Introdução ao processo de desenvolvimento de programas

Alexsandro Santos Soares
prof.asoares@gmail.com

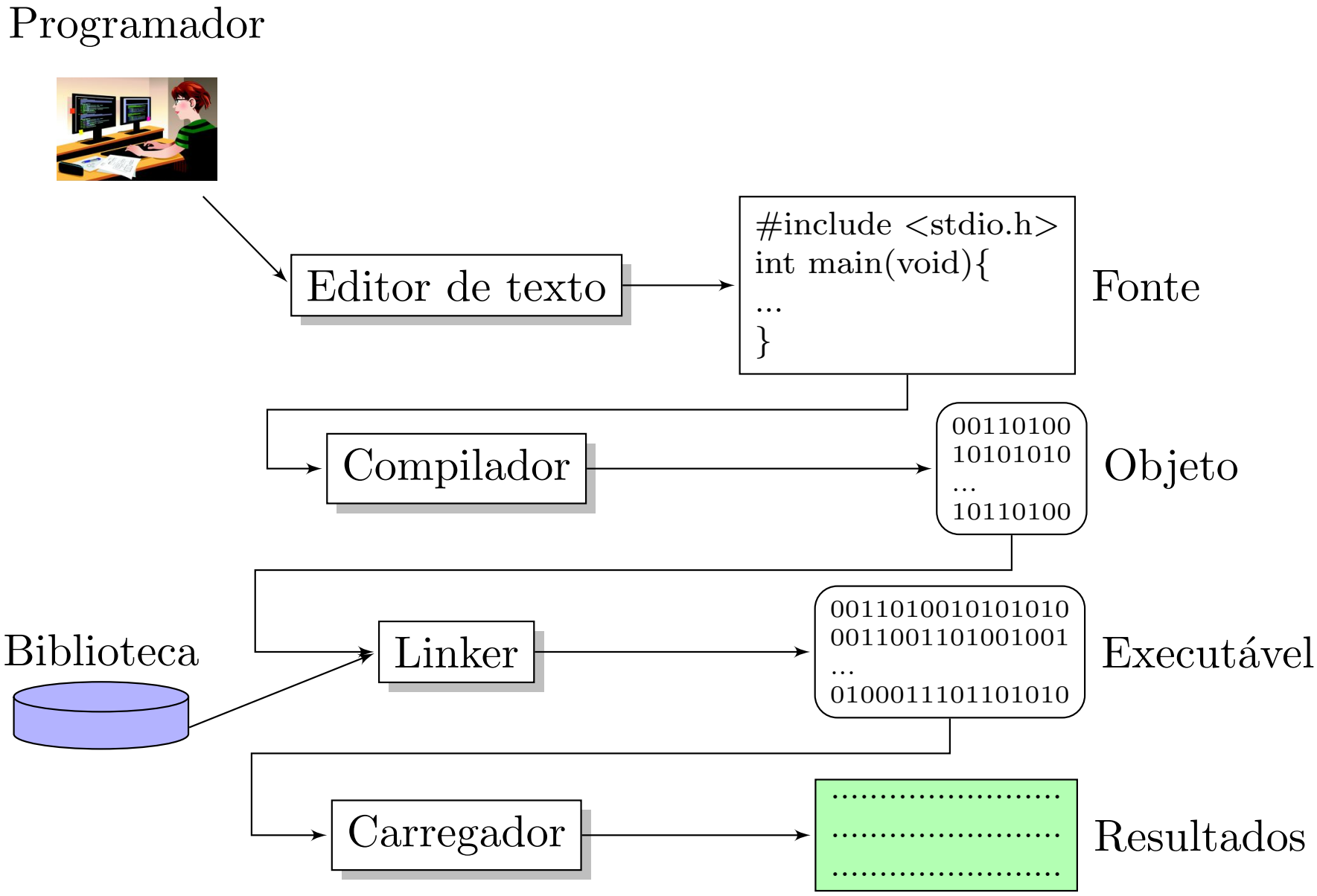
Universidade Federal de Uberlândia
Faculdade de Computação

Nada pode parar a automação

Criando e executando programas

- Sabemos que um computador somente compreende um programa se ele for codificado em sua linguagem de máquina.
- A tarefa do programador é escrever e testar o programa sob sua responsabilidade. Para isto deve-se:
  - Escrever e editar o programa.
  - Compilá-lo.
  - Ligar o programa aos módulos de biblioteca.
  - Executá-lo.

