Linguagens de Programação

Thiago Figueiredo Marcos

30 de maio de 2024

Resumo

Essa disciplina será baseada no livro: `Conceito de Linguagens de Programação` de **Robert Sebesta**. Utilizei a bibliteca digital da UFPR para acessar o exemplar, além de vídeos aulas do Prof. Dr. Filipe Braida no youtube.

1 Capitulo 1 - Preliminares

O autor relata diversos motivos pelos quais um estudante de ciência da computação deveria se debruçar sobre conceitos gerais de linguagens de programação, entre eles, a capacidade de expressar a complexidade dos pensamentos e também o maior acesso a uma diferenciada gama de ferramentas para resolver diversos tipos de problemas diferentes.

1.1 Domínios de programação

Aqui o autor relata as diversidades das aplicações dos computadores modernos, e como as linguagens de programação podem ser construidas por motivos e aplicabilidades diferentes. Deixo abaixo alguns destaques:

1. Aplicações Científicas
2. Inteligência Artificial
3. Softwares para a WEB

1.2 Critérios de avaliação de LPs

Nesta seção é levantado alguns critérios genéricos para avaliar uma LP, essas características serão análisadas com maior profundidade mais para frente, de momento, será passado um entendimento geral para caracterizar uma determinada linguagem.

A facilidade com o qual é feita a manutenção em um software é determinada pela legibilidade do código, e esta, se torna uma medida importante para determinar a qualidade do programa e da linguagem.

Os critérios que são levados em conta ao caracterizar uma linguagem são os seguintes:

1. Legibilidade

2. Facilidade de escrita
3. Confiabilidade
4. Custo

Esses valores não necessáriamente é posto em grau de importância, mas sim de consenso para avaliação.

A simplicidade geral da linguagem também afeta sua legibilidade, uma linguagem com muitos operadores ou ainda, com diversas maneiras de fazer uma operação, pode ternala uma linguagem de pouca legibilidade. A sobrecarga de operadores, ou seja, multiplos significados em um único operador reduz a legibilidade, porém nem sempre a simplicidade pode contribuir com a legibilidade dos programas, um exemplo é a linguagem assembly.

A ortogonalidade também está relacionada à simplicidade, quanto mais ortogonal for uma linguagem maior as combinações e manipulações dos tipos primitivos e menor o numero de exceções ás regras da linguagem.

Por fim, o estudo de linguagens de programação aumenta nossa capacidade de usarmos diferentes construções ao escrevermos programas, nos permite escolher uma linguagem com menores chances de maiores custos, além de facilitar o aprendizaddo de novas linguagens.

2 Nomes, vinculações e escopos

Aqui discutiremos as adversidades das variaveis em seus diferentes escopos, onde e como essas variáveis são alocadas, além do tempo de vida.

As linguagens imperativas, ou seja, aquelas que são executadas linha a linha, carregam em sí a arquitetura de `Von Neumann`, que consiste em armazenar além dos dados as instruções do programa para posteriormente serem fornecidas ao processador.

Uma célula de memoria é um bloco fundamental da memoria do computador, ou seja, um circuito eletrônico que tem a capacidade de armazenar bits, `isso é visto com mais detalhes na disciplina de circuitos digitais` em um programa, nós ciêntistas da computação, abstraimos esse circuito e a chamamos de váriavel. Em uma arquitetura convencional de 64 bits um `inteiro`, tem, geralmente, 4 bits. Em um alto nivel de abstração das células de memória um tipo de dado `inteiro` poderia armazenar 4 `char`, por isso uma variável possui propiedades e atributos para serem caracterizadas e distinguidas, ainda assim, em baixo nível, essas atribuições podem ocorrrer, causando falhas no programa, sem que o compilador reconheça.

2.1 Nomes

Aqui vou pular a parte de nomeação de variáveis, pois foi exaustivamente tratada na disciplina de introdução a ciência da computação. Nos restringiremos a tratar das palavras reservadas e o

significado delas.

Uma palavra reservada em uma linguagem de programação é uma palavra `especial`, que não pode ser usada como nomes, logo, se uma linguagem possui um número alto de palavras reservadas, o programador terá que usar da criatividade para atribuir nomes.

2.2 Endereços

O endereço de uma variável, é o endereço de cécula de memória na qual ela está associada. Em linguagens de programação o endereço da variável é geralmente associado ao valor esquerdo de uma sentença, lembrando que, é possivel que tenha tipos de variáveis que podem ser associadas a um mesmo endereço de memória. Quando mais de uma variável é usada para acessar um único endereço, é necessário muito cuidado, pois, os valores podem ser alterados por uma atribuição, o que também pode prejudicar a legibilidade e manutençaõ do código. O momento no qual uma variável se associa a um endereço é muito importante e diz muito sobre a linguagem usada.

2.3 Tipos e Valores

O tipo de valor de uma variável, determina a faixa de valores que ela pode armazenar e o conjunto de operações que podem ser realizadas para esse tipo. Já o valor de uma variável se refere a o conteúdo da célula de memória. Uma célula de memória, pode ter um espaço pré-definido, como por exemplo 1 byte, porém, são valores que podem não caber determinado dado, que pode ter mais que 1 byte, logo, é conveniente, que pensemos em células de memória, como espaços abstratos, que por sua vez, se adapta ao tamanho do dado.

O valor da variável em muitos casos é chamado de lado direito.

2.4 Vinculações

Vinculação é uma associação entre um atributo e uma entiedade, para exemplificar pense no simbolo da operação de soma, `'automaticamente'`, sabemos que é o simbolo de `( + )`, esse é um tipo de vinculação, outro exemplo seria uma declaração de tipo em C, como por exemplo **int** a, sabemos que a variável a está associada a o tipo de dado inteiro, que por sua vez, está vinculado a uma faixa de valores.

A grande questão é quando essas vinculações ocorrem, na sentença: **int** a o valor **int** será associado a uma faixa de valores em tempo de projeto da linguagem, ou seja quando estivermos projetando a linguagem, reservaremos a palavra **int** e a associaremos uma faixa de valor. Já o valor **a**, é associado a o valor inteiro em tempo de compilação, no caso da linguagem C.

O entendimento completo de tempos de vinculação é fundamental para um real entendimento do funcionamento semântico de uma linguagem de programação.

2.4.1 Vinculação Dinâmica

Este tipo de vinculação é feito quando uma célula de memória, recebe um dado, o tipo dado determinará o tipo da váriavel, ou seja, o tipo variável é determinada pelo seu valor.

2.4.2 Vinculação Estática

Este tipo de vinculação é feito quando determinamos o tipo de dado que uma célula de memória vai receber, ou seja, o tipo da variável é determinado antes de receber o dado.

A principal diferença entre elas é que a dinâmica da mais flexibilidade para o programador, enquanto a estática, define a variável aquele tipo e não pode mais ser alterada em tempo de execução, o que pode, evitar falhas no programa.

3 Tempo de vida de variáveis

Uma variável deve ser associada a uma céula de memória, essa célula deve ser obtida de um conjunto de células disponíveis, esse processo é chamado de `alocação`, quando disponibilizamos novamente essa célula para o conjunto de células disponiveis, esse processo chamamos de `liberação`. Dizemos que o tempo de vida de uma variável é a diferença entre o tempo que ela foi liberada e alocada, ou seja, enquanto ela está vinculada a uma célula de memória.

Existem diversos tipos de vinculações e elas podem ocorrer em diferentes tempos, como por exemplo as variáveis `estáticas`, que são vinculadas a uma célula de memória antes do inicio da execução do programa, ou seja, em tempo de compilação, isso pode gerar vantagens e desvantagens. Uma vantagem seria uma menor sobrecarga de tempo para alocações de memória em tempo de execução, uma desvantagem muito grande é a impossibilidade de criar funções recursivas.

Existem outras variáveis que chamamos de dinâmicas, pois são vinculadas em tempo de execução e só podem ser referênciadas por ponteiros. Esses tipos de variáveis devem ser alocadas e liberadas por instruções explicitas e em tempo de execução, em C por exemplo, temos a função `malloc` que faz esse tipo de alocação. `Reforçamos que é dever do programador liberar a memória alocada depois de alocada.`

O principal problema em usar esse tipo de alocação é o cuidado com o gerenciamento de memória da `heap` que se não for administrado da forma correta, podem ocasionar `overflow` entre outros erros.

4 Escopos

Um dos fatores para o entendimento das variáveis é o escopo. Uma variável é visível se ela pode ser referenciada ou atribuida em sentenças.

Muitas linguagens permitem que sejam criados novos escopos no meio do código, esse escopo permite que tenha suas próprias variáveis locais, cujo escopo é restrito no bloco. Uma variável é local quando ela é alocada em um escopo.

4.1 Escopo Estático

O escopo estático é caracteristica comum de determinadas linguagens imperativas, como C e Pascal. Existem dois tipos de escopos estáticos: Aqueles nos quais podem se criar funções dentro de funções e as que não podem. Outra caracteristica desse tipo de escopo é o acesso a variáveis, que deve ser feito considerando o escopo `(Global ou Local)`. Também deve-se tomar cuidado com a substituição quando possuem o mesmo nome. Aqui também tem a questão do custo para acessar variáveis de escopos distantes, pois, só será possivel passando a variável por referência, o que pode gerar algum custo, se o escopo estiver muito longe da raiz.

4.2 Escopo Dinâmico

A principal diferença com o escopo estático é que no escopo dinâmico quando um subprograma é acionado, todos os seus descendentes também terão suas variáveis possiveis de acesso, isso pode gerar facilidades como também complicações, a variável é definida junto com seu escopo em tempo de execução. A variável será acessivel a qualquer outro subprograma, independente da proximidade dos subprogramas na árvore de chamadas.