N1, N2, D : inteiro

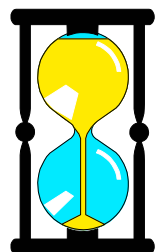
  Escreva("Digite dois números")
  Leia(N1,N2)
  Se N2 = 0 Então
    Escreva("Impossível dividir")
  senão
    D ← N1 / N2
    Escreva(D)
  Fim se
Fim
    
```



47

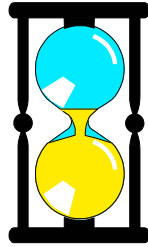
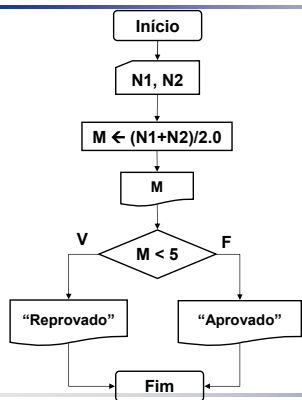
Questão

- ❑ Utilize essas idéias para escrever um algoritmo utilizando fluxograma para calcular a média aritmética entre duas notas de um aluno e mostrar a média e a situação do aluno, que pode ser aprovado ou reprovado
- ❑ Você tem 5 minutos para escrever o algoritmo



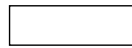
48

Solução



49

Diagrama NS: Símbolos



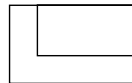
☐ Cálculos e atribuições de valores, entrada e saída de dados



☐ Tomada de decisão, com possibilidade de desvios



☐ Laço com teste no início (enquanto)



☐ Laço com teste no final (repita ... enquanto)

50

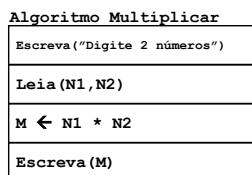
Diagrama NS: Exemplo 1

- ☐ Faça um algoritmo para mostrar o resultado da multiplicação de dois números
- ☐ Algoritmo em pseudo-código

```

Algoritmo Multiplicar
Início
  declare N1, N2, M : inteiro

  Escreva("Digite dois números")
  Leia(N1, N2)
  M ← N1 * N2
  Escreva(M)
Fim
  
```



51

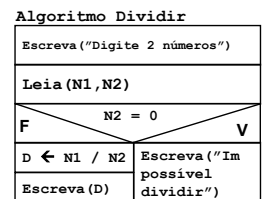
Diagrama NS: Exemplo 2

- ☐ Faça um algoritmo para mostrar o resultado da divisão de dois números
- ☐ Algoritmo em pseudo-código