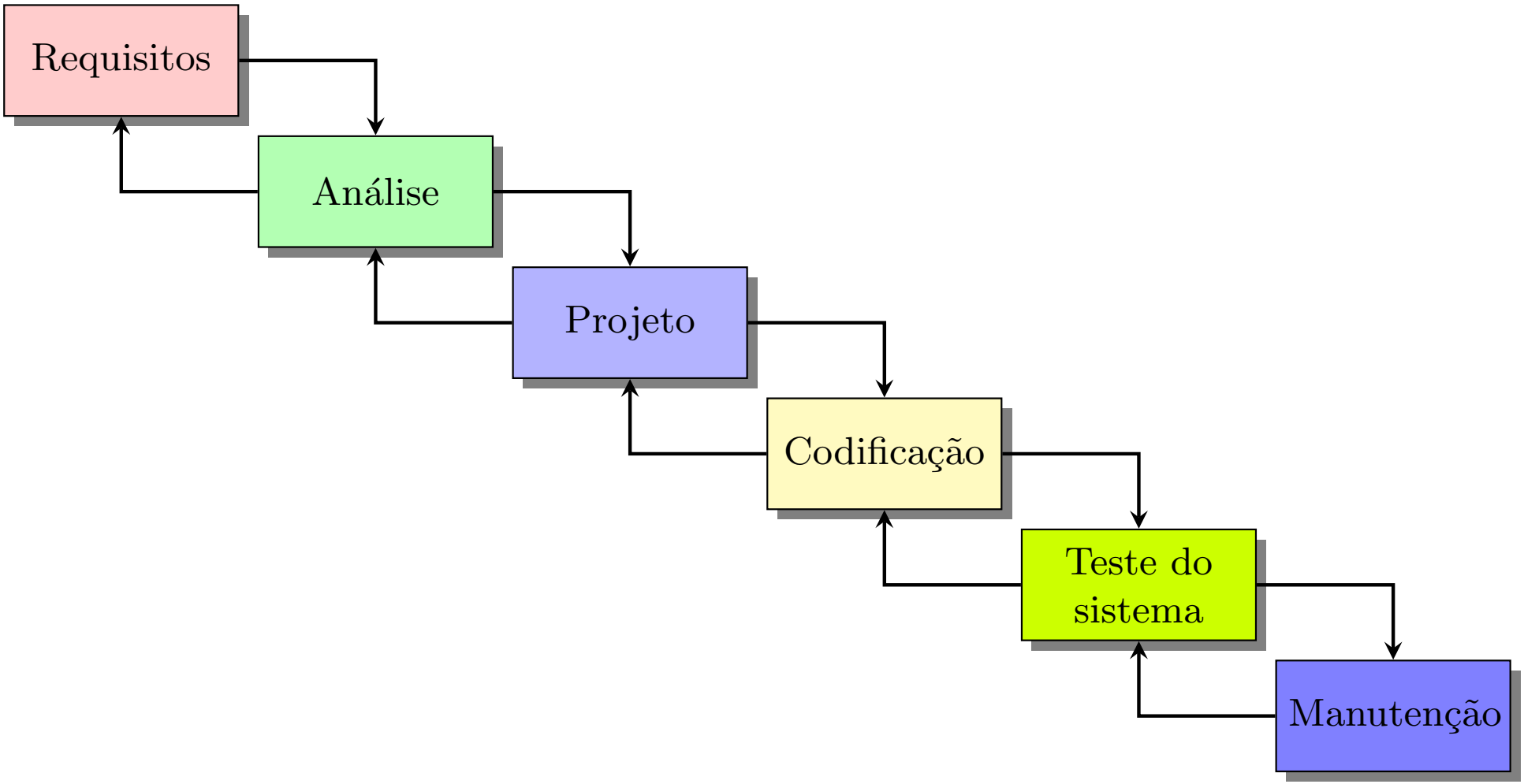
Processo de compilação



Desenvolvimento de sistemas

- Os projetos de programação modernos e de grande escala são construídos usando uma série de fases chamada de ciclo de vida do desenvolvimento do sistema.
- Um ciclo de vida muito popular é o modelo cascata.
  - Dependendo da empresa e do tipo de software sendo desenvolvido, este modelo pode consistir de cinco a sete fases.
- No próximo slide é apresentado uma variante possível deste modelo.

Modelo cascata



Requisitos do sistema

- Estabelecimento dos requisitos do produto que, normalmente, consistem nos serviços ofertados, limitações e objetivos do programa.
  - Os requisitos são geralmente enunciados em uma linguagem próxima a do cliente.
- Esta etapa inclui também a documentação e o estudo da facilidade e da viabilidade do projeto.

Análise do sistema

- Nesta fase, a coleta de requisitos é intensificada.
- Busca-se compreender melhor o domínio da informação, a função, as interfaces e o desempenho exigidos.
- Proposição de alternativas diferentes do ponto de vista do sistema.
- Os requisitos, tanto do sistema quanto do software, são documentados e revistos com o cliente.

