CEUB

EDUCAÇÃO SUPERIOR

Linguagem de Programação

ceub.br

Professor: Rogério Alves

CEUB
EDUCAÇÃO SUPERIOR

Programação Orientada a Objetos

Conceitos de Linguagens de Programação
Aula 01 – Introdução

O que é uma Linguagem de Programação?

- **Na programação de computadores, uma linguagem de programação serve como meio de comunicação entre o indivíduo que deseja resolver um determinado problema e o computador.**

- **A linguagem de programação deve fazer a ligação entre o pensamento humano (muitas vezes de natureza não estruturada) e a precisão requerida para o processamento pelo computador.**

O que é uma Linguagem de Programação?

Uma linguagem de programação auxilia o programador no processo de desenvolvimento de software:

- **Análise de Requisitos;**
- **Projeto;**
- **Implementação;**
- **Teste;**
- **Verificação;**
- **Manutenção do software;**

O que é uma Linguagem de Programação?

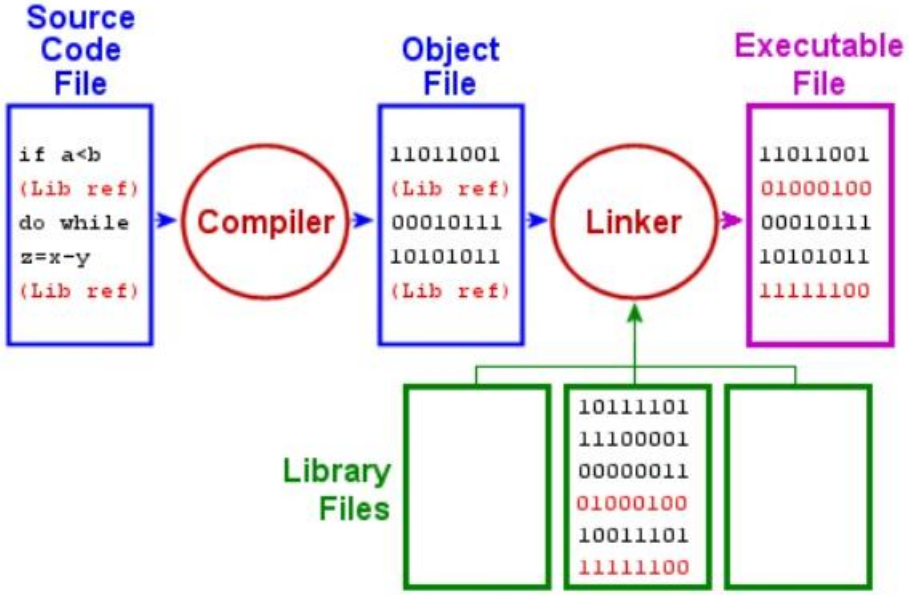
Uma linguagem de programação é uma linguagem destinada para ser usada por uma pessoa para expressar um processo através do qual um computador possa resolver um problema.

Os modelos/paradigmas de linguagens de programação correspondem a diferentes pontos de vista dos quais processos podem ser expressados.

Exemplos: Imperativo, orientado a objeto, funcional, lógico

O que é uma Linguagem de Programação?

É uma formar de interagir com as máquinas(computadores), tornando operacionais os programas escritos em linguagens de alto nível que necessitam ser traduzidos para linguagem de máquina.