```

Algoritmo Dividir
Início
  declare N1, N2, D : inteiro

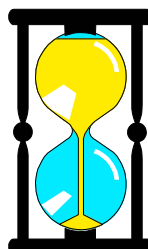
  Escreva("Digite dois números")
  Leia(N1, N2)
  Se N2 = 0 Então
    Escreva("Impossível dividir")
  senão
    D ← N1 / N2
    Escreva(D)
  Fim se
Fim
  
```



52

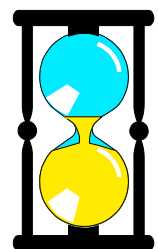
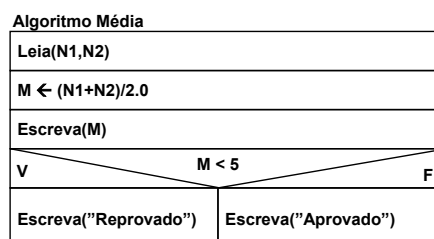
Questão

- ☐ Utilize essas idéias para escrever um algoritmo utilizando diagrama de NS para calcular a média aritmética entre duas notas de um aluno e mostrar a média e a situação do aluno, que pode ser aprovado ou reprovado
- ☐ Você tem 5 minutos para escrever o algoritmo



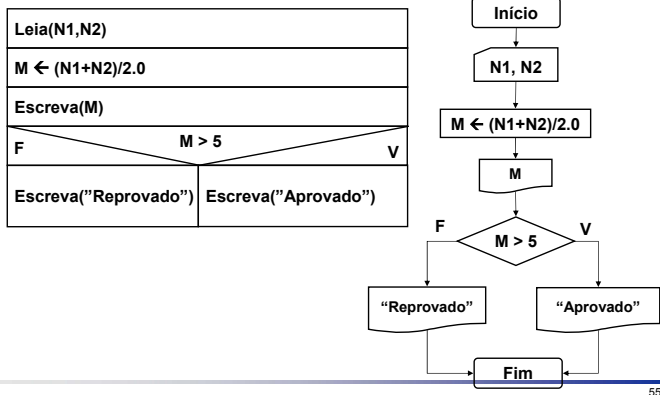
53

Solução



54

Solução: NS e Fluxograma



Pseudo-Código & C++

- ❑ Nos slides seguintes são fornecidos conceitos adicionais, inicialmente em pseudo-código
 - O objetivo da utilização do pseudo-código é tornar claro o conceito em questão e que é válido para **qualquer** linguagem de programação procedural
 - É importante lembrar que uma vez escrito um algoritmo em pseudo-código, ele pode ser traduzido facilmente para **qualquer** linguagem de programação procedural
- ❑ Logo após o conceito em pseudo-código é mostrado como ele é implementado na linguagem de programação C++ ou algum detalhe mais específico da linguagem

Tipos de Dados

- ❑ Numérico
 - Inteiro (3, 7, -6, 7829)
 - Real (23.8, 3.683, -6281.232, 2.71e15)
 - ❖ O ponto (e não a vírgula) é utilizado como separador decimal
 - ❖ 2.71e15 significa 2.71×10^{15}
- ❑ Caractere: um único caractere ('a', '2', 'X')
 - Apóstrofes ' são usados como delimitadores
- ❑ Cadeia de caracteres ou *string* ("abc", "ana paula", "3 + 4 = 7")
 - Aspas " são usadas como delimitadores
- ❑ Lógico ou booleano (verdadeiro/falso; true/false)
- ❑ Ponteiro

Tipos de Dados C/C++

- ❑ Inteiro
 - enum, unsigned int, short int, **int**, unsigned long, long
- ❑ Real
 - **float**, double, long double
- ❑ Caractere
 - char, unsigned char
- ❑ Cadeia de caracteres ou *string*
 - char [] (C/C++)
 - string (C++)
- ❑ Lógico ou booleano
 - bool (C++)
 - Em C é necessário definir: enum bool {false, true};
- ❑ Ponteiro

Tipos de Dados C/C++

A quantidade de bytes e intervalos podem variar de compilador para compilador

Tipo de Dado	Bytes	Intervalo de Valores	Dígitos de Precisão
enum	2	-32 768 até 32 767	5
unsigned int	2	0 até 65 535	5
short int	2	-32 768 até 32 767	5
int	2	-32 768 até 32 767	5
unsigned long	4	0 até 4 294 967 295	10
long	4	-2 147 483 648 até 2 147 483 647	10
float	4	±3.4e±38	7
double	8	±1.7e±308	15
long double	10	±1.2e±4932	19
char	1	'a', 'b', ..., 'z'; 'A', 'B', ..., 'Z'; '0', '1', ..., '9', ...	-
bool	1	false , true	-
ponteiro	4	-	-

Exercício

Indique o tipo de cada uma das seguintes constantes

- ❑ 10
- ❑ 10.0
- ❑ -10
- ❑ "10"
- ❑ 6.02e23
- ❑ "2 + 3 = 5"
- ❑ -2e-5
- ❑ "3e-5"
- ❑ 0.1234
- ❑ "fim de questão"
- ❑ true

Solução

Indique o tipo de cada uma das seguintes constantes

- 10 (inteiro)
- 10.0 (real)
- 10 (inteiro)
- "10" (string)
- 6.02e23 (real)
- "2 + 3 = 5" (string)
- 2e-5 (real)
- "3e-5" (string)
- 0.1234 (real)
- "fim de questão" (string)
- true (booleano)

61

Operações Primitivas

- Adição e subtração são representadas de forma matemática usual
- Multiplicação
 - Para evitar a possível confusão com a letra **x**, a multiplicação é indicada por um ***** (asterisco)
- Divisão $\frac{8}{2}$ é representada como 8/2
- Exponenciação 2^4 é representada por 2^4

62

Operações Primitivas C/C++

- Adição: 2 + 3 + 4
- Subtração: 10 - 4 - 1
- Multiplicação: 2 * 3 * 4
- Divisão: 10.0 / 4
- Exponenciação: `pow(3,2)`
 - Em C/C++ não existe o operador `^`
 - Ele pode ser substituído pela função embutida `pow(x,y)`
 - `pow(3,2)` é equivalente a 3^2

63

Operações Primitivas C/C++

- Além dos operadores aritméticos convencionais, existem outros operadores ou funções embutidas:
 - Resto da divisão
 - Em C/C++: representado por `%`
 - Ex: 9 % 4 resulta em 1; 27 % 5 resulta em 2
 - Raiz quadrada
 - Em C/C++: representado por `sqrt(expressão)`
 - Ex: `sqrt(16)` resulta em 4; `sqrt(25)` resulta em 5; `sqrt(25 + 11)` resulta em 6

64

Funções Embutidas

- Funções embutidas são rotinas pré-escritas, fornecidas pelos projetistas da linguagem de programação
