UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

BCC6002 - Aspectos de Linguagens de Programação

Prof. Dr. Rodrigo Hübner

Aula 01: conceitos e análise inicial de LPs; domínios de LPs; paradigmas

Análise sobre LPs

Níveis de LPs: baixo nível

Parâmetros	Nível de máquina	Nível de montagem
Nível de hierarquia	Menor nível de hierarquia; zero abstração	Acima do nível de máquina; pequena abstração
Curva de aprendizado	Não é "humanamente" legível	"Fácil" de ler e manter
Como é escrita	Código binário (0s e 1s)	Escrita simples em inglês
Geração	1ª geração de LPs	2ª geração de LPs
Eficiência de memória	Executado diretamente	Necessário um montador para; converter assembly em código de máquina

Níveis de LPs: alto nível

Parâmetros	Nível de máquina	Nível de montagem
Nível de entendimento	Próximo ao hardware	Próximo ao usuário (programador)
Tempo de execução	Rápido de executar	Pode ser muito lento (abstração)
Ferramenta necessária	Necessário um conversor de montagem	Necessita do compilador
Portabilidade	Não	Portável (na maioria dos casos)
Eficiência de memória	Eficiente	Pouca eficiência
Mantenimento	Difícil	Mais fácil

Tipos / Modelos de programação

- **Procedural**
  - Série de procedimentos bem estruturados
  - `Adobe Dreamweaver`, `Eclipse` ou `Microsoft visual studio`, `BASIC`, `C`, `Java`, `PASCAL`, `Go!`, `Julia` e `FORTRAN`
- **Orientado a objeto**
  - Facilita abstração, encapsulamento, polimorfismo, herança e classes
  - `Java`, `C++`, `C#`, `Python`, `Javascript` e muitas outras

Tipos / Modelos de programação

- **Programação lógica**
  - Baseado na lógica formal
  - Não diz "como" fazer, mas emprega restrições sobre o que deve ser considerado
  - `Prolog` e `ASAP` (*Answer Set A Programming*)
- **Funcional**
  - Baseado em aplicação e composição de funções
  - Fundamento do cálculo lambda
  - `Haskell`, `SML`, `Scala`, `F#`, `ML`, `Scheme`, `Racket` (+ atuais), `Rust`.

Tipos / Modelos de programação

- **Script**
  - Não necessita de estágio de compilação (traduzida)
  - Criação de plugins, extensões, etc
  - `Javascript`, `PHP` e `PERL` (server side)
  - `Javascript`, `AJAX`, `Jquery` e `Shell` (client side)
  - `PERL` e `Python` (administração de sistemas)
  - `Ruby`, `Python`, `Javascript` e `Typescript` (desenvolvimento web)

Tipos / Modelos de programação

- **Concorrentes**
  - Facilita programação para hardware multicore e GPUs.
  - `Cilk`, `Fortress`, `X10`, `Chapel`, `Futhark`, `NESL`, `ZPL`, `PLH`, entre outras

Índices de LPs no mundo

- **TIOBE**: https://www.tiobe.com/tiobe-index/
- **Top 10 In-Demand Programming Languages for 2024** : https://www.crossover.com/blog/top-10-in-demand-programming-languages-for-2024
- **Analytics Insight**: https://www.analyticsinsight.net/category/insights/coding/programming-languages/

Influências no projeto de linguagens

- **Arquitetura do Computador**
  - Arquitetura de von Neumann
  - Arquiteturas *multicore*
- **Metodologias de programação**
  - Orientada a procedimentos (`Pascal`, `C`, ...)
  - Orientada a dados (`Ada`, `Fortran`, `Cobol`, ...)
  - Orientação a objetos (`Smalltalk`, `C++`, `C#`, `Python`, `Java`, ...)

Domínios de Programação

- **Aplicações científicas**
  - Estruturas simples (arranjos e matrizes)
  - Muitas operações com pontos flutuantes
  - `Fortran`, `Algol`, `C/C++`, `Go`, `Ada`
  - `Fortress`, `Matlab` (Versão open source -> `Octave`), `Julia`, `R`, `numpy` (`Python`)

Domínios de Programação

- **Aplicações comerciais**
  - Produção de relatórios
  - Formatação de números decimais e caracteres
    - Software contábil, vendas, etc
  - `Cobol`, `Java`, `C++` (`Builder`), `C#` (`Visual Studio`), `Delphi`

Domínios de Programação

- **Inteligência artificial**
  - Manipulação de símbolos (lista ligada)
  - Criação e execução de código
  - `Lisp`, `Prolog` (aplicações legadas)
  - `C/C++`, `Python` (`SciKitLearn` [`sklearn`]), `TensorFlow` [abstração `keras`]

Domínios de Programação

- **Software de sistema**
  - Eficiência devido ao uso contínuo
  - Desenvolvimento de SOs (kernel)
  - `C/C++`
  - `D`, `Go`, `Rust` (também encontrados por aí...)

Domínios de Programação

- **Web**
  - Código [geralmente] interpretado dentro do mesmo conjunto de documentos
  - `Javascript`, `Typescript`, `PHP`, `Java`, `Ruby`, `Python`

Critérios para avaliar LPs

- **Facilidade de leitura (legibilidade)**
  - Simplicidade
  - Ortogonalidade
  - Tipos de dados
  - Sintaxe
- **Facilidade de escrita**
  - Simplicidade e ortogonalidade
  - Suporte para abstração
  - Expressividade

Critérios para avaliar LPs

- **Confiabilidade**
  - Verificação de tipos
  - Manipulação de exceções
  - Apelidos
  - Facilidade de leitura e escrita
- **Custo**
  - Treinar programadores
  - Escrever, compilar e executar programas
  - Confiabilidade

Critérios para avaliar LPs

- **Outros critérios**
  - Portabilidade
  - Padronização

Programadores, projetistas e implementador da LP possui visões diferentes

Próxima aula

- Conceitos de sistemas de computação: compiladores, *assemblers*, *linkers* (estático/dinâmico), *loaders*, interpretadores, MVs, depuradores, `JIT`, entre outros...