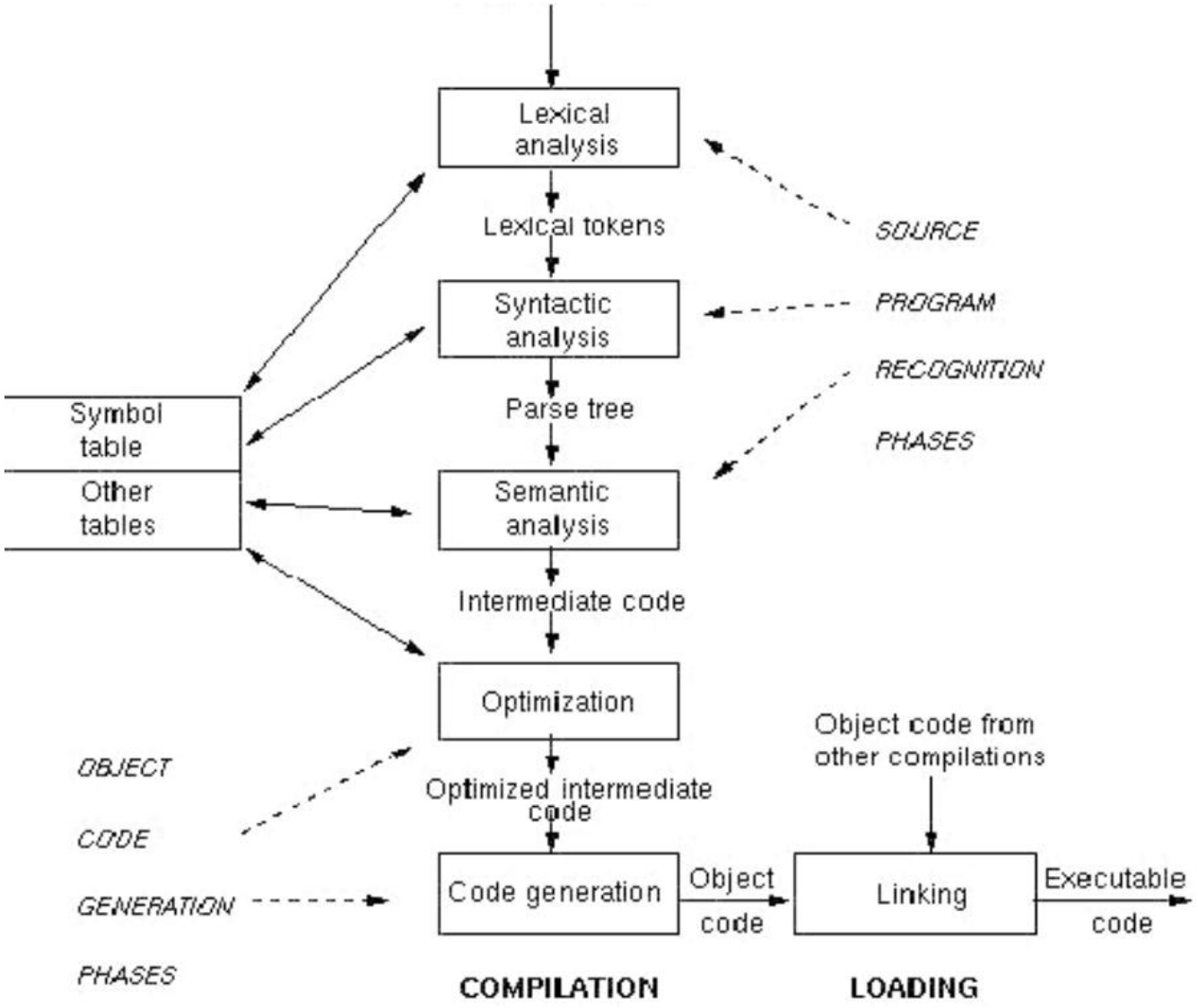


O que é uma Linguagem de Programação?

A conversão de um código em linguagem alto nível para linguagem de máquina é realizada através de sistemas especializados:

Compiladores ou Interpretadores

Esses sistemas recebem como entrada uma representação textual da solução de um problema (expresso em uma linguagem fonte) e produzem uma representação do mesmo algoritmo expresso em uma linguagem de máquina.


Lexical analysis
Lexical tokens
Syntactic analysis
Parse tree
Semantic analysis
Intermediate code
Optimization
Optimized intermediate code
Code generation
Object code
Linking
Executable code
Object code from other compilations
Symbol table
Other tables
SOURCE PROGRAM
RECOGNITION PHASES
OBJECT CODE GENERATION PHASES
COMPILATION
LOADING

Por que estudar Linguagens de Programação?

Ampliar e melhorar a capacidade de resolver problemas e expressar ideias

- **Conhecimento amplo dos recursos de linguagem reduz as limitações no desenvolvimento de software;**
- **A melhor compreensão das funções e implementação das estruturas de uma linguagem de programação nos leva a usar a linguagem de modo a extrair o máximo de sua funcionalidade e eficiência;**
- **Recursos ou facilidades podem ser simulados.**

Por que estudar Linguagens de Programação?

Ampliar o leque de conhecimento sobre linguagens de programação:

- **Algumas linguagens são mais apropriadas para resolver determinados problemas;**
- **Torna mais fácil aprender novas LPs e acompanhar a tecnologia(quanto mais LPs você souber, mais fácil se torna aprender uma nova LP).**
- **Escolher a melhor linguagem para um problema específico devido ao conhecimento de novos recursos é difícil para:**
  - ❖ **Programadores antigos;**
  - ❖ **Desenvolvedores sem educação formal;**

CEUB
EDUCAÇÃO SUPERIOR

Por que estudar Linguagens de Programação?

Entender a importância da implementação no processo de desenvolvimento de software:

- **Leva a um entendimento do porquê das linguagens serem projetadas de determinada maneira;**
- **Possibilitar o entendimento sobre quais linguagens de programação podem ser utilizadas e o resultado direto dessas escolhas**
- **Possibilita o desenvolvimento de códigos, programas mais efetivos (eficientes e eficazes), aumentando o seu vocabulário de construção úteis de programação**

Por que estudar Linguagens de Programação?

Maior capacidade para aprender novas linguagens:

- **Na computação, é fundamental que haja um aprendizado contínuo e eficiente de linguagens de programação**
- **Compreender os conceitos gerais das linguagens torna mais fácil entender como eles são incorporados na linguagem que está sendo aprendida;**