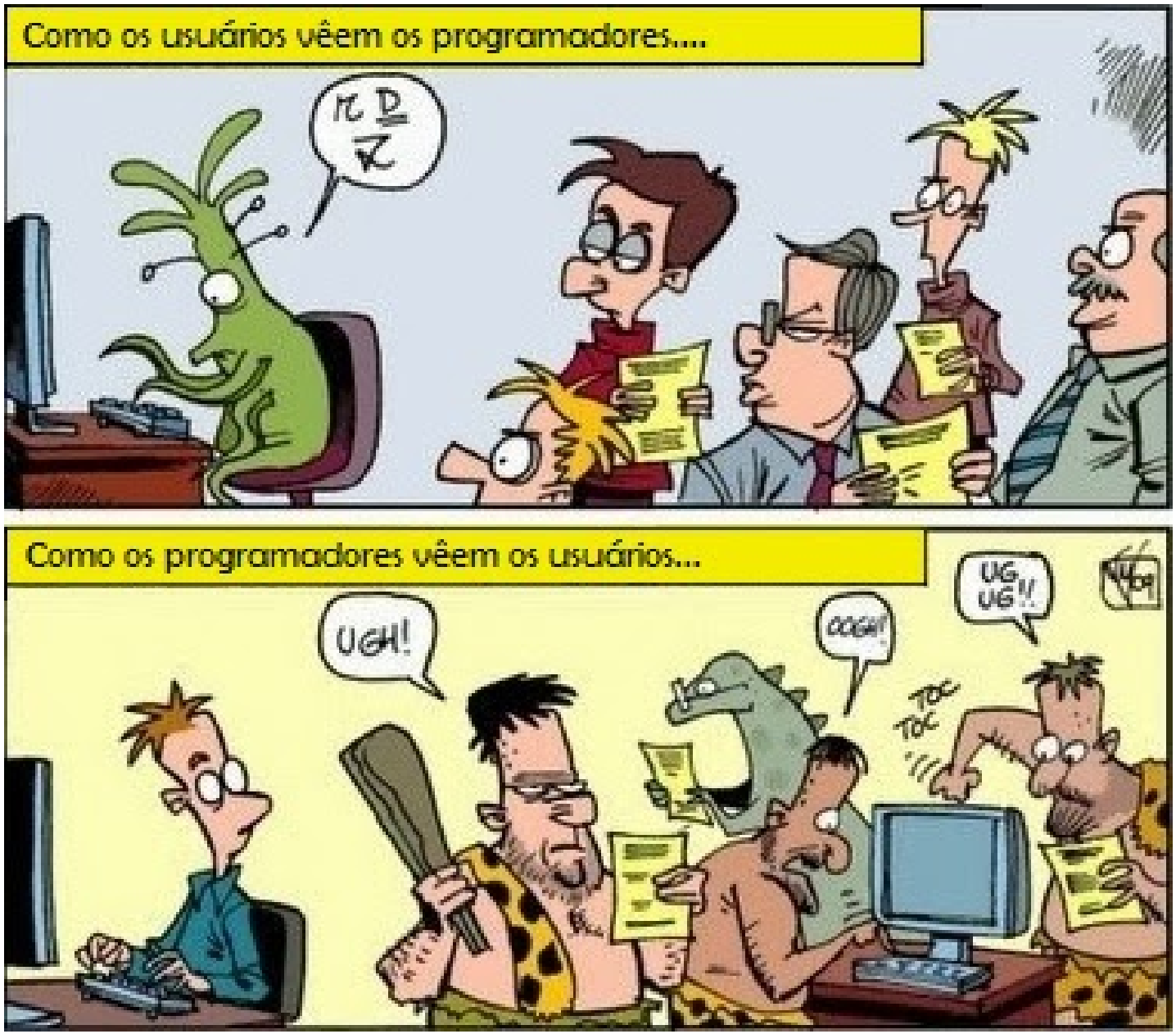
Projeto do sistema

- Visa determinar como o sistema será construído:
  - Estrutura de dados.
  - Arquitetura do software.
  - Detalhes procedurais.
  - Caracterização das interfaces.
- Os requisitos são representados de uma forma que permita a codificação do produto.
  - As representações podem ser avaliadas quanto à qualidade, antes do início da codificação.
  - O projeto é documentado e transforma-se em uma parte do software.

Codificação

- Nesta fase são criados os programas em alguma linguagem de programação.
  - Esta é a fase que nos ocuparemos nesta disciplina.
- Se o projeto for bem detalhado, a codificação será quase automática.
- Pode incluir testes unitários dos módulos.
  - As unidades de código produzidas são testadas individualmente, antes de passar para a etapa de integração e teste global.