- O conjunto de funções embutidas pode variar para cada linguagem de programação
- Por exemplo, a função `sqrt` denomina a operação de raiz quadrada

65

Algumas Funções Embutidas C/C++

- C: `#include <math.h>`
- C++: `#include <cmath>`

Função	Significado
<code>abs(x)</code>	Valor absoluto de x
<code>sqrt(x)</code>	Raiz quadrada de x
<code>log(x)</code>	Logaritmo base e (logaritmo neperiano ou natural)
<code>log10(x)</code>	Logaritmo base 10
<code>exp(x)</code>	Exponencial, o resultado é e^x
<code>sin(x)</code>	Seno de x; x em radianos
<code>cos(x)</code>	Co-seno de x; x em radianos
<code>tan(x)</code>	Tangente de x; x em radianos
<code>floor(x)</code>	Valor truncado para o maior inteiro menor ou igual a x
<code>ceil(x)</code>	Valor arredondado para o menor inteiro maior ou igual a x
<code>pow(x,y)</code>	x elevado a y: x^y

x	floor(x)	ceil(x)
2.00	2.00	2.00
2.15	2.00	3.00
2.30	2.00	3.00
2.45	2.00	3.00
2.60	2.00	3.00
2.75	2.00	3.00
2.90	2.00	3.00
3.05	3.00	4.00
3.20	3.00	4.00
3.35	3.00	4.00
3.50	3.00	4.00
3.65	3.00	4.00
3.80	3.00	4.00
3.95	3.00	4.00
4.10	4.00	5.00

67

Operações Primitivas

- O resultado de uma operação com os dois operandos inteiros é inteiro
- O resultado de uma operação com um operando real e o outro real ou inteiro é real
- Qual o resultado da expressão?

$$\frac{1}{10} \times 10$$

68

Operações Primitivas

- O resultado de uma operação com os dois operandos inteiros é inteiro
- O resultado de uma operação com um operando real e o outro real ou inteiro é real
- Qual o resultado da expressão?

$$\frac{1}{10} \times 10$$

O resultado 0.1 é truncado para zero pois o resultado da divisão de dois inteiros deve ser um inteiro

- $1 / 10 * 10 = 0.1 * 10 = 0 * 10 = 0$

69

Operações Primitivas

- Para solucionar esta questão, é necessário transformar um dos operandos em real

$$\frac{1.0}{10} \times 10 \quad \text{ou} \quad \frac{1}{10.0} \times 10$$

70

Exercício

- Indique o resultado e o tipo de cada uma das seguintes expressões:
 - $1 + 4 - 2$
 - $2 + 3 * 4$
 - $3 * 4.0 - 2$
 - $3 * 4 - 2.0$
 - $29.0 / 9 + 4$
 - $1 / 4 + 2$
 - $1.0 / 4 + 2$
 - $1 / 4.0 + 2$
 - $1 / 4 + 2.0$
 - $5 ^ 10 + 2$
 - $3.0 ^ 5.0 + 1$

71

Solução

- Indique o resultado e o tipo de cada uma das seguintes expressões:
 - $1 + 4 - 2 = 3$ (inteiro)
 - $2 + 3 * 4 = 14$ (inteiro)
 - $3 * 4.0 - 2 = 10.0$ (real)
 - $3 * 4 - 2.0 = 10.0$ (real)
 - $29.0 / 9 + 4 = 7.22$ (real)
 - $1 / 4 + 2 = 2$ (inteiro)
 - $1.0 / 4 + 2 = 2.25$ (real)
 - $1 / 4.0 + 2 = 2.25$ (real)
 - $1 / 4 + 2.0 = 2.0$ (real)
 - $5 ^ 10 + 2 = 9765627$ (inteiro)
 - $3.0 ^ 5.0 + 1 = 244.0$ (real)

72

Variável

- Uma **variável** é uma entidade que possui um valor, sendo conhecida no programa por um **nome** (ou **identificador**)
- Uma variável representa alguma coisa, especificamente um dado em uma expressão
- Uma variável pode receber muitos valores diferentes em um programa, mas em um determinado momento no tempo ela possui um único valor
- Como regra geral, assumir que toda variável declarada que não recebeu um valor (por meio de uma atribuição ou leitura), contém um valor que é considerado "lixo", ou seja, algo que não tem valor semântico para o programa e esse valor pode mudar a cada execução do programa

73

Variável

- ❑ Existem algumas regras simples para dar nome a uma variável, podendo variar dependendo da linguagem de programação
- ❑ Em geral, o nome de uma variável segue a sintaxe de formação de um *identificador*:
 - começa sempre com uma letra; os demais caracteres podem ser letras ou números e alguns símbolos especiais (e.g., sublinhado)
 - brancos não são permitidos
 - Na formação de um identificador, algumas linguagens diferenciam letras minúsculas de maiúsculas
- ❑ Algumas linguagens de programação requerem que uma variável seja **declarada** antes de ser utilizada

74

Variável: Exemplos

- ❑ Nomes válidos para variáveis
 - X
 - SOMA
 - A123
 - LadoEsquerdo
 - CAIXA_AMARELA
 - Um_Nome_Longo
- ❑ Note a utilização do símbolo de sublinhado “_” (ou *underscore*) nos exemplos
- ❑ Nomes inválidos para variáveis
 - 2X (não pode começar com número)
 - X+Y (“+” não é permitido)
 - Duas Palavras (espaço em branco não é permitido)
- ❑ Sempre escolha nomes significativos para variáveis

75

Variável

```
Algoritmo DeclaraVariaveis
Inicio
  // declaração da variável "a" inteira
  declare a : inteiro

  // declaração das variáveis "b" e "c" inteiras
  declare b,c : inteiro

  // declaração das variável inteira "Quantidade"
  declare Quantidade : inteiro

  // declaração de 5 variáveis reais
  declare X,Y,Z,Hipotenusa,A : real
Fim
```

76

Variável C/C++

- ❑ Toda variável deve ser declarada antes de ser utilizada
- ❑ Há diferença entre letras minúsculas de maiúsculas
 - Em C/C++ a variável **X** é diferente da variável **x**, ou seja, **X** representa uma variável e **x** representa outra variável que é distinta de **X**

77

Exemplo C++

```
#include <iostream>
using namespace std;

int main()
{ // declaração da variável "a" inteira
  int a;

  // declaração das variáveis "b" e "c" inteiras
  int b,c;

  // declaração das variável inteira "Quantidade"
  int Quantidade;

  // declaração de 5 variáveis reais
  float X,Y,Z,Hipotenusa,A;

  return 0;
}
```

78

Operação de Atribuição

- ❑ É a forma de especificar que a uma variável será dado um valor
- ❑ O comando de atribuição é indicado pelo símbolo **←** cuja forma geral é:
 - **<variável> ← <expressão>**
- ❑ O operador **←** separa os componentes em dois lados: esquerdo e direito
 - O **lado esquerdo** (do operador **←**) deve ser **sempre** uma variável
 - O **lado direito** (do operador **←**) é uma expressão

79

Operação de Atribuição

Algoritmo DeclaraAtribui

→ Início

declare A: inteiro

A ← 3

Fim

Memória	
Endereço	Valor

80

Operação de Atribuição

Algoritmo DeclaraAtribui

Início

→ declare A: inteiro

A ← 3

Fim

Memória	
Endereço	Valor
A	

81

Operação de Atribuição

Algoritmo DeclaraAtribui

Início

declare A: inteiro

→ A ← 3

Fim

Memória	
Endereço	Valor
A	3

82

Operação de Atribuição

Algoritmo DeclaraAtribui

Início

declare A: inteiro

A ← 3

→ Fim

Memória	
Endereço	Valor

83

Operação de Atribuição

- O comando
 - A ← 3
 - Indica que à variável **A** é atribuído o valor 3
- A variável **A** pode ser vista como uma palavra na memória do computador
- Após a execução do comando de atribuição, a variável de nome **A** conterá o número 3
- Desde que a memória só pode conter um único valor por vez, o número 3 substitui qualquer valor que a variável possuía anteriormente

84

Operação de Atribuição

- Assim, a atribuição é uma operação **destrutiva**
- Se a seqüência de atribuições for executada:
 - A ← 10
 - A ← -2
 - A ← 1
- Qual o valor da variável A no final?

85

Operação de Atribuição

Assim, a atribuição é uma operação **destrutiva**

Se a seqüência de atribuições for executada:

- A ← 10
- A ← -2
- A ← 1

Qual o valor da variável A no final?

Memória	
Endereço	Valor
A	10

86

Operação de Atribuição

Assim, a atribuição é uma operação **destrutiva**

Se a seqüência de atribuições for executada:

- A ← 10
- A ← -2
- A ← 1

Qual o valor da variável A no final?

Memória	
Endereço	Valor
A	-2

87

Operação de Atribuição

Assim, a atribuição é uma operação **destrutiva**

Se a seqüência de atribuições for executada:

- A ← 10
- A ← -2
- A ← 1

Qual o valor da variável A no final?

Memória	
Endereço	Valor
A	1

88

Operação de Atribuição

Assim, a atribuição é uma operação **destrutiva**

Se a seqüência de atribuições for executada:

- A ← 10
- A ← -2
- A ← 1

Qual o valor da variável A no final?

- O valor da variável A após as três operações é 1
- Os valores 10 e -2 foram destruídos

Memória	
Endereço	Valor
A	1

89

Operação de Atribuição C/C++

Em C/C++, o operador de atribuição é o símbolo =

A linguagem permite atribuições sucessivas. Por exemplo, o comando

- a = b = c = 15;

atribui o valor 15 às variáveis a, b e c

A linguagem também permite declarar e atribuir um valor inicial simultaneamente

- int x = 0;
 - float a, b = 2;
- declara variável x inteira e atribui o valor 0 à x
declara variáveis a e b reais e atribui valor 2 somente à variável b (a variável a possui um valor indefinido ou "lixo")

90

Exemplo 1

Algoritmo Exemplo1

Início

```
declare a, b : inteiro
```

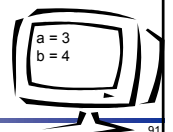
```
a ← 3
```

```
b ← 4
```

```
Escreva("a = ", a)
```

```
Escreva("b = ", b)
```

Fim



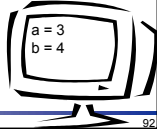
91

Exemplo 1 C++

```
#include <iostream>
using namespace std;

int main()
{ int a,b;

  a = 3;
  b = 4;
  cout << "a = " << a << endl;
  cout << "b = " << b << endl;
  return 0;
}
```



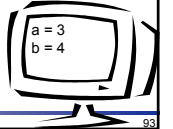
92

Exemplo 1 C++

```
#include <iostream>
using namespace std;

int main()
{ int a=3,b=4;

  cout << "a = " << a << endl;
  cout << "b = " << b << endl;
  return 0;
}
```



93

Expressões

- ❑ Expressão é uma combinação válida de variáveis, constantes e operadores
- ❑ No comando de atribuição, o resultado da avaliação da expressão é o valor que é atribuído à variável indicada
- ❑ Ex:
 - $K \leftarrow 3 + 15 + 2$
 - Então 20 é atribuído à variável K

94

Expressões

- ❑ Uma expressão pode conter variáveis, cujos valores devem ser previamente atribuídos
- ❑ Ex:
 - $\text{Termo1} \leftarrow 14.6 + 6.4$
 - $\text{Termo2} \leftarrow 0.7 + 19.32$
 - $\text{Resultado} \leftarrow \text{Termo1} / \text{Termo2}$
 - Então
 - ❖ 21.0 é atribuído a Termo1
 - ❖ 20.02 é atribuído a Termo2
 - ❖ 1.048951 é atribuído a Resultado

95

Expressões

- ❑ Qualquer variável utilizada em uma expressão deve ter um valor no momento em que a expressão é avaliada
- ❑ Ex:
 - $\text{Termo1} \leftarrow 14.6 + 6.4$
 - $\text{Resultado} \leftarrow \text{Termo1} / \text{Termo2}$
 - $\text{Termo2} \leftarrow 0.7 + 19.32$
 - ❖ Há um erro de programação, embora os mesmos comandos estejam presentes
 - ❖ Quando a expressão " $\text{Termo1} / \text{Termo2}$ " está para ser avaliada, a variável Termo2 ainda não recebeu um valor (normalmente, tem um valor mas é considerado como "lixo")
- ❑ É de responsabilidade do programador assegurar que todas as variáveis que aparecem em uma expressão tenham valores no momento em que ela é avaliada

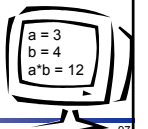
96

Exemplo 2

Algoritmo Exemplo2

```
Início
  declare a,b : inteiro

  a ← 3
  b ← 4
  Escreva("a = ", a)
  Escreva("b = ", b)
  Escreva("a*b = ", a*b)
Fim
```



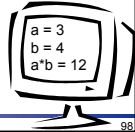
97

Exemplo 2 C++

```
#include <iostream>
using namespace std;

int main()
{ int a,b;

  a = 3;
  b = 4;
  cout << "a = " << a << endl;
  cout << "b = " << b << endl;
  cout << "a*b = " << a*b << endl;
  return 0;
}
```



Exemplo 3

Algoritmo Exemplo3

Início

declare a,b,x : inteiro

a ← 3

b ← 4

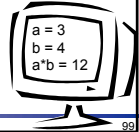
x ← a * b

Escreva("a = ", a)

Escreva("b = ", b)

Escreva("a*b = ", x)

Fim

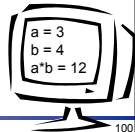


Exemplo 3 C++

```
#include <iostream>
using namespace std;

int main()
{ int a,b,x;

  a = 3;
  b = 4;
  x = a * b;
  cout << "a = " << a << endl;
  cout << "b = " << b << endl;
  cout << "a*b = " << x << endl;
  return 0;
}
```



Alterando um Valor Armazenado

- Suponha que **X** seja uma variável inteira e que temos um valor armazenado nela
- Se um novo valor for atribuído a **X** ele substituirá o valor anterior
- Seja **A** uma variável inteira e considere os seguintes comandos:
 - X ← 0
 - A ← 0
 - X ← A + 1

Memória	
Endereço	Valor
X	
A	

Alterando um Valor Armazenado

- Suponha que **X** seja uma variável inteira e que temos um valor armazenado nela
- Se um novo valor for atribuído a **X** ele substituirá o valor anterior
- Seja **A** uma variável inteira e considere os seguintes comandos:
 - X ← 0
 - A ← 0
 - X ← A + 1

Memória	
Endereço	Valor
X	0
A	

Alterando um Valor Armazenado

- Suponha que **X** seja uma variável inteira e que temos um valor armazenado nela
- Se um novo valor for atribuído a **X** ele substituirá o valor anterior
- Seja **A** uma variável inteira e considere os seguintes comandos:
 - X ← 0
 - A ← 0
 - X ← A + 1

Memória	
Endereço	Valor
X	0
A	0

Alterando um Valor Armazenado

- Suponha que **X** seja uma variável inteira e que temos um valor armazenado nela
- Se um novo valor for atribuído a **X** ele substituirá o valor anterior
- Seja **A** uma variável inteira e considere os seguintes comandos:
 - $X \leftarrow 0$
 - $A \leftarrow 0$
 - ➔ ▪ $X \leftarrow A + 1$

Memória	
Endereço	Valor
X	1
A	0

104

Alterando um Valor Armazenado

- Suponha que **X** seja uma variável inteira e que temos um valor armazenado nela
- Se um novo valor for atribuído a **X** ele substituirá o valor anterior
- Seja **A** uma variável inteira e considere os seguintes comandos:
 - $X \leftarrow 0$
 - $A \leftarrow 0$
 - $X \leftarrow A + 1$
- **X** e **A** recebem o valor zero (nos dois primeiros comandos)
- No terceiro comando **X** recebe o valor da variável **A** adicionado em uma unidade
- Assim, $X \leftarrow A + 1$ pode ser lido como “tome o valor atual da variável **A** (que é zero), adicione 1 a ele e atribua o resultado à variável **X**”
- Após o terceiro comando, **X** tem o valor 1 enquanto **A** mantém o valor zero

105

Alterando um Valor Armazenado

- Suponha que tivéssemos escrito
 - $X \leftarrow 0$
 - $X \leftarrow X + 1$

Memória	
Endereço	Valor
X	

106

Alterando um Valor Armazenado

- Suponha que tivéssemos escrito
 - ➔ ▪ $X \leftarrow 0$
 - $X \leftarrow X + 1$

Memória	
Endereço	Valor
X	0

107

Alterando um Valor Armazenado

- Suponha que tivéssemos escrito
 - $X \leftarrow 0$
 - ➔ ▪ $X \leftarrow X + 1$

Memória	
Endereço	Valor
X	1

108

Alterando um Valor Armazenado

- Suponha que tivéssemos escrito
 - $X \leftarrow 0$
 - $X \leftarrow X + 1$
- **X** recebe o valor zero no primeiro comando
- No segundo comando, a variável **X** recebe o valor da variável **X** adicionado em uma unidade
- Assim, $X \leftarrow X + 1$ também é lido como “tome o valor atual da variável **X** (que é zero), adicione 1 a ele e atribua o resultado à variável **X**”
- Após o segundo comando, **X** tem o valor 1
- Modificamos o valor da variável **X** adicionando 1 a ela

109

Alterando um Valor Armazenado

- ❑ Note que $X \leftarrow X + 1$ não significa que X é igual a $X+1$
- ❑ O símbolo \leftarrow significa **atribuição** e não **igualdade**, já que isso não tem sentido matematicamente
- ❑ O aparecimento de uma variável no lado esquerdo de um comando de atribuição indica que seu valor deve ser **alterado**
- ❑ O aparecimento de uma variável no lado direito de um comando de atribuição indica que seu valor deve ser **utilizado**
- ❑ $X \leftarrow X + 1$ significa o aumento do valor de X em 1

110

Alterando um Valor Armazenado C/C++

- ❑ A linguagem permite que o comando
 - $X = X + 1$
- ❑ também seja representado como
 - $X += 1$
- ❑ ou como (forma mais comum)
 - $X++$
- ❑ Em geral, o comando
 - $X = X + a$
- ❑ também pode ser representado como
 - $X += a$
- ❑ Analogamente para as demais operações primitivas

Pseudo-Código	C/C++
$X \leftarrow X + 1$	$X = X + 1;$
	$X++;$
	$++X;$
$X \leftarrow X - 1$	$X -= 1;$
	$X--;$
	$--X;$
	$X -= 1;$

111

Alterando um Valor Armazenado C/C++

- ❑ A linguagem permite que o comando
 - $X = X + 1$
- ❑ também seja representado como
 - $X += 1$
- ❑ ou como (forma mais comum)
 - $X++$
- ❑ Em geral, o comando
 - $X = X + a$
- ❑ também pode ser representado como
 - $X += a$
- ❑ Analogamente para as demais operações primitivas

Pseudo-Código	C/C++
$X \leftarrow X + a$	$X = X + a;$
	$X += a;$
$X \leftarrow X - a$	$X = X - a;$
	$X -= a;$
$X \leftarrow X * a$	$X = X * a;$
	$X *= a;$
$X \leftarrow X / a$	$X = X / a;$
	$X /= a;$
$X \leftarrow X \% a$	$X = X \% a;$
	$X \% = a;$

112

Exemplo 4

Algoritmo Exemplo4

Início

declare a : inteiro

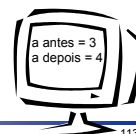
a ← 3

Escreva("a antes = ", a)

a ← a + 1

Escreva("a depois = ", a)

Fim



113

Exemplo 4 C++