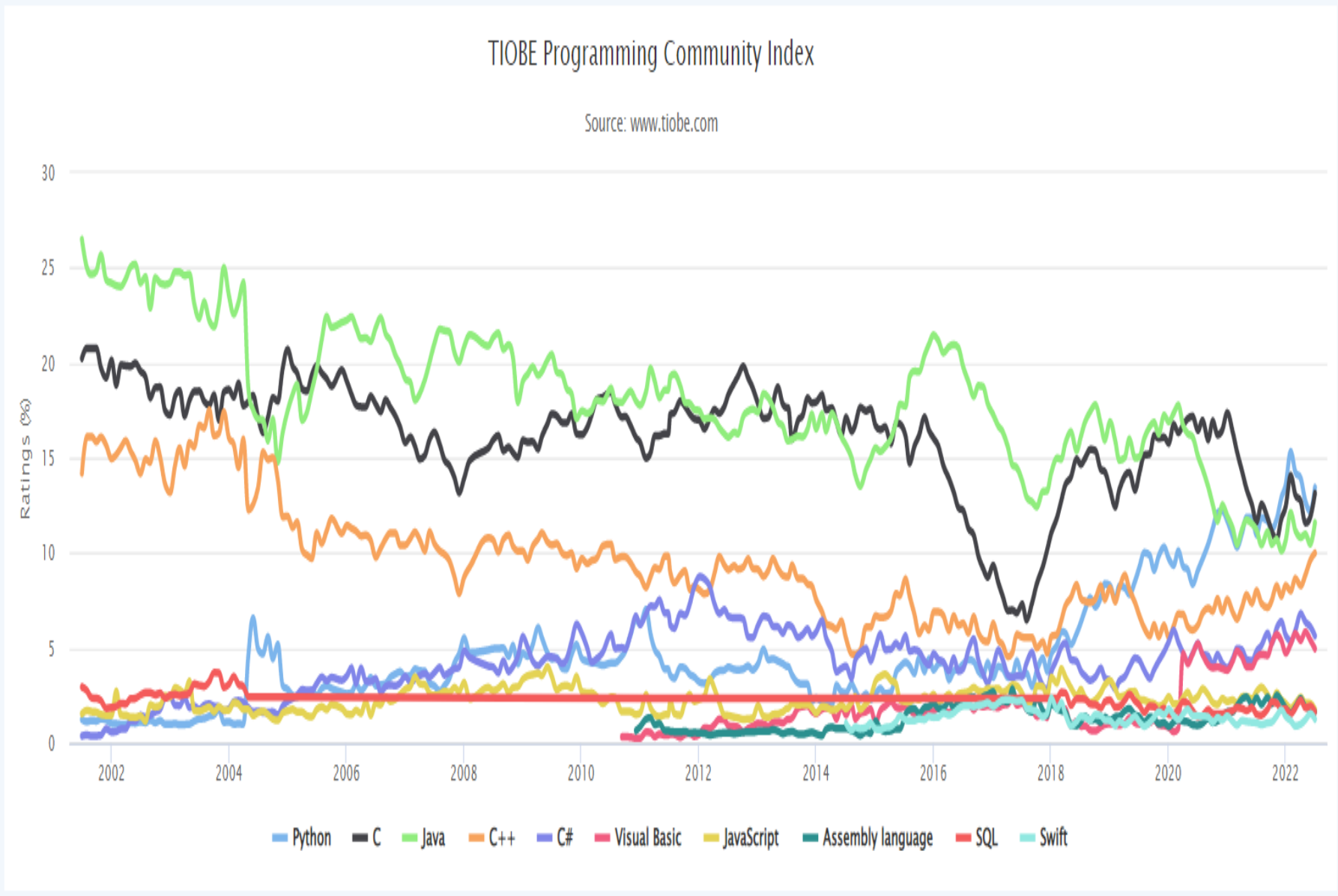
- **TIOBE Index for February 2023:**

- https://www.tiobe.com/tiobe-index/

when starting to build a new software system. The definition of the TIOBE index can be found here.

Feb 2023	Feb 2022	Change	Programming Language	Ratings	Change
1	1		Python	15.49%	+0.16%
2	2		C	15.39%	+1.31%
3	4	↑	C++	13.94%	+5.93%
4	3	↓	Java	13.21%	+1.07%
5	5		C#	6.38%	+1.01%
6	6		Visual Basic		-1.09%
7	7		JavaScript	2.52%	
8	10	↑	SQL	2.12%	+0.58%
9	9		Assembly language	1.38%	-0.21%
10	8	↓	PHP	1.29%	-0.49%
11	11		Go	1.11%	-0.12%
12	13	↑	R	1.08%	-0.04%
13	14	↑	MATLAB	0.99%	-0.04%
14	15	↑	Delphi/Object Pascal	0.95%	+0.05%
15	12	↓	Swift	0.93%	-0.25%
16	16		Ruby	0.83%	-0.06%
17	19	↑	Perl	0.79%	-0.01%

Programação Orientada a Objetos



CEUB
EDUCAÇÃO SUPERIOR

Por que estudar Linguagens de Programação?

Avanço global da computação:

Aprender programação é uma das habilidades do século 21 e deveria ser tão importante quanto ler ou escrever.

Ter a capacidade de entender que nem sempre as melhores linguagens de programação são as mais populares.

Desenvolver o entendimento que existem muitas linguagens de programação, cada uma com a finalidade de específica para resolução problemas.

Áreas de Aplicação de Linguagens de Programação

As linguagens de programação tem sido utilizadas em um grande número de aplicações nas mais diversas áreas do conhecimento, tais como:

- **Aplicações Cientificas**
- **Aplicações para Hardware**
- **Inteligência Artificial e Ciência de Dados**
- **Software básico**
- **Criação de Softwares comerciais**

CEUB
EDUCAÇÃO SUPERIOR

Características de uma Linguagem de Programação

Torna-se interessante comentar que o principal objetivo de uma LP é dar suporte ao programador no desenvolvimento dos sistemas.

