Conceitos e Análise Inicial de LPs; Domínios de LPs; Paradigmas

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

A tabela apresenta uma análise comparativa entre dois níveis de programação de computadores: o nível de máquina e o nível de montagem.

O nível de máquina é o mais baixo nível de programação, onde as instruções são escritas diretamente em código binário, ou seja, uma sequência de 0s e 1s que representam as operações a serem realizadas pela máquina. Esse nível é considerado o mais próximo da linguagem de máquina e, por isso, é extremamente detalhado e pouco abstrato. O nível de hierarquia é o menor, e não há abstração, ou seja, tudo deve ser especificado em detalhes.

Por outro lado, o nível de montagem é um pouco mais abstrato e usa uma linguagem de programação chamada assembly, que é uma linguagem de baixo nível, mas ainda é mais fácil de entender e de escrever do que o código binário. Nesse nível, a linguagem é escrita em inglês, tornando a escrita mais simples e acessível. A hierarquia é um pouco maior do que no nível de máquina, permitindo uma maior facilidade na escrita e manutenção do código.

A curva de aprendizado para o nível de máquina é muito alta, pois não é humanamente legível, o que significa que é muito difícil para os seres humanos entenderem o código e escreverem programas nesse nível. Por outro lado, a curva de aprendizado para o nível de montagem é relativamente mais baixa, pois é mais fácil de ler e entender do que o código binário.

O nível de máquina é considerado a primeira geração de linguagens de programação, enquanto o nível de montagem é considerado a segunda geração de linguagens de programação.

Finalmente, a eficiência de memória para o nível de máquina é executada diretamente, enquanto para o nível de montagem é necessário um montador para converter o código de assembly em código de máquina executável, o que pode ser visto como um pequeno custo adicional.

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	Port´ável (na maioria dos casos)
Eficiência de memória	Eficiente	Pouca eficiência
Mantenimento	Difícil	Mais fácil

A tabela apresenta uma análise comparativa entre duas abordagens de programação de alto nível: o nível de máquina e o nível de montagem.

O nível de máquina em programação de alto nível é caracterizado por uma linguagem de programação que é muito próxima do hardware do computador. Isso significa que essa abordagem é adequada para aplicações que exigem alto desempenho ou manipulação de recursos de hardware específicos. O nível de entendimento é próximo ao hardware, o que torna essa abordagem menos abstrata.

Já o nível de montagem em programação de alto nível é caracterizado por uma linguagem de programação mais próxima do usuário (programador) e menos próxima do hardware. Isso significa que essa abordagem é mais adequada para aplicações que exigem facilidade de escrita e manutenção de código. O tempo de execução pode ser mais lento devido à maior abstração.

Para o nível de máquina, é necessário um conversor de montagem para traduzir o código de alto nível em código de máquina executável. Para o nível de montagem, é necessária uma ferramenta de compilação que traduz o código de alto nível em código de máquina. Portanto, a portabilidade do nível de montagem é maior do que a do nível de máquina.

A eficiência de memória é eficiente no nível de máquina, pois as instruções são executadas diretamente pelo hardware do computador. No entanto, no nível de montagem, a eficiência de memória pode ser menor devido à maior abstração. Por fim, a manutenção do código é

mais difícil no nível de máquina, enquanto no nível de montagem é mais fácil devido à sua maior abstração e proximidade com o usuário.

Tipos / Modelos de programação

Procedural: é um modelo de programação baseado em uma série de procedimentos bem estruturados, onde o programa é dividido em funções/procedimentos, que podem ser chamados por outras partes do código. As linguagens de programação mais utilizadas nesse modelo incluem BASIC, C, Java, PASCAL, Go!, Julia, e FORTRAN. Alguns dos ambientes de desenvolvimento de software mais populares para essa abordagem são Adobe Dreamweaver, Eclipse ou Microsoft Visual Studio.

Orientado a objetos: é um modelo de programação baseado na abstração, encapsulamento, polimorfismo, herança e classes. Ele se concentra em objetos, que contêm dados e métodos que operam nesses dados. As linguagens de programação mais comuns que usam esse modelo incluem Java, C++, C#, Python, Javascript e muitas outras.

Programação lógica: é baseado em lógica formal e emprega restrições sobre o que deve ser considerado em vez de dizer "como" fazer algo. Ele é usado para resolver problemas complexos de forma mais declarativa, em vez de imperativa. As linguagens de programação mais populares nesse modelo incluem Prolog e ASAP (Answer Set A Programming).

Funcional: é um modelo de programação baseado em aplicação e composição de funções. Ele se concentra em funções que recebem argumentos e retornam valores, e se baseia no cálculo lambda. As linguagens de programação mais populares que usam esse modelo incluem Haskell, SML, Scala, F#, ML, Scheme e Racket (+ atuais).

Script: é um modelo de programação que não precisa passar pelo estágio de compilação antes de ser executado. Em vez disso, é executado diretamente pelo interpretador da linguagem. Esse modelo é amplamente usado para criar plugins, extensões e outras funcionalidades personalizadas. As linguagens de programação mais comuns que usam esse modelo incluem Javascript, PHP e PERL (server-side), Javascript, AJAX, Jquery e Shell (client-side), PERL e Python (administração de sistemas) e Ruby, Python, Javascript e Typescript (desenvolvimento web).

Influências no projeto de linguagens

Arquitetura do Computador: A arquitetura do computador é uma influência fundamental na concepção de linguagens de programação. A arquitetura de von Neumann, por exemplo, influenciou muitas das linguagens de programação criadas no século XX. Essa arquitetura descreve uma máquina que tem uma unidade central de processamento (CPU) capaz de executar instruções armazenadas em memória. Além disso, a evolução das arquiteturas multicore, com múltiplos núcleos de processamento em uma única CPU, também pode afetar a concepção de linguagens de programação.

Metodologias de programação: As metodologias de programação também são influências importantes no projeto de linguagens de programação. A orientação a processos (ou a procedimentos) foi uma das primeiras metodologias utilizadas para estruturar programas de computador e muitas linguagens foram desenvolvidas com essa abordagem em mente. A orientação a dados, por outro lado, enfatiza a organização dos dados em um programa, e a orientação a objetos é uma abordagem mais recente que se concentra em encapsular dados e comportamentos em objetos.

Outras influências: Além dessas influências, outras podem afetar o projeto de linguagens de programação. A evolução da tecnologia da informação, por exemplo, pode influenciar a necessidade de novas linguagens de programação para atender a novas demandas. As tendências em computação em nuvem, inteligência artificial, aprendizado de máquina e outras áreas também podem ter impacto no design de linguagens de programação.

Domínios de Programação

Aplicações científicas: Essas aplicações geralmente envolvem estruturas simples, como arranjos e matrizes, e muitas operações com números de ponto flutuante. As linguagens comumente usadas incluem Fortran, Algol, C/C++, Go, Ada, Fortress, Matlab e R.

Aplicações comerciais: Essas aplicações geralmente envolvem a produção de relatórios e a formatação de números decimais e caracteres. As linguagens mais comuns incluem Cobol, Java, C++ (Builder), C# (Visual Studio) e Delphi.

Inteligência artificial: As aplicações de inteligência artificial geralmente envolvem a manipulação de símbolos, como listas ligadas, e a criação e execução de código. As linguagens comumente usadas incluem Lisp, Prolog, C/C++ e Python (SciKitLearn [sklearn], TensorFlow [abstração Keras]).

Software de sistema: Essas aplicações são projetadas para serem altamente eficientes, uma vez que são usadas continuamente. Elas geralmente envolvem o desenvolvimento de sistemas operacionais (kernel). As linguagens comumente usadas incluem C/C++, D, Go e Rust.

Web: Essas aplicações geralmente envolvem código interpretado dentro do mesmo conjunto de documentos. As linguagens comumente usadas incluem Javascript, Typescript, PHP, Java, Ruby e Python.

Critérios para avaliar LPs

Facilidade de leitura (legibilidade): A legibilidade de uma LP refere-se à facilidade de ler e entender o código escrito nessa linguagem. A simplicidade, ortogonalidade (ou consistência) dos conceitos, a disponibilidade de tipos de dados e a sintaxe são importantes para tornar o código mais legível.

Facilidade de escrita: A facilidade de escrita refere-se à facilidade com que os programadores podem escrever código em uma determinada LP. A simplicidade e ortogonalidade (ou consistência), suporte para abstração e expressividade são importantes para tornar a escrita do código mais fácil.

Confiabilidade: A confiabilidade refere-se à segurança do código escrito em uma determinada LP. A verificação de tipos, a manipulação de exceções, os apelidos e a facilidade de leitura e escrita são importantes para garantir a confiabilidade do código.

Custo: O custo da LP refere-se a quanto tempo e recursos financeiros são necessários para treinar programadores, escrever, compilar e executar programas, e garantir a confiabilidade do código. As LPs com maior facilidade de leitura e escrita e confiabilidade podem ter um custo menor.

Outros critérios: Alguns outros critérios importantes incluem a portabilidade e a padronização da LP. A portabilidade refere-se à capacidade de um programa escrito em uma determinada LP ser executado em diferentes sistemas operacionais e hardware. A padronização refere-se à conformidade com os padrões da indústria de programação.

Por fim, é importante ressaltar que os programadores, projetistas e implementadores de LPs podem ter visões diferentes sobre quais critérios são mais importantes. Por exemplo, um programador pode valorizar mais a facilidade de escrita, enquanto um projetista pode valorizar mais a confiabilidade do código.