```
#include <iostream>
using namespace std;

int main()
{ int a;

  a = 3;
  cout << "a antes = " << a << endl;
  a = a + 1;
  cout << "a depois = " << a << endl;
  return 0;
}
```



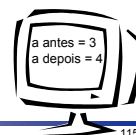
114

Exemplo 4 C++

```
#include <iostream>
using namespace std;

int main()
{ int a;

  a = 3;
  cout << "a antes = " << a << endl;
  a++;
  cout << "a depois = " << a << endl;
  return 0;
}
```



115

Prioridade dos Operadores

- ❑ Considere o seguinte comando
 - $X \leftarrow 3 + 6 * 10$
- ❑ Qual o valor recebido pela variável X?

116

Prioridade dos Operadores

- ❑ Considere o seguinte comando
 - $X \leftarrow 3 + 6 * 10$
- ❑ Qual o valor recebido pela variável X?
- ❑ Depende da ordem na qual os operadores matemáticos (* e +) são processados
- ❑ Se processarmos da esquerda para a direita, o resultado é 90; da direita para esquerda é 63

$$\begin{array}{c} X \leftarrow 3 + 6 * 10 \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 9 \quad \quad \quad \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 90 \end{array}$$

$$\begin{array}{c} X \leftarrow 3 + 6 * 10 \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 60 \quad \quad \quad \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 63 \end{array}$$

117

Prioridade dos Operadores

- ❑ Para eliminar essa ambigüidade, a matemática definiu regras adicionais para a avaliação de expressões
- ❑ À cada operador é associada uma prioridade
- ❑ Operadores com maior prioridade são processados em primeiro lugar, da esquerda para a direita
- ❑ Por exemplo, a multiplicação tem maior prioridade que a adição
- ❑ Então na expressão $3 + 6 * 10$, o termo $6 * 10$ é processado em primeiro lugar, resultando em 60; a seguir, o operador de adição é processado, somando 3 com 60, resultando em 63

118

Prioridade dos Operadores

$$8 + 7 * 3 + 4 * 5$$

119

Prioridade dos Operadores

$$\begin{array}{c} 8 + 7 * 3 + 4 * 5 \\ \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 21 \end{array}$$

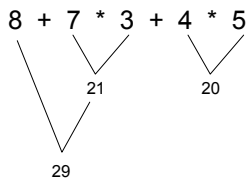
120

Prioridade dos Operadores

$$\begin{array}{c} 8 + 7 * 3 + 4 * 5 \\ \quad \quad \quad \swarrow \quad \searrow \quad \quad \quad \swarrow \quad \searrow \\ \quad \quad \quad 21 \quad \quad \quad 20 \end{array}$$

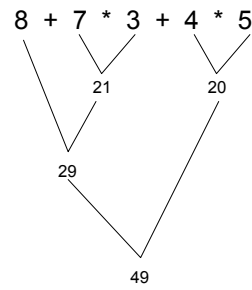
121

Prioridade dos Operadores



122

Prioridade dos Operadores



123

Prioridade dos Operadores

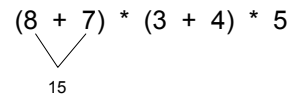
- ❑ Parênteses podem ser usados para alterar a prioridade nas operações
- ❑ Operações entre parênteses são processadas em primeiro lugar

$$(8 + 7) * (3 + 4) * 5$$

124

Prioridade dos Operadores

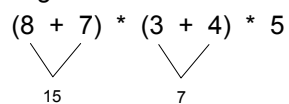
- ❑ Parênteses podem ser usados para alterar a prioridade nas operações
- ❑ Operações entre parênteses são processadas em primeiro lugar



125

Prioridade dos Operadores

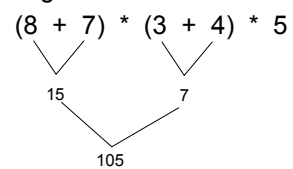
- ❑ Parênteses podem ser usados para alterar a prioridade nas operações
- ❑ Operações entre parênteses são processadas em primeiro lugar



126

Prioridade dos Operadores

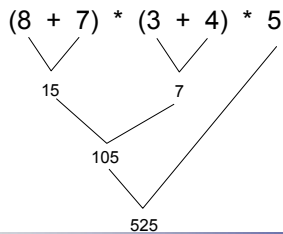
- ❑ Parênteses podem ser usados para alterar a prioridade nas operações
- ❑ Operações entre parênteses são processadas em primeiro lugar



127

Prioridade dos Operadores

- Parênteses podem ser usados para alterar a prioridade nas operações
- Operações entre parênteses são processadas em primeiro lugar



128

Prioridade dos Operadores

- Operadores podem ser agrupados em classes
- Exponenciação tem maior prioridade; seguida por mais e menos unários; seguidos pela multiplicação, módulo e divisão e finalmente adição e subtração