Entre as características gerais que definem uma boa linguagem pode-se destacar:

- **Simplicidade: clareza e concisão semântica (linguagem com um mínimo número de conceitos e estruturas), e clareza sintática (sintaxe deve representar cada conceito de uma maneira apenas).**

- **Suporte para abstração de dados: representação de um objeto deve incluir somente os atributos relevantes.**

Características de uma Linguagem de Programação

- **Expressividade: refere-se a facilidade com que um objeto pode ser representado; a linguagem deve oferecer estruturas de dados e de controle apropriadas.**

- **Ortogonalidade: refere-se a interação entre conceitos, isto é, o grau com que diferentes conceitos podem ser combinados uns com os outros de uma maneira consistente; ortogonalidade reduz o número de exceções das regras de uma linguagem, tornando mais fácil o seu aprendizado e memorização.**

Características de uma Linguagem de Programação

- **Suporte à manutenção e portabilidade: habilidade de manter programas que devem ser fáceis de entender e alterar; é afetada pelas características anteriores.**

- **Eficiência: a avaliação precisa de uma linguagem, baseada em critérios pré-definidos, é extremamente importante; as medidas mais comuns são a eficiência da execução do programa, da tradução do programa, e da criação, teste e uso do programa.**

CEUB
EDUCAÇÃO SUPERIOR

Escolha de uma Linguagem de Programação

Existe um grande número de LP deve-se escolher qual é a mais adequada para cada tipo de aplicação.

Existem várias considerações que devem ser levadas em conta, técnicas e não técnicas, estratégicas e táticas.

A escolha da linguagem mais adequada está intimamente ligada a três fatores:

- **complexidade do sistema a ser desenvolvido;**
- **características peculiares da aplicação (por exemplo, sistemas de tempo real); e**
- **facilidades que as linguagens oferecem ao suporte de metodologias de desenvolvimento.**

Escolha de uma Linguagem de Programação

ATENÇÃO!!!!!

Para pequenos programas, feitos e mantidos por uma pessoa, é provável que a linguagem mais adequada seja aquela melhor dominada pela pessoa.

No entanto, para grandes sistemas com programação em tempo real, vários tipos de exceções a serem tratadas e muitas pessoas envolvidas, a escolha da linguagem mais adequada deve ser feita criteriosamente.

Critérios de Avaliação de Linguagens

- **Um dos critérios mais importantes para julgar uma linguagem de programação é a facilidade com que os programas são lidos e entendidos.**

- **Antes dos anos 70: linguagens foram criadas pensado em termos de escrita de código.**

  - **Eficiência e legibilidade da máquina;**
  - **As linguagens foram projetadas mais do ponto de vista do computador do que do usuário.**

- **Na década de 70 foi desenvolvido o conceito de ciclo de vida de software: manutenção.**

Critérios de Avaliação: Legibilidade

- **A legibilidade deve ser considerada no contexto do domínio do problema.**
- **Quão facilmente um programa pode ser lido e entendido**
- **Um programa escrito em uma linguagem não apropriada para o domínio do problema se mostra antinatural, "enrolado" e difícil de ser lido.**

Critérios de Avaliação: Legibilidade

Simplicidade geral: A simplicidade geral de uma linguagem de programação afeta fortemente sua legibilidade;

Uma linguagem com um grande número de componentes básicos é mais difícil de ser manipulada do que uma com poucos desses componentes.

- **Os programadores que precisam usar uma linguagem grande tendem a aprender um subconjunto dela e ignorar seus outros recursos.**
- **Isso pode ser um problema quando o leitor do programa aprende um conjunto diferente de recursos daquele que o autor aplicou em seu programa.**

Critérios de Avaliação: Legibilidade

Simplicidade geral: **Uma segunda característica que complica a legibilidade é a multiplicidade de recursos (mais que uma maneira de realizar uma operação particular);**

Multiplicidade de recursos complica a legibilidade: Exemplo

```
cont = cont + 1
cont += 1
cont++
++cont
```

CEUB
EDUCAÇÃO SUPERIOR

Critérios de Avaliação: Legibilidade

Simplicidade geral:

A simplicidade de linguagens, no entanto pode ser levada ao extremo, por exemplo a forma e o significado da maioria das instruções em Assembly são modelos de simplicidade, entretanto torna os programas em Assembly menos legíveis.

Falta instruções de controle mais complexas, o que torna necessário o uso de mais comandos para expressar problemas do que os necessário em linguagens de alto nível.

Assembly – Exemplo

```
R12: 0x555555554580 (<_start>:  xor    ebp,ebp)
R13: 0x7fffffffe1f0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x5555555546ab <frame_dummy+43>:
    jmp    0x5555555545f0 <register_tm_clones>
   0x5555555546b0 <main>:       push   rbp
   0x5555555546b1 <main+1>:     mov    rbp,rsp
=> 0x5555555546b4 <main+4>:     mov    edi,0xf
   0x5555555546b9 <main+9>:     call   0x5555555546f5 <recursion_c>
   0x5555555546be <main+14>:    mov    esi,eax
   0x5555555546c0 <main+16>:    lea    rdi,[rip+0x10d]        # 0x5555555547d4
   0x5555555546c7 <main+23>:    mov    eax,0x0
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffe110 --> 0x555555554750 (<__libc_csu_init>:     push   r15)
0008| 0x7fffffffe118 --> 0x7ffff7a5a2e1 (<__libc_start_main+241>:       mov    e
di,eax)
0016| 0x7fffffffe120 --> 0x40000
0024| 0x7fffffffe128 --> 0x7fffffffe1f8 --> 0x7fffffffe4e4 ("/home/felipe/tst")
0032| 0x7fffffffe130 --> 0x1f7b9b508
0040| 0x7fffffffe138 --> 0x5555555546b0 (<main>:        push   rbp)
0048| 0x7fffffffe140 --> 0x0
```