Codificação



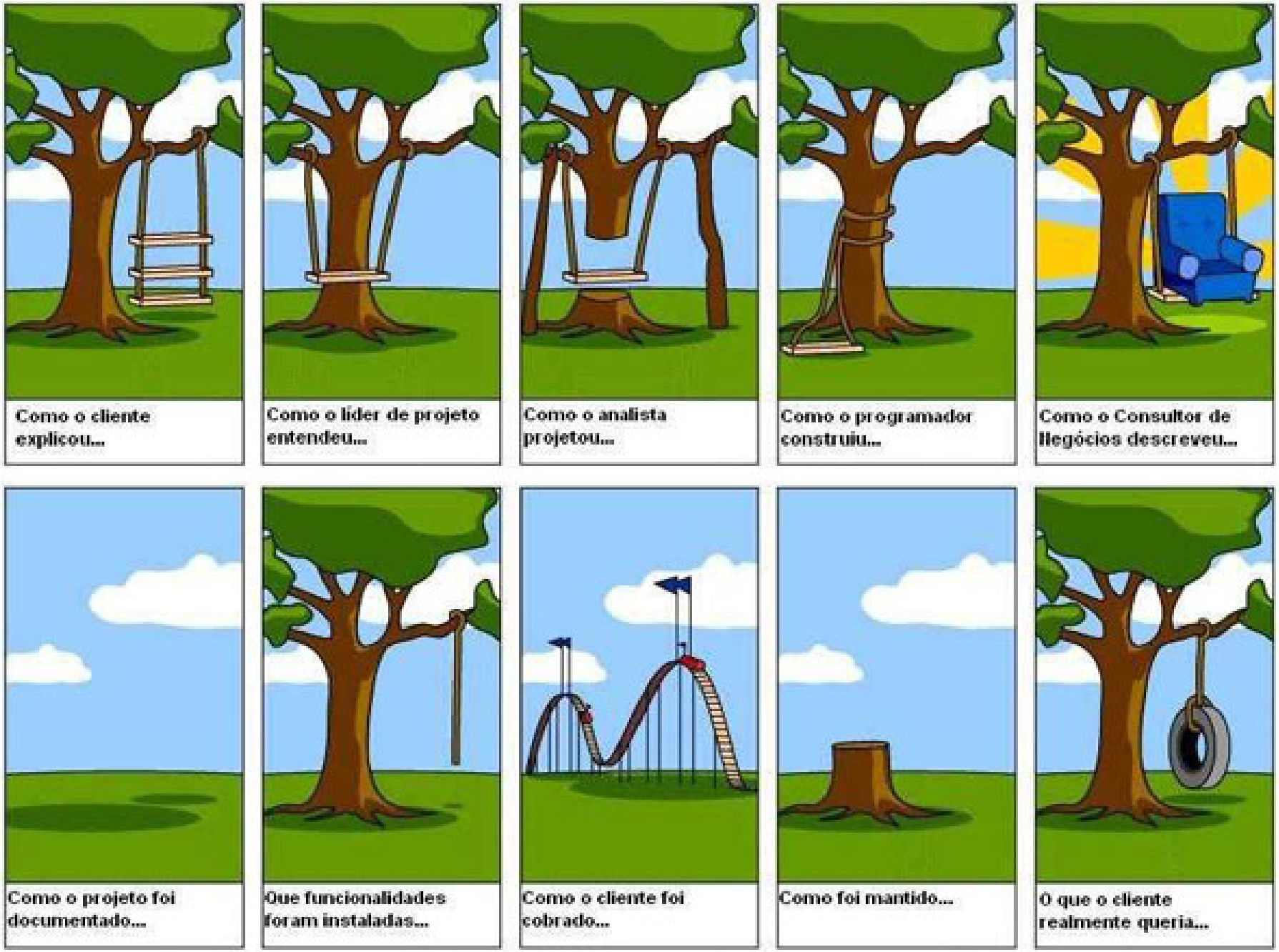
Teste do sistema

- Todos os programas criados são testados conjutamente para se assegurar que o sistema funciona como um todo.
- O processo de teste centraliza-se em dois pontos principais:
  - Aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas.
  - Aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados esperados e coincidentes com os requisitos especificados.

Manutenção

- O software poderá sofrer mudanças após ser entregue ao cliente.
  - Erros que não foram previamente detectados.
  - Melhorias funcionais e de preferência.
  - Adaptação para acomodar mudanças em seu ambiente externo.
- Esta é a etapa na qual se deve manter o software funcionando após entrar em produção.

Fases de um projeto



Desenvolvimento de programas

- Ao sermos incumbidos de desenvolver um programa, a nós serão dados o enunciado dos requisitos do programa e o projeto das interfaces do programa.
- Também deveríamos receber uma visão geral do projeto completo para que compreendamos como nossa parte se encaixa no todo.
- Nosso trabalho é determinar como receber as entradas dadas e convertê-las nas saídas já especificadas.
- Chamamos esse processo de projeto de programa.
- Para termos uma ideia de como o processo funciona, vamos analisar o seguinte problema:
  - Calcular a metragem quadrada de uma casa.

Uma planta baixa possível

Compreendendo o problema

- O primeiro passo para resolver qualquer problema é compreendê-lo.
- Leia atentamente o documento de requisitos e depois repasse tudo com o usuário e/ou o analista de sistemas.
  - Faça perguntas para confirmar sua compreensão.
- Para o nosso problema, poderíamos formular algumas perguntas:

- Qual é a definição de metragem quadrada?
- Como a metragem quadrada será usada?
  - Para propósito de seguro?
  - Para pintar o interior ou o exterior da casa?
  - Para acarpetar a casa inteira?
- A garagem será incluída?
- Os armários e corredores serão incluídos?

- Cada uso requer uma medição diferente. Se não soubermos o propósito exato, daremos a resposta errada.

Desenvolvendo a solução

- Após termos de fato compreendido o problema, precisamos desenvolver uma solução.
- Três ferramentas podem nos auxiliar:

  Diagrama de estrutura modular usado para modelar o programa como um todo.

  Pseudocódigo usado para projetar partes individuais do programa chamadas de módulos.

  Fluxograma possui a mesma função do pseudocódigo, mas é visual.
- Em geral, usamos somente dois deles: um diagrama de estrura com um pseudocódigo ou fluxograma.

Diagrama de estrutura modular

- Um diagrama de estrutura modular (DEM) mostra o fluxo funcional através de um programa.
- Ele mostra como quebrar o programa em passos lógicos.
  - Cada passo será um módulo separado.
- Ele também exibe a interação entre todas as partes (módulos) do programa.
  - Lembre-se que o projeto, representado aqui pelo DEM, é feito antes da escrita do programa.
  - Eles são como as plantas de um engenheiro, ou seja, um conjunto de planos construtivos.
- Um dos erros mais comuns entre os programadores iniciantes, e também com os experientes, é começar a codificar antes que o projeto esteja completo.

Tentação do código

- A pressa em começar a codificar é devida em parte a duas causas:
  1. Os programadores pensam que compreendem inteiramente o problema.
  2. Ansiedade em resolver um problema novo.
- No primeiro caso, descobre-se que não haviam compreendido de fato o problema.
  - Resista à tentação de codificar.
- O segundo caso é devido à natureza humana.
  - Programar é uma tarefa legal demais.
  - Ver um projeto tomar forma e ver o programa funcionando pela primeira vez leva a uma forma de realização pessoal.

Revisão do Diagrama de estrutura modular

- Ao se completar o DEM faz-se um painel de revisão envolvendo usuários, programadores, analistas e, às vezes, o pessoal dos testes.
- A ideia é que ao se mostrar o DEM, eles possam oferecer sugestões construtivas sobre como melhorar o projeto.
  - Isso aumenta a qualidade e pode economizar tempo.
  - Quanto mais cedo um erro for detectado, mais fácil será consertá-lo.
- Aqui, em sala, você pode rever o seu projeto com colegas ou com o professor.

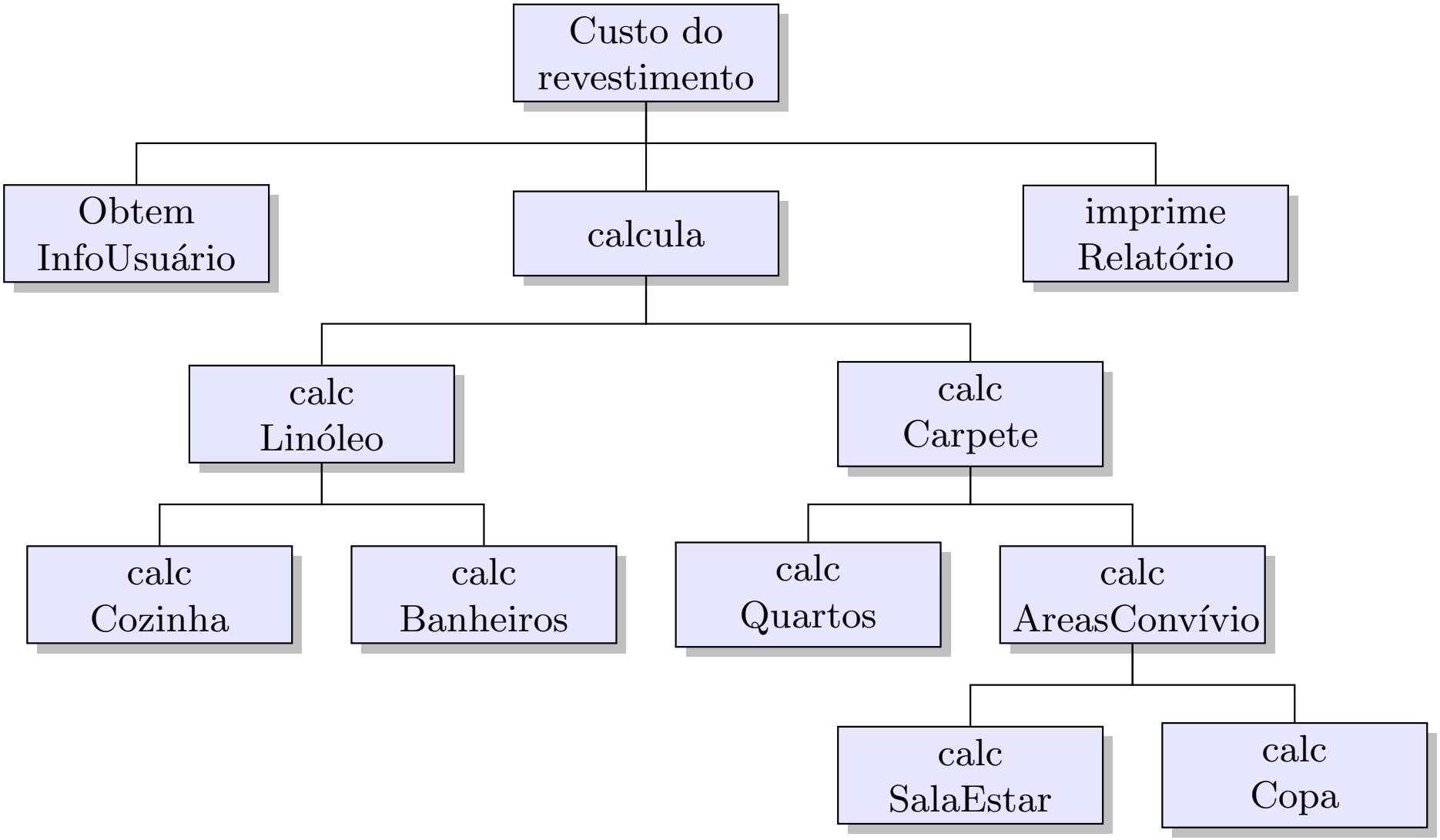
DEM para calcular a metragem quadrada

Assuma que para as perguntas feitas anteriormente sobre a metragem quadrada foram respondidas assim:

1. O objetivo de calcular a metragem quadrada é para instalação de um novo revestimento para o pavimento.
2. Somente o espaço de convivência será acarpetado. A garagem e os armários não seram considerados.
3. A cozinha e os banheiros serão revestidos com linóleo. O restante da casa será acarpetado.

- Com este entendimento, decidimos escrever módulos separados para a cozinha, banheiros, quartos e sala de estar.
  - Quantidades diferentes de linóleo e carpete serão necessárias para cada cômodo.
- Um possível DEM para esta configuração será mostrado no próximo slide.

DEM para a metragem quadrada



Dicas de projeto

- Para completar o projeto pode-se usar um fluxograma ou pseudocódigo. Isto dependerá da:
  - Experiência do programador.
  - Dificuldade do processo sendo projetado.
  - Cultura e padrões organizacionais da empresa.
- Programadores iniciantes devem primeiro aprender a projetar programas usando fluxogramas.
  - É visual e mais fácil de criar do que pseudocódigo.
  - Entretando, pseudocódigo é mais comum entre profissionais.

Pseudocódigo

- Pseudocódigo é uma mistura de português com lógica de programação. Seu objetivo é descrever, com detalhes algoritmos precisos, o quê o programa faz.
- Isso requer a definição dos passos para realizar uma tarefa com detalhamento suficiente para que seja convertido em um programa de computador.

Exemplo de pseudocódigo

O pseudocódigo abaixo determina o custo do linóleo para os banheiros:

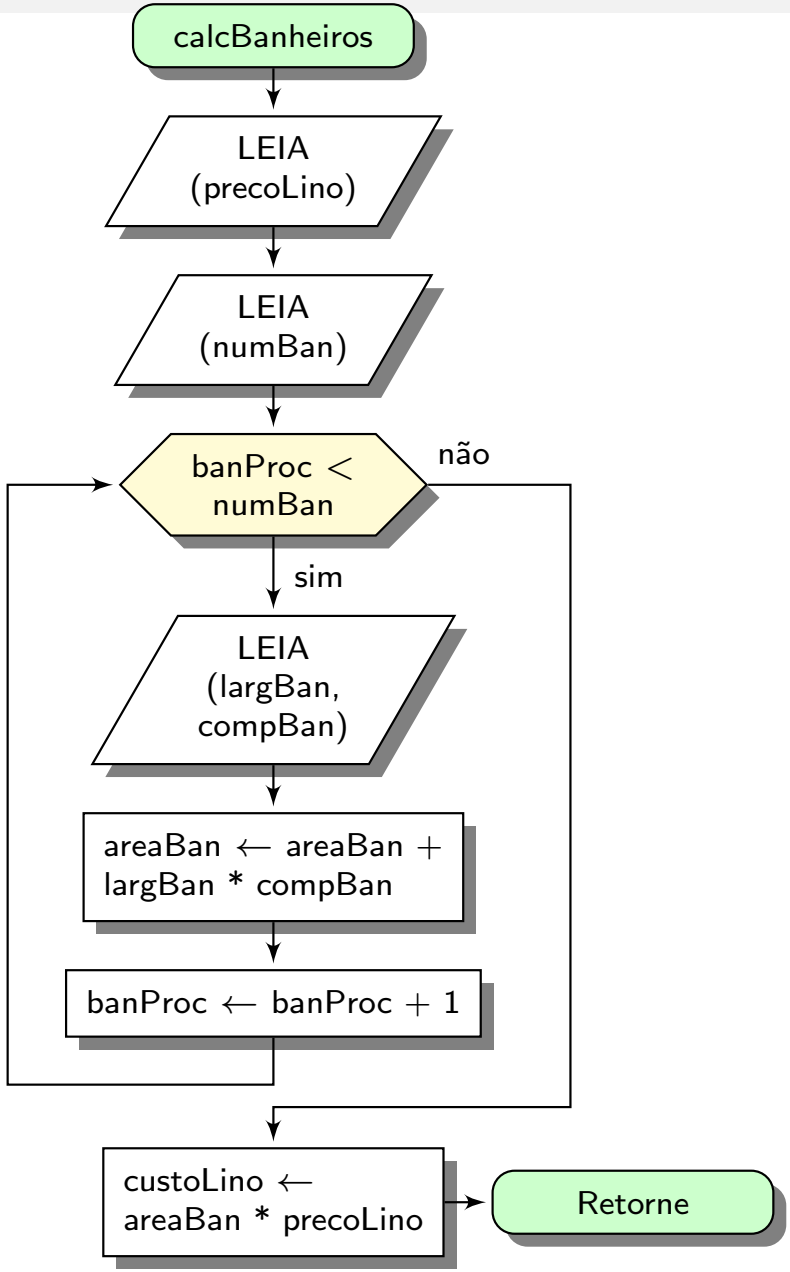
Algoritmo 1: Calcula Banheiros

```
1  avise o usuário e leia o preço do linóleo;
2  avise o usuário e leia o número de banheiros;
3  área total do banheiro ← 0;
4  banheiros processados ← 0;
5  enquanto banheiros processados < número de banheiros faça
6  |   avise o usuário e leia a largura e o comprimento do banheiro;
7  |   área total do banheiro ← área total do banheiro + largura do banheiro ×
   |   comprimento do banheiro;
8  |   adicione 1 a banheiros processados;
9  fim enqto
10 custo do banheiro ← área total do banheiro × preço do linóleo;
11 retorna custo do banheiro;
```

Fluxograma

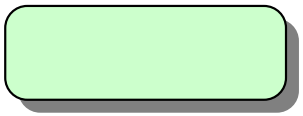
- Um fluxograma é uma ferramenta de projeto na qual símbolos gráficos são usados para representar o fluxo lógico dos dados através de uma função.
- No próximo slide será mostrado o fluxograma do custo do linóleo para os banheiros.

Exemplo de fluxograma



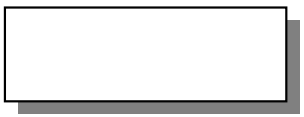
Símbolos do fluxograma

Abaixo estão os principais símbolos de um fluxograma e seus usos.

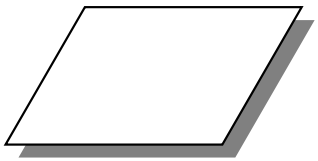


marca o início e o fim de um algoritmo

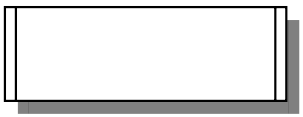
marca pontos de conexão no algoritmo



comando de atribuição

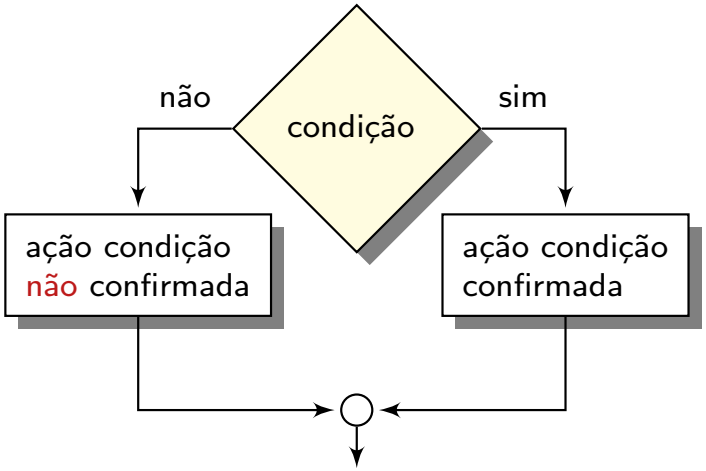


comando de entrada ou saída

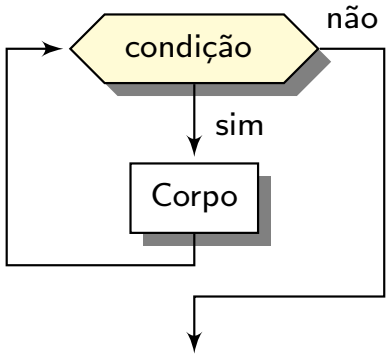


chamada de módulo ou função

Símbolos do fluxograma



comando de seleção condicional



comando de repetição

Escrita do programa

- Terminada a fase de projeto começamos a escrever o programa.
- Vamos recapitular os passos que usamos até aqui
  1. Compreender o problema.
  2. Desenvolver uma solução.
     1. Projetar o programa usando o diagrama de estrutrua modular.
     2. Projetar os algoritmos para o programa usando fluxogramas e/ou pseudocódigo.
  3. Escrever o programa.

Implementação top-down

- Ao escrever o programa, podemos iniciar no caixa no topo do DEM e continuar escrevendo os demais módulos até chegar embaixo.
- Esse modo de trabalhar é chamado de implementação de cima para baixo, *top-down* em inglês.
- Em nossos primeiros programas existirá apenas um único módulo, representado a caixa no topo do DEM.
  - Os primeiros programas serão simples e não necessitarão de divisão adicional.

Teste do programa

- Ao terminar a escrita do programa devemos testá-lo.
- O teste do programa pode ser muito tedioso e consumir bastante tempo da fase de desenvolvimento.
- Como programadores é nossa responsabilidade testar nossos programas completamente.
  - Em grandes projetos existem especialistas responsáveis pelos testes gerais do sistema.
- Existem ao menos dois tipos de testes:

  caixa-preta também chamado de teste funcional, ele é realizado pela equipe de engenharia de testes juntamente com os usuários.

  caixa-branca também chamado de teste estrutural, ele é realizado pelo programador.

Teste de caixa-preta

- O nome deste teste vem do conceito de testar o programa sem saber como ele funciona internamente.
- Eles são derivados apenas da especificação dos requisitos, sem considerar detalhes de implementação.
- A partir daí é produzido um conjunto de casos de testes.
  - Devemos sempre pedir para ver esse plano de testes antes de escrever os programas.
  - Isso nos ajudará a entender melhor os requisitos e também a criar nossos próprios conjuntos de testes.

Teste de caixa-branca

- No teste de caixa-branca asume-se que o testador compreende totalmente o código fonte do programa.
- A responsabilidade por este teste é do programador e devemos nos assegurar que quaisquer instruções e situações possíveis tenham sido testadas.
- A experiência ajudará a escrever bons conjuntos de testes, mas devemos desde o início ter o hábito de escrevê-los.
  - Iniciamos o plano de teste já na fase de projeto.
  - Ao construirmos o DEM devemos nos questionar sobre situações especiais, principalmente as não usuais, que precisaremos testar.
  - Devemos tomar nota disso imediatamente pois poderemos não lembrar depois.
- Lembre-se que, exceto para programas muito simples, um conjunto de casos de testes não validará completamente um programa.

Testes

- Como saber se o seu programa está completamente testado?
  - Não há como saber isso com certeza!
- Algumas dicas podem ajudar e aumentar a segurança:
  1. Verifique se todas as linhas de códigos foram executadas pelo menos uma vez.
  2. Verifique se cada comando condicional no programa foi executado tanto para os casos em que a condição é verdadeiro quanto para aqueles nos quais ela é falsa.
  3. Em quaisquer condições envolvendo intervalos, assegure-se que os testes incluam o primeiro e o último item do intervalo, assim como ítens que fiquem aquém do primeiro ou além do último.
  4. Se existirem condições de erro sendo verificadas, certifique-se que todos os erros lógicos sejam testados.
     - Talvez serão necessárias modificações temporárias em nossos programas para forçar os erros.

Para saber mais

- Ana Lúcia Figueredo Regino. *A engenharia de requisitos de software aplicada ao Sistema de Inclusão de Profissional ao Mercado de Trabalho - SIPMT*. Disponível em `repositorio.uniceub.br/bitstream/235/7854/1/51202763.pdf`
- Ricardo Sequeira Ferreira. *Desenvolvimento, Testes e Qualidade de Software*. Disponível em `http://recil.grupolusofona.pt/handle/10437/1299`.

Fontes

- Forouzan, B. A and Gilbert, R. F. *Computer Science: a structured programming approach using C*. 3rd edition. Cengage Learning, 2007.
- Humor / Charge em Gerenciamento de Projetos. Disponível em `https://www.projetodiario.net.br/humor-charge-em-gerenciamento-de-projetos`. Acesso em 30 mar. 2018.