- Operadores da mesma classe têm a mesma prioridade
- Note ordem da aplicação da exponenciação: $2^3 2 = 2^{3^2} = 2^9 = 512$

Classe	Operador	Significado
1	()	Parênteses mais internos
2	sqrt, log...	Funções Embutidas
3	^	Exponenciação (aplicada da direita para a esquerda)
4	-, +	Sinal dos operandos (menos e mais unários aplicados da direita para a esquerda)
5	*, %, /	Multiplicação, módulo e divisão (aplicadas da esquerda para a direita)
6	+, -	Adição e subtração (aplicadas da esquerda para a direita)

129

Exemplo

- Escreva as seguintes expressões matemáticas como expressões de computador

$$\frac{a}{bc} \quad \frac{a + |b|}{c + d} \quad \frac{a + \frac{b}{\sin(c)}}{d - \sqrt{e}}$$

$a/b/c$ ou $a/(b*c)$ $(a+abs(b))/(c+d)$ $(a+b/sin(c))/(d-sqrt(e))$

130

Exercício

- Assuma que A, B e C sejam variáveis reais com valores e que I, J e K sejam variáveis inteiras. Dados A = 2.0, B = 3.0 e I = 3, indique o valor final dos comandos seguintes:
 - $C \leftarrow A * B - I$ C = _____
 - $J \leftarrow I / 4 * 6$ J = _____
 - $C \leftarrow B / A + 2.5$ C = _____
 - $K \leftarrow I / 2 + 4.7$ K = _____
 - $J \leftarrow I / A + B$ J = _____

131

Solução

- Assuma que A, B e C sejam variáveis reais com valores e que I, J e K sejam variáveis inteiras. Dados A = 2.0, B = 3.0 e I = 3, indique o valor final dos comandos seguintes:
 - $C \leftarrow A * B - I$ C = 3.0
 - $J \leftarrow I / 4 * 6$ J = 0
 - $C \leftarrow B / A + 2.5$ C = 4.0
 - $K \leftarrow I / 2 + 4.7$ K = 5
 - $J \leftarrow I / A + B$ J = 4

132

Exercício

- Dada a equação algébrica $y = ax^3 + 7$, quais dos seguintes comandos representam corretamente a equação?
 - $y \leftarrow a * x * x * x + 7$
 - $y \leftarrow a * x * x * (x + 7)$
 - $y \leftarrow (a * x) * x * x + 7$
 - $y \leftarrow (a * x) * x * (x + 7)$
 - $y \leftarrow a * (x * x * x) + 7$
 - $y \leftarrow a * x * (x * x + 7)$

133

Solução

□ Dada a equação algébrica $y = ax^3 + 7$, quais dos seguintes comandos representam corretamente a equação?

- a) $y \leftarrow a * x * x * x + 7$
- b) $y \leftarrow a * x * x * (x + 7)$
- c) $y \leftarrow (a * x) * x * x + 7$
- d) $y \leftarrow (a * x) * x * (x + 7)$
- e) $y \leftarrow a * (x * x * x) + 7$
- f) $y \leftarrow a * x * (x * x + 7)$

Resposta: (a), (c), (e)

134

Entrada e Saída

□ *Leia(lista de variáveis)*

- Permite ler valores, atribuindo-os à variáveis indicadas
- A entrada pode vir do teclado, de um arquivo, do *scanner* de código de barras, etc

□ *Escreva(lista de saída)*

- Permite mostrar os valores da lista de saída
- A saída pode aparecer em terminais de vídeo, ser impressa em papel, armazenada em arquivos, etc

135

Entrada e Saída

□ *Leia(lista de variáveis)*

- A lista de entrada fornece os nomes das variáveis às quais os valores lidos devem ser atribuídos na mesma ordem em que são encontrados no fluxo de entrada
- Ex: *Leia(A,B,C)*
Os próximos três valores lidos serão atribuídos às variáveis A, B e C: o primeiro valor para A, o segundo valor para B e o terceiro valor para C

136

Entrada e Saída

□ *Escreva(lista de saída)*

- A lista de saída fornece as expressões (incluídas variáveis e constantes) que devem ser impressas
- Ex:
 $A \leftarrow 2$
 $B \leftarrow 3$
*Escreva(A," multiplicado por ",B," = ",A*B)*
- Saída impressa:
2 multiplicado por 3 = 6

137

Entrada e Saída em C++

□ `#include <iostream>`

□ `using namespace std;`

□ *Leia(A,B,C)*

- `cin >> A >> B >> C;`

□ *Escreva(A,B,C)*

- `cout << A << B << C;`
- `cout << A << B << C << endl;`

□ São equivalentes:

- `cout << A << "\n";`
- `cout << A << endl;`

139

Exemplo 5

```
#include <iostream>
using namespace std;
```

```
int main()
{ int a;

  cout << "Entre um valor: ";
  cin >> a;
  cout << "Valor digitado = " << a << endl;
  return 0;
}
```

140

Exemplo 6

```
#include <iostream>
using namespace std;

int main()
{ int a,b;

  cout << "Entre dois valores: ";
  cin >> a >> b;
  cout << "Valores digitados = "
        << a << " e " << b << endl;
  return 0;
}
```

141

Exemplo 7

```
#include <iostream>
using namespace std;

int main()
{ int a,b;

  cout << "Entre dois valores: ";
  cin >> a >> b;
  cout << "O dobro de " << a << " = " << 2*a << endl;
  cout << "O triplo de " << b << " = " << 3*b << endl;
  return 0;
}
```

142

Comentários

- Comentários são textos que podem ser inseridos nos programas com o objetivo de documentá-los
- Os comentários não são analisados pelo compilador, ou seja, todo comentário é ignorado pelo compilador, não fazendo parte do código executável

143

Comentários em C/C++

- Os comentários podem ocupar uma ou várias linhas
- Para delimitar comentários de várias linhas, os símbolos `/*` e `*/` são utilizados
- Para delimitar comentários de uma única linha, o símbolo `//` é utilizado e encerra automaticamente no final da linha

144

Exercício em C++

- Indique, quando aplicável, o que cada um dos comandos imprime. Se nada é impresso, então responda "nada". Assuma `x = 2` e `y = 3`
- `cout << x;`
 - `cout << x + x;`
 - `cout << "x=";`
 - `cout << "x = " << x;`
 - `cout << x+y << " = " << y + x;`
 - `z = x + y;`
 - `cout << "";`
 - `/* cout << "x + y = " << x + y; */`
 - `cout << "\n";`
 - `cout << "\n**\n**\n***\n***\n****\n****\n";`

147

Solução em C++

- Indique, quando aplicável, o que cada um dos comandos imprime. Se nada é impresso, então responda "nada". Assuma `x = 2` e `y = 3`
- `cout << x;` 2
 - `cout << x + x;` 4
 - `cout << "x=";` x =
 - `cout << "x = " << x;` x = 2
 - `cout << x + y << " = " << y + x;` 5 = 5
 - `z = x + y;` (nada)
 - `cout << "";` (string vazia)
 - `/* cout << "x + y = " << x + y; */` (nada)
 - `cout << "\n";` (pula 1 linha)
 - `cout << "\n**\n**\n***\n***\n****\n****\n";` **
**

148

Exercício

1. Elabore um algoritmo que leia uma temperatura na escala Celsius ($^{\circ}\text{C}$) e imprima a equivalente em Fahrenheit ($^{\circ}\text{F}$). A fórmula de conversão é $^{\circ}\text{F} = \frac{9}{5}^{\circ}\text{C} + 32$.
2. As raízes de uma equação quadrática da forma $ax^2+bx+c=0$ são reais se e somente se o discriminante dado por b^2-4ac for maior ou igual a zero. Preparar um algoritmo para ler os valores dos coeficientes **a**, **b** e **c** e imprimir o valor do discriminante.

149

Solução Exercício 1

Algoritmo Conversão. Este algoritmo lê uma temperatura na escala Celsius ($^{\circ}\text{C}$) e imprime a equivalente em Fahrenheit ($^{\circ}\text{F}$).

```
Início
declare C,F : real

Escreva("Temperatura em graus Celsius?")
Leia(C)
F ← 9.0 / 5.0 * C + 32
Escreva("Temperatura em graus Fahrenheit = ",F)
Fim
```

150

Solução Exercício 1 em C++

```
#include <iostream>
using namespace std;
/* Algoritmo Conversão. Este algoritmo lê uma
temperatura na escala Celsius ( $^{\circ}\text{C}$ ) e imprime a
equivalente em Fahrenheit ( $^{\circ}\text{F}$ ).
*/
int main()
{ float C, F;

cout << "Temperatura em graus Celsius? ";
cin >> C;
F = 9.0 / 5.0 * C + 32;
cout << "\nTemperatura em graus Fahrenheit = "
<< F << endl;
return 0;
}
```

152

Solução Exercício 2

Algoritmo Discriminante. Este algoritmo lê os coeficientes da equação quadrática da forma $a*x^2+b*x+c=0$, calcula e imprime o valor do discriminante dado por $b^2-4*a*c$.

```
Início
declare a,b,c,delta : real

Escreva("Coeficientes a,b,c da equação ax2+bx+c=0?")
Leia(a,b,c)
delta ← b^2 - 4 * a * c
Escreva("Discriminante = ",delta)
Fim
```

153

Solução Exercício 2 em C++

```
#include <iostream>
using namespace std;
/* Algoritmo Discriminante. Este algoritmo lê os
coeficientes da equação quadrática da forma
a*x^2+b*x+c=0, calcula e imprime o valor do
discriminante dado por b^2-4*a*c.
*/
int main()
{ float a,b,c,delta;

cout << "Coeficientes a,b,c da equação ax2+bx+c=0?";
cin >> a >> b >> c;
delta = b * b - 4 * a * c;
cout << "\nDiscriminante = " << delta << endl;
return 0;
}
```

155

Resumo

- Nesta aula foram vistos alguns conceitos básicos sobre programação de computadores: tipos de dados, expressões, comando de atribuição, entrada e saída de dados
- Esses conceitos permitem escrever programas simples
- Programas mais complexos requerem estruturas de controle mais complexas, que serão vistas nas próximas aulas

156