Critérios de Avaliação: Legibilidade

Facilidade de ler e escrever programas

- **Legibilidade influi:**
  - **desenvolvimento e depuração de programas**
  - **manutenção de programas**
  - **desempenho de equipes de programação**

Fatores que melhoram a legibilidade:

- **Abstração de dados**
- **Comandos de controle**
- **Modularização de programas**
- **Documentação**
- **Convenções léxicas, sintaxe e semântica**
  - **Exemplo em Java: nomes de classes iniciam por letra maiúscula, nomes de campos usam letras minúsculas**

Critérios de Avaliação: Capacidade de Escrita

- **A capacidade de escrita é a medida da facilidade em que uma linguagem pode ser usada para criar programas para um domínio de problema escolhido;**

- **A maioria das características da linguagem que afetam a legibilidade também afetam a capacidade de escrita;**

- **Deve ser considerada no contexto do domínio de problema-alvo da linguagem.**

Critérios de Avaliação: Capacidade de Escrita

- **Simplicidade e Ortogonalidade:**
  - **Se uma linguagem de programação tem um grande número de construções, alguns programadores não estarão familiarizados com todas;**
  - **Pode acarretar o uso incorreto de alguns recursos e uma utilização escassa de outros que podem ser mais elegantes ou eficientes do que os usados;**
  - **Podem ocorrer usos de recursos desconhecidos com resultados não esperados.**

Critérios de Avaliação: Capacidade de Escrita

Suporte para abstração:

Abstração: capacidade de definir e, depois usar estruturas ou operações complicadas de uma maneira que permita ignorar muito dos detalhes.

Exemplo: uso de funções provenientes de bibliotecas;

Tipos de Abstração

- **Abstração de Processo: algoritmos em geral;**
- **Abstração de Dados: tipos de dados e estruturas de dados.**

Critérios de Avaliação: Confiabilidade

- **Um programa é confiável se ele se comportar de acordo com suas especificações sob todas as condições.**

- **Recursos que afetam a confiabilidade:**
  - **Verificação de Tipos;**
  - **Manipulação de Exceções;**
  - **Apelidos (Aliasing);**
  - **Legibilidade e Facilidade de Escrita;**

Critérios de Avaliação: Confiabilidade

Verificação de Tipos:

Visa testar se existem erros de tipos de dados no programa por meio do compilador ou durante a execução do programa;

- **A verificação de tipos durante a compilação é a mais indicada. Quanto antes for detectado o erro, menos caro é fazer todos os reparos necessários.**
  - **Exemplo: Java**
- **A verificação de tipos em C é bastante fraca:**

```
intvet[50];
vet[100] = 8.5;
```

Critérios de Avaliação: Confiabilidade

Manipulação de Exceções:

- Capacidade de um programa de interceptar erros em tempo de execução, pôr em prática medidas corretivas e, depois, prosseguir.
- Exemplos em C++ e Java:

```
try
{
   ...
}
catch (Exception &exception)
{
   ...
}
```

```
try
{
   ...
}
catch(Exception e)
{
   ...
}
```

Critérios de Avaliação: Confiabilidade

Apelidos (Aliasing):

Referências à mesma posição de memória

Ex.: python, ponteiros e unions em C

Legibilidade e Capacidade de Escrita afetam confiabilidade

Critérios de Avaliação: Confiabilidade

Legibilidade e Facilidade de Escrita:

- Tanto a legibilidade como a facilidade de escrita influenciam a confiabilidade.
- Quanto mais fácil é escrever um programa, mais probabilidade ele tem de estar correto.
- Programas de difícil leitura complicam também sua escrita e sua modificação.

Critérios de Avaliação: Custo

Custo final de uma linguagem de programação é uma função de muitas de suas características.

- **Custo de Treinamento: cresce em função da simplicidade e da ortogonalidade da linguagem e da experiência dos programadores;**
- **Custo da Escrita: Os esforços originais para projetar e implementar linguagens de alto nível foram motivados pelos desejos de diminuir os custos para criar software;**
- **Custo da Má confiabilidade: falhas podem ocasionar insucesso do software e ações judiciais;**
- **Custo de Manutenção: depende principalmente da legibilidade. O custo de manutenção pode atingir de duas a quatro vezes o custo de desenvolvimento;**

Critérios de Avaliação: Custo

- **Custo do Sistema de Implementação**: uma linguagem de programação cujo sistema de implementação seja caro, ou rode somente em hardware caro, terá muito menos chance de tornar-se popular;

- **Custo do Projeto da Linguagem**: se uma linguagem de programação exigir muitas verificações de tipos durante a execução, proibirá a execução rápida do código;

- **Custo de Compilação**: problema amenizado com o surgimento de compiladores otimizados e de processadores mais rápidos.

Fatores que influenciam na escolha de uma linguagem

- **Implementação**
  - **Disponibilidade quanto à plataforma**
  - **Eficiência: velocidade de execução do programa objeto**

- **Competência na Linguagem de Programação**
  - **Experiência do Programador**
  - **Competência do grupo envolvido**

- **Portabilidade**
  - **Necessidade de executar em diferentes plataformas**

Fatores que influenciam na escolha de uma linguagem

- **Ambientes de Programação**
  - **Ferramentas para desenvolvimento de software diminuem o esforço de programação**
  - **Bibliotecas**

- **Modelo de Computação**
  - **Aplicação vs modelo de computação**
  - **Ex.: para realização de busca heurística é adequado o Paradigma Lógico, para simulações, o Paradigma Orientado a Objeto**

Classificação das Linguagens

- As linguagens de programação podem ser classificadas em relação a três critérios:
  - Em relação ao nível:
    - Baixo nível, Médio nível, ou Alto nível;
  - Em relação à geração:
    - 1ª Geração, 2ª Geração, 3ª Geração, 4ª Geração, ou 5ª Geração;
  - Em relação ao paradigma:
    - Imperativo, Funcional, Lógico, Orientado a Objetos, ...;

Classificação quanto ao Nível

Baixo Nível:

- **As linguagens de Baixo Nível são aquelas voltadas para a máquina, ou seja as que são escritas utilizando as instruções do microprocessador do computador.**

- **São genericamente chamadas de linguagens Assembly. Os programas escritos com Alto Nível geralmente podem ser convertidos com programas especiais para Baixo Nível.**

Classificação quanto ao Nível

Baixo Nível:

- **Vantagens:**
  - Os programas são executados com maior velocidade de processamento;
  - Os programas ocupam menos espaço na memória;

- **Desvantagens:**
  - Em geral, programas em Assembly tem pouca portabilidade, isto é, um código gerado para um tipo de processador não serve para outro;
  - Códigos Assembly não são estruturados, tornando a programação
    mais difícil...

Classificação quanto ao Nível

Médio Nível:

- **São linguagens voltadas ao ser humano e a máquina;**
- **Estas linguagens são uma mistura entre as linguagens de Alto Nível e as de Baixo Nível;**
- **Estas linguagens de programação contêm comandos muito simples e outros mais complicados, o que pode tornar a programação um pouco "complicada".**

Classificação quanto ao Nível

Médio Nível:

- **Exemplo – Linguagem C: pode-se acessar registros do sistema e trabalhar com endereços de memória (características de linguagens de baixo nível) e ao mesmo tempo realizar operações de alto nível (if...else; while; for).**

```
int x, y, *p;
y = 0;
p = &y;
x = *p;
x = 4;
(*p)++;
x--;
(*p) += x;
```

```
int vet[6] = {1, 2, 3, 4, 5};

printf("%d\n", vet);
printf("%d\n", *vet);
printf("%d\n", *(vet + 2));
```

Classificação quanto ao Nível

Médio Nível:

- **Vantagens:**
  - **Geralmente são linguagens poderosas, permitindo a criação de diversos softwares, desde jogos a programas de alta performance.**
- **Desvantagens:**
  - **Alguns comandos têm uma sintaxe um pouco difícil de compreender.**

Classificação quanto ao Nível

Alto Nível:

- São linguagens voltadas para o ser humano. Em geral utilizam sintaxe mais estruturada, tornando o seu código mais fácil de entender.

- São linguagens independentes de arquitetura.
  - Um programa escrito em uma linguagem de alto nível, pode ser migrado de uma máquina a outra sem nenhum tipo de problema.

- Permitem ao programador se esquecer completamente do funcionamento interno da máquina.
  - Sendo necessário um tradutor que entenda o código fonte e as características da máquina.

Classificação quanto ao Nível

Alto Nível:

- **Exemplos: Lua, Java, C#, Python, etc...**

```
nota = io.read()

if nota < 3.0 then
  io.write("Reprovado")
elseif nota >= 5.0 then
  io.write("Aprovado")
else
  io.write("Prova final")
end
```

```
Scanner entrada = new Scanner(System.in);
mes = entrada.nextInt();
switch (mes)
{
  case 1:System.out.println("Janeiro");
         break;
  case 2:System.out.println("Fevereiro");
         break;
  case 3:System.out.println("Março");
         break;
  default:
         System.out.println("Outro");
         break;
}
```

Classificação quanto ao Nível

Alto Nível:

– **Vantagens:**

- **Por serem compiladas ou interpretadas, têm maior portabilidade, podendo ser executados em várias plataformas com pouquíssimas modificações.**
- **Em geral, a programação é mais fácil.**

– **Desvantagens:**

- **Em geral, as rotinas geradas (em linguagem de máquina) são mais genéricas e, portanto, mais complexas e por isso são mais lentas e ocupam mais memória.**

Classificação quanto à Gerações

- 1ª Geração
  - linguagens em nível de máquina
- 2ª Geração
  - linguagens de montagem (Assembly)
- 3ª Geração
  - linguagens orientadas ao usuário
- 4ª Geração
  - linguagens orientadas à aplicação
- 5ª Geração
  - linguagens do conhecimento (IA, Funcionais)

Classificação quanto ao Paradigma

- Paradigma é um **modelo interpretativo** (ou conceitualização) de uma **realidade**.

- Permite organizar as ideias com vista:
  - Ao entendimento dessa realidade;
  - À determinação de qual a melhor forma de atuar sobre essa realidade.

- Pode dizer-se que um paradigma é um **ponto de vista**: um ponto de vista que determina como uma realidade é entendida e como se atua sobre ela.

Classificação quanto ao Paradigma

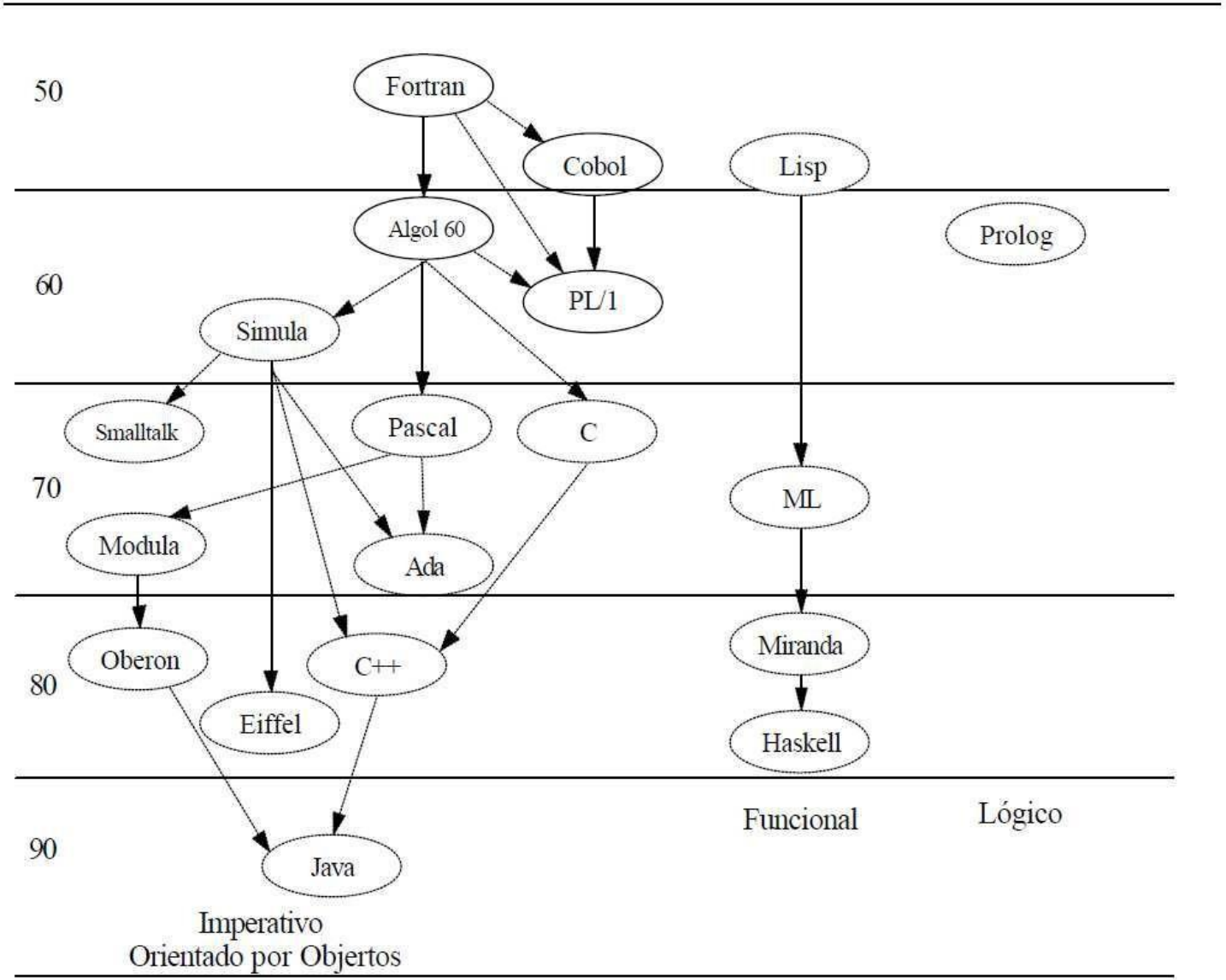
- Algumas linguagens criadas durante a história introduziram **novas formas de se pensar sobre programação**, resultando em formas distintas de modelagem de soluções de software.
  - FORTRAN (imperativa);
  - LISP (funcional);
  - Simula (orientadas a objetos);
  - Prolog (lógica).

- Outras linguagens são o resultado da evolução de linguagens mais antigas, muitas vezes **mesclando** características de diferentes linguagens existentes.
  - Por exemplo, C++ é uma evolução do C com características de orientação a objetos, importadas de Simula.

Classificação quanto ao Paradigma

- **Paradigma imperativo** (sequência, atribuição, estado): Basic, Pascal, C, Ada, Fortran, Cobol, Assembly...
- **Paradigma funcional** (função, aplicação, avaliação): Lisp, Haskell, Erlang, Scheme...
- **Paradigma lógico** (relação, dedução): Prolog.
- **Paradigma orientado a objetos** (objeto, estado, mensagem): C++, Java, C#, Eiffel, Smalltalk, Python...
- **Paradigma concorrente** (processo, comunicação (síncrona ou assíncrona)): C++, C#, Java...

Classificação quanto ao Paradigma



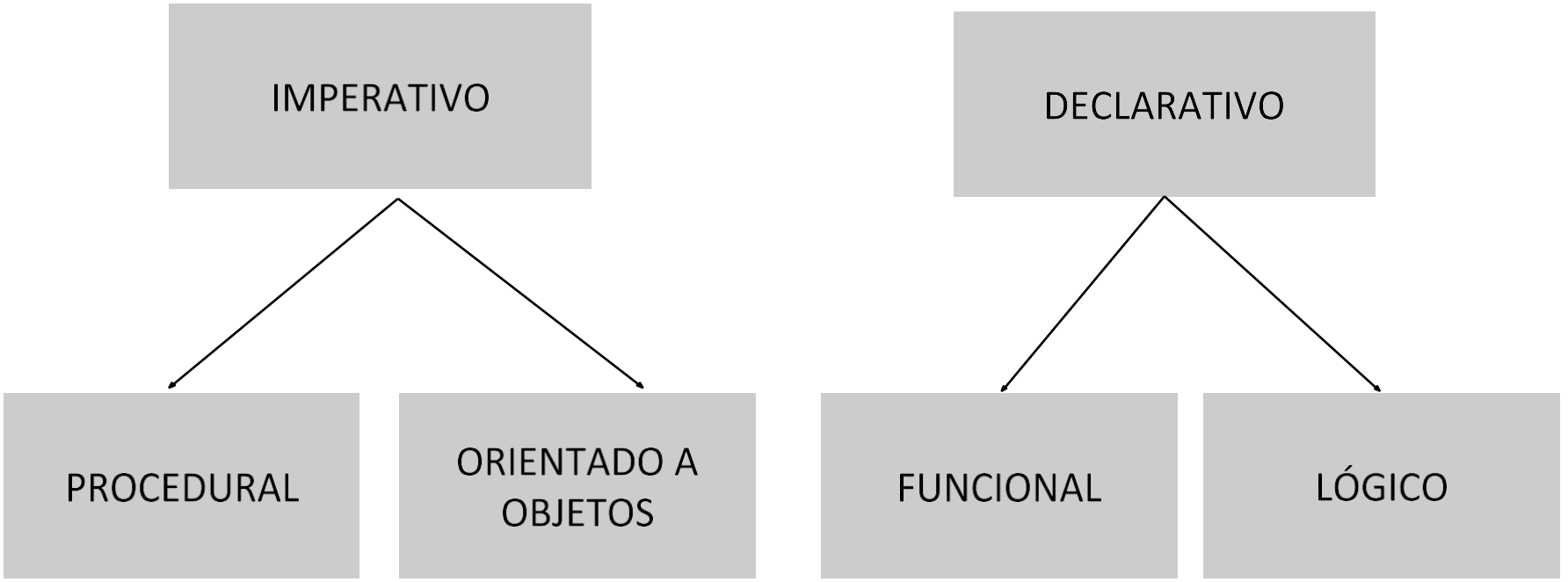
Paradigmas de programação

Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e a execução do programa.

- **Por exemplo:**
  - **Em programação orientada a objetos, o foco e abstração do mundo real, criação de classes e objetos**
  - **Em programação lógica os programadores abstraem o programa como um conjunto de predicados que estabelecem relações entre objetos (axiomas), e uma meta (teorema) a ser provada usando os predicados.**

- **Diferentes linguagens de programação propõem diferentes paradigmas de programação.**

Paradigmas de programação



Paradigmas de programação

Categorias: programação imperativa e declarativa

Programação imperativa

Descreve a computação como ações, enunciados ou comandos que mudam o estado (variáveis) de um programa, enfatizando como resolver um problema.

Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.

Paradigmas de programação

Categorias: programação imperativa e declarativa

Programação imperativa

O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo.

- **Exemplos:**
  - **procedimental: C, Pascal, etc, e**
  - **orientado a objetos: Smalltalk, Java, etc**

Paradigmas de programação

Programação declarativa

- **Descreve o que o programa faz e não como seus procedimentos funcionam.**

- **Ênfase nos resultados, no que se deseja obter.**

- **Exemplos: paradigmas**

  - **Funcional: Haskell, F#, Miranda, OCaml, LISP, etc, e**

  - **Lógico: Prolog, etc.**

CEUB
EDUCAÇÃO SUPERIOR

Referências

- Sebesta, Robert W. Conceitos de Linguagens de Programação. Editora Bookman, 2011.
- http://edirlei.3dgb.com.br/
- Sampaio, A. (2008) “Paradigmas de Linguagens de Programação”,
- http://www.cin.ufpe.br/~in1007/transparencias/aulaIntroducaoPLP.ppt, Agosto.
- Paula, A. (2008) “Paradigmas de Linguagens de Programação Motion Capture White
- Fernandes, E., Carvalho, K., Villar, L., Getirana, N. e Gaudêncio, V. (2008)
- “Paradigmas de linguagem de programação Motion Capture White Paper”,
- http://http://www.inf.unisinos.br/~anapaula/disciplinas/60023/, Agosto.