SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Lógica de Programação e Algoritmos

Contexto:

Os alunos serão introduzidos ao universo da programação com Python, aprendendo conceitos fundamentais como lista de instruções (IL), arquivos-fonte e os elementos básicos de linguagens de programação. Explorar como os programas de computador funcionam, transformando máquinas em ferramentas úteis através de instruções simples. Será abordado diferença entre linguagens naturais e de programação, destacando como a comunicação com computadores é estruturada. Discutiremos também a importância do alfabeto, léxico, sintaxe e semântica em linguagens, e a distinção entre linguagens de máquina e de alto nível.

1 - Introdução a Programação

Começaremos aprendendo sobre alguns dos conceitos universais de programação, como lista de instruções, arquivo de origem, elementos de linguagem, compilação e interpretação.

2 - Como funciona um programa de computador?

Um programa torna um computador utilizável. Sem um programa, um computador, mesmo o mais poderoso, não passa de um objeto. Da mesma forma, sem um jogador, um piano nada mais é do que uma caixa de madeira.

Os computadores são capazes de executar tarefas muito complexas, mas essa capacidade não é inata. A natureza de um computador é bem diferente. Ele pode executar apenas operações extremamente simples. Por exemplo, um computador não consegue entender sozinho o valor de uma função matemática complicada, embora isso não esteja além dos limites das possibilidades no futuro próximo.

Os computadores contemporâneos só podem avaliar os resultados de operações muito fundamentais, como adicionar ou dividir, mas podem fazer isso com muita rapidez e podem repetir essas ações praticamente várias vezes.

Imagine que você quer saber a velocidade média que atingiu durante uma longa jornada. Você sabe a distância, você sabe o tempo, você precisa da velocidade.

Naturalmente, o computador poderá calcular isso, mas não está ciente de coisas como distância, velocidade ou tempo. Portanto, é necessário instruir o computador para:

- aceitar um número que represente a distância;
- aceitar um número que represente o tempo de viagem;
- dividir o valor anterior pelo último e armazenar o resultado na memória;
- exibir o resultado (representando a velocidade média) em um formato legível.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Essas quatro ações simples formam um programa. Obviamente, esses exemplos não são formalizados e estão muito longe do que o computador pode entender, mas são bons o suficiente para serem traduzidos para um idioma que o computador possa aceitar. Linguagem é a palavra-chave.

3 - Linguagens naturais vs. linguagens de programação

Uma linguagem é um meio (e uma ferramenta) para expressar e registrar pensamentos. Há muitos idiomas ao nosso redor. Alguns deles não exigem fala nem escrita, como a linguagem corporal; é possível expressar seus sentimentos mais profundos sem dizer uma palavra.

Outra linguagem que você usa todos os dias é a sua língua nativa, que você usa para manifestar sua vontade e refletir sobre a realidade. Os computadores também têm sua própria linguagem, chamada de linguagem de máquina, o que é muito rudimentar.

Um computador, mesmo o mais tecnicamente sofisticado, não tem nem um traço de inteligência. Você pode dizer que é como um cão bem treinado - responde apenas a um conjunto predeterminado de comandos conhecidos.

Os comandos que ele reconhece são muito simples. Podemos imaginar que o computador responde a pedidos como "pegue esse número, divida por outro e salve o resultado".

Um conjunto completo de comandos conhecidos é chamado de lista de instruções, às vezes abreviada para ***IL*** (***lista de instruções***). Diferentes tipos de computadores podem variar de acordo com o tamanho das ILs, e as instruções podem ser completamente diferentes em modelos diferentes.

Nenhum computador é capaz de criar um novo idioma no momento. No entanto, isso pode mudar em breve. Assim como as pessoas usam vários idiomas diferentes, as máquinas também têm muitos idiomas diferentes. A diferença, no entanto, é que as linguagens humanas se desenvolveram naturalmente.

Além disso, eles ainda estão evoluindo, e novas palavras são criadas todos os dias à medida que palavras antigas desaparecem. Esses idiomas são chamados de linguagens naturais.

4 - O que faz uma linguagem?

Podemos dizer que cada linguagem (máquina ou natural, não importa) consiste nos seguintes elementos:

- ***um alfabeto***, um conjunto de símbolos usados para criar palavras de um determinado idioma (por exemplo, o alfabeto latino para inglês, o alfabeto cirílico para russo, o kanji para japonês e assim por diante);
- ***um léxico*** (também conhecido como dicionário), um conjunto de palavras que o idioma oferece aos usuários (por exemplo, a palavra "computador" vem do dicionário de português,

enquanto "cmoptrue" não; o termo "chat" está presente nos dicionários de inglês e francês, mas com significados diferentes);

- ***uma sintaxe***, um conjunto de regras (formais ou informais, escritas ou intuitivamente) usadas para determinar se uma determinada sequência de palavras forma uma sentença válida (por exemplo, "eu sou uma píton" é uma frase sintaticamente correta, enquanto "eu uma píton estou" não é);
- ***semântica***, um conjunