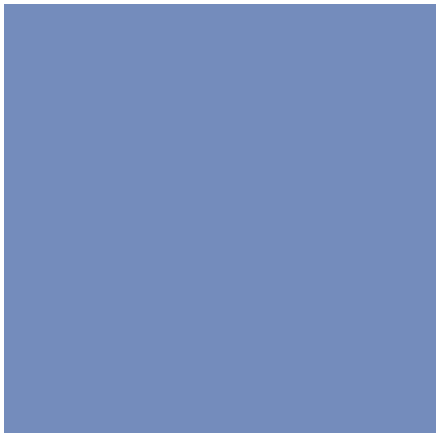
Algoritmos
Programação
Sintaxe
Lógica



INF110 – Programação I

Prof. Alcione/André Gustavo
DPI/UFV – 2020/1

UFV

Componentes do computador

- Hardware
  - Equipamentos, dispositivos, …
- Software
  - Instruções, programas, aplicativos, ...
- Realizam 4 operações principais
  - Entrada: teclado, mouse, ...
  - Processamento: CPU (unidade central de processamento), ...
  - Saída: monitor, impressora, ...
  - Armazenamento: memória RAM, disco rígido, pen-drive, ...

Algoritmos

- Sequência de ações executáveis
- Exemplos
  - Receita de bolo
  - Instruções de montagem de um equipamento
  - Instruções de uso de um medicamento
  - Instruções para configuração de um dispositivo
  - Instruções para chegar a algum local

Algoritmos

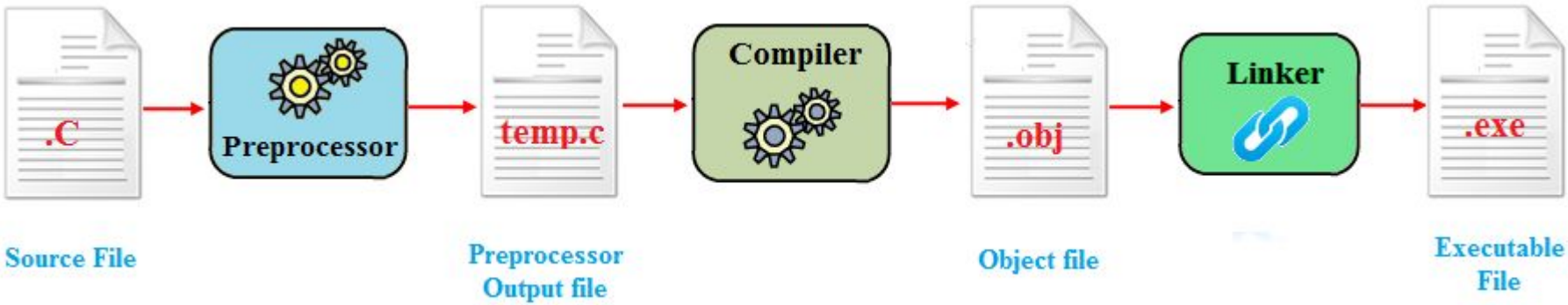
- Sequência finita de instruções bem definidas e não ambíguas

Programas

- São algoritmos que podem ser executados por computadores
- São implementações de algoritmos

Linguagens de Programação

- Instruções para o computador são escritas em uma linguagem de programação
- Computadores operam com uma “linguagem de máquina” – 0’s e 1’s que controlam chaves liga/desliga em circuitos
- Compiladores (e interpretadores) convertem de linguagem de programação para linguagem de máquina



Linguagens de Programação

- Instruções para o computador são escritas em uma linguagem de programação
- Computadores operam com uma “linguagem de máquina” – 0’s e 1’s que controlam chaves liga/desliga em circuitos
- Compiladores (e interpretadores) convertem de linguagem de programação para linguagem de máquina

Linguagens de Programação

- O código gerado por um compilador é específico para uma arquitetura de máquina e sistema operacional.
- Código interpretado ou gerado para uma máquina virtual é mais independente de máquina, mas executa mais lentamente.

Linguagens de Programação

- Um programa em linguagem de máquina de um computador específico que permite que um usuário insira dois números, adicione os dois números e exiba o total pode ser parecido como algo do tipo:

00000	10011110
00001	11110100
00010	10011110
00011	11010100
00100	10111111
00101	00000000

Linguagens de Programação

- Semelhantes às linguagens naturais (português, espanhol, etc)
- Descuidos causam problemas no entendimento
- **Sintaxe** da linguagem
  - São as regras da linguagem
  - Regulamentam uso de palavras e pontuação

Importância da Sintaxe

- Qual a diferença?
  - “Agora não quero sair.”
  - “Agora não, quero sair.”
- Qual a diferença?
  - Ele disse agora: “não quero sair.”
  - Ele disse: “agora não quero sair.”
- Qual a diferença?
  - Eles ouviram os gritos.
  - Eles ouvirão os gritos.

Importância da Sintaxe

- Seres humanos até poderiam inferir o sentido correto, pelo contexto
- Computadores não
- Em programação, a sintaxe é rígida

Importância da Sintaxe

- O que você entende?
  - “Como chego eu aeroporto no?”
- Mais uma vez, seres humanos entenderiam
- Mas em programação, a sintaxe é rígida

Importância da lógica

- Receita de bolo
  - Quebre dois ovos
  - Mexa
  - Adicione um galão de gasolina
  - Asse a 350º por 45 minutos
  - Coloque três xícaras de farinha
- Erros vão além da sintaxe!
- Instruções fora de ordem, faltantes, ou que não fazem sentido no programa, mesmo que escritas corretamente (sem erros de sintaxe), podem gerar erros de lógica.

Importância da lógica

- A esposa de um programador diz:
- “João, vá à padaria e traga 2 pães. Se tiver ovos, traga 6.”
- Ele vai, e volta com 6 pães.
- “Por que trouxe 6?”, pergunta a esposa.
- Tinha ovos.

- Erros de lógica são muito mais difíceis de localizar que de sintaxe
- É mais fácil verificar a grafia de “pães” que dizer se há em excesso

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

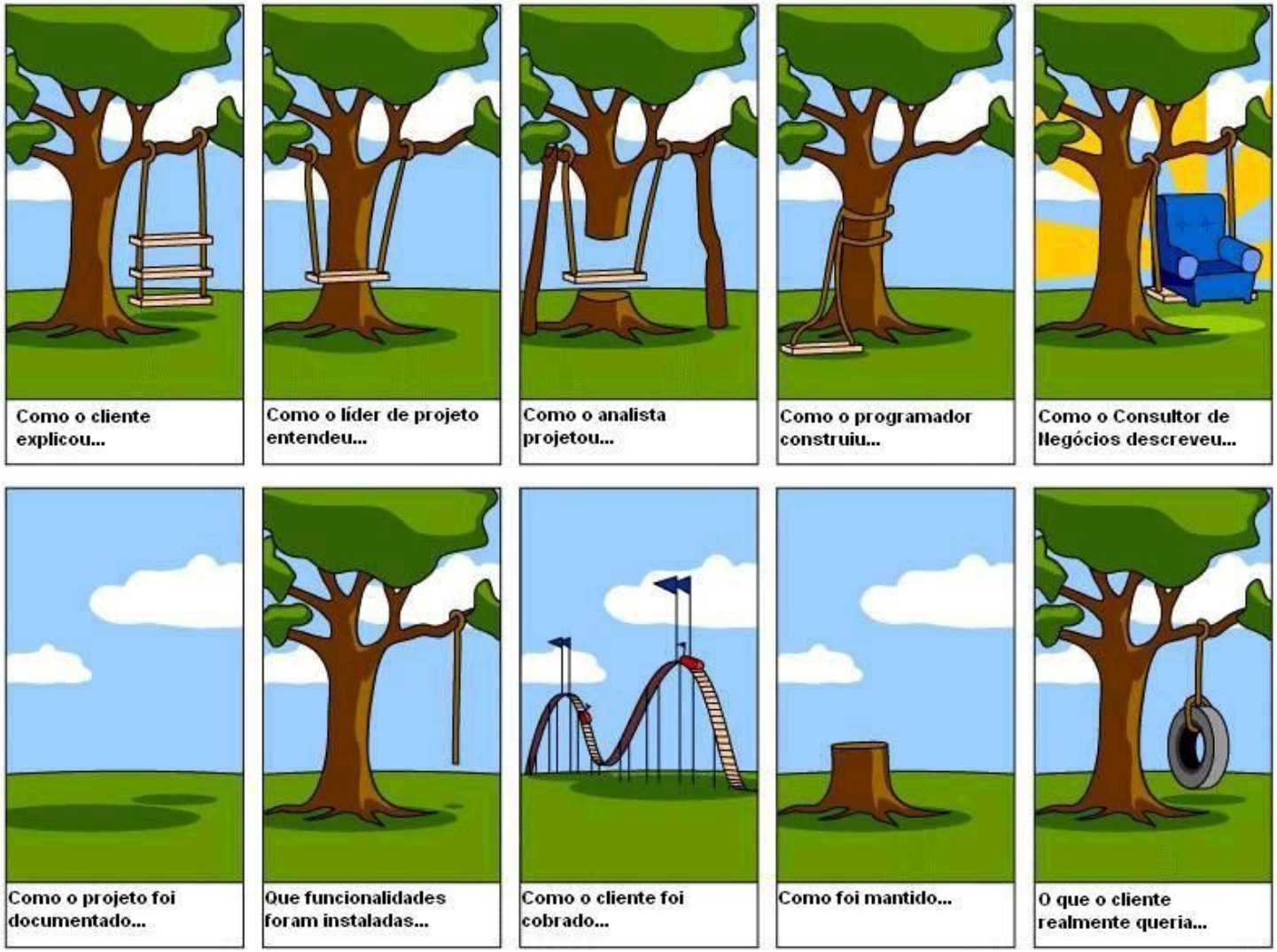
Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Entender o problema

- Na maioria das vezes, os programas são feitos a pedido de terceiros (os usuários). Então é preciso entender o que eles desejam.
- “Faça uma lista de todos os alunos da Computação nos últimos 5 anos.”
  - Devo incluir os que ingressaram neste ano?
  - Devo incluir os que se formaram há 5 anos?
  - A lista deve conter algo além dos nomes? Matrícula? Endereço?
- “Fiz 3 provas, quero um programa para calcular a nota final.”
  - A nota final é a soma? A média?

Entender o problema



WEBLOG.CEMPER.COM – Typical Project Life / Posted on 09-09-2003 (Internet Archive)

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Planejar a lógica

- É uma parte fundamental!
- Consiste em planejar os passos do programa: decidir quais são incluídos, e em que ordem
- Existem várias maneiras; as duas mais comuns são o uso de diagramas e pseudocódigo (pseudolinguagem)
- Ambas descrevem os passos numa linguagem mais natural
- Os diagramas usam ainda uma linguagem “visual”

Planejar a lógica – pseudocódigo

LEIA notaProva1

LEIA notaProva2

LEIA notaProva3

notaFinal = notaProva1 + notaProva2 + notaProva3

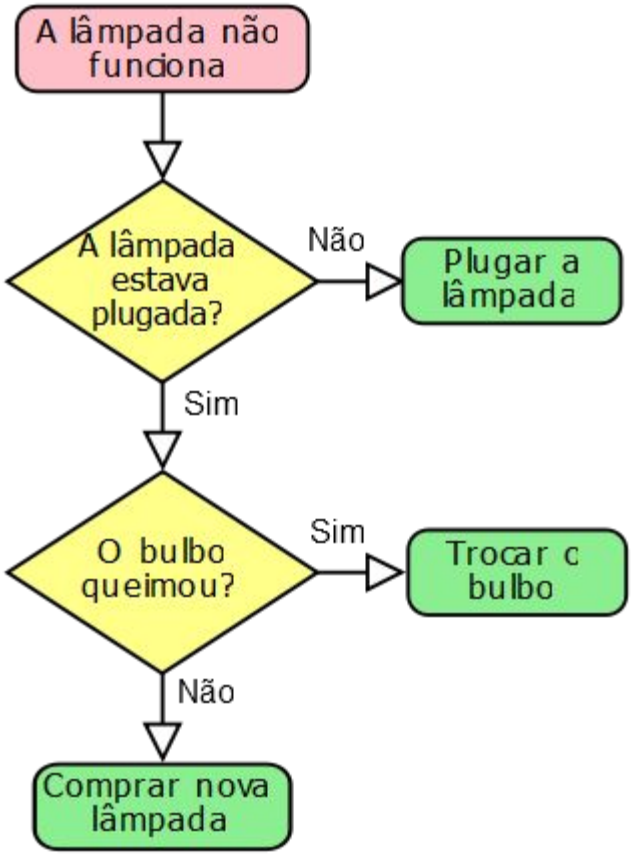
Se notaFinal >= 60

    ESCREVA “Aprovado”

Se notaFinal < 60

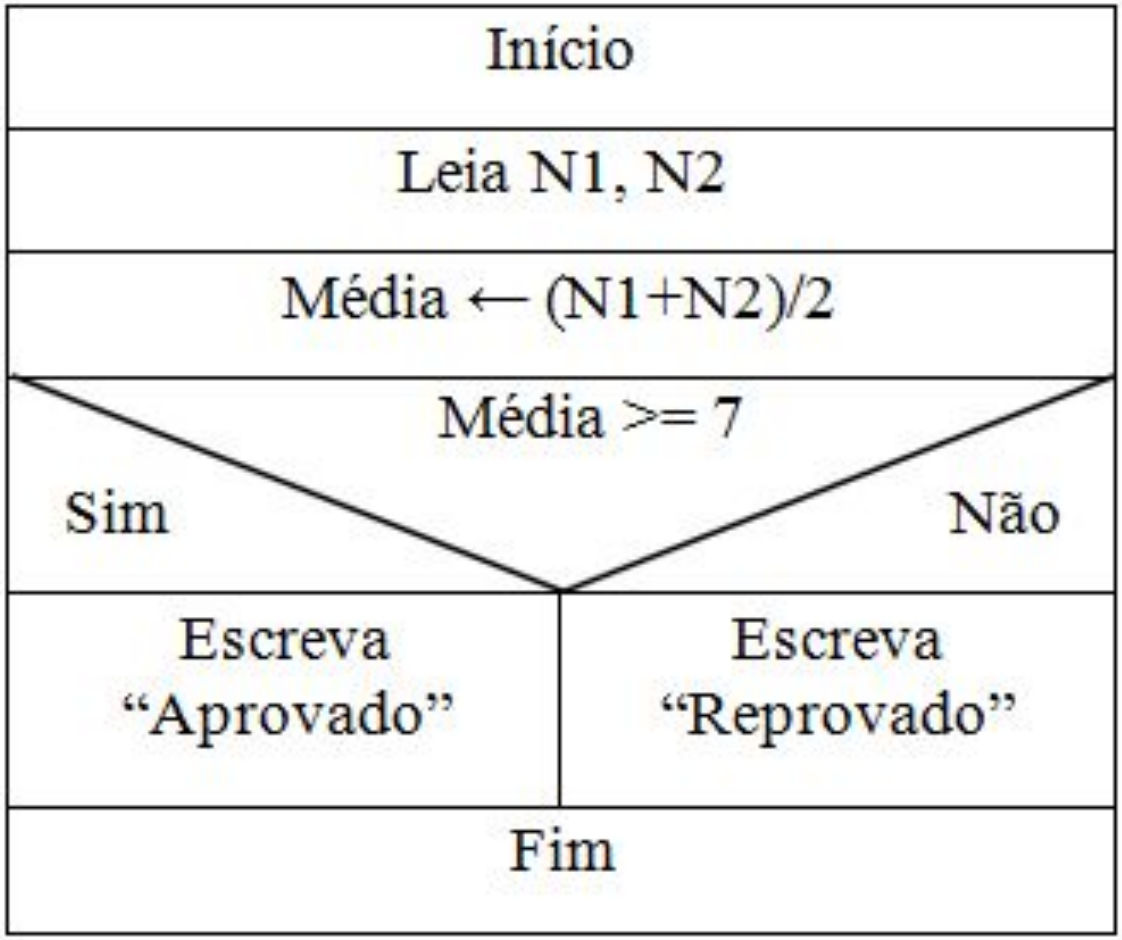
    ESCREVA “Reprovado”

Planejar a lógica – Diagrama Fluxograma



Fonte: wikipedia

Planejar a lógica – Diagrama de Chapin ou de Nassi-Shneiderman



Fonte: https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Codificar o programa

- Depois de desenvolvida a lógica do programa, pode-se escrever o programa em alguma linguagem de programação
- O programador escolhe a linguagem dependendo do contexto, pois algumas são mais eficientes para certos tipos de operações
- A lógica desenvolvida para solucionar o programa pode ser executada em praticamente qualquer tipo de linguagem
- Só depois de escolhida a linguagem, preocupa-se com a grafia correta de cada comando (sintaxe)

Obs: programadores mais experientes conseguem combinar o planejamento da lógica com a codificação do programa

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Compilar o programa

- Traduzir o programa em linguagem de alto nível para linguagem de baixo nível
  - Linguagem de alto nível: mais próxima à linguagem humana
  - Linguagem de baixo nível: mais próxima à linguagem de máquina
- Erros de sintaxe
  - Erro “ortográfico”, comandos inexistentes, gramática “ilegal”
- O compilador só traduz quando não há erros de sintaxe
- Pode emitir uma mensagem informando o erro encontrado
- Algo semelhante a um tradutor português-inglês com a frase:
  - “a meinina vão para a escola”

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Testar o programa

- Programas sem erro de sintaxe não estão livres de erros!
- Erros de lógica (erros de semântica)
  - Sintaticamente correto, mas não chega ao resultado esperado
- Programas devem ser testados com muitas amostras de dados, para testar seu funcionamento em muitas situações

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Colocar o programa em produção

- Depende do contexto
  - Pode ser executá-lo uma vez
  - Pode ser incluí-lo como parte de outro sistema
  - Pode envolver treinamento de pessoal
  - Pode exigir conversão de dados

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção

Programar

1. Entender o problema
2. Planejar a lógica
3. Codificar o programa
4. Compilar o programa
5. Testar o programa
6. Colocar o programa em produção
7. Manutenção

Manutenção

- Realização de alterações (atualizações) no programa
  - Adequá-lo a uma nova legislação
  - Alterar o formato de entrada ou de saída
  - Incluir informações adicionais
  - Corrigir erros que não foram pensados no planejamento da lógica ou codificados incorretamente e não vistos na fase de teste

- Ex.:
  - Número de matrícula da UFV (estamos chegando no 6º digito!)
  - “Bug” do milênio (link)
  - “Bug” Boeing 737 MAX (link)

Continua em:

Algoritmos, Programação, Sintaxe e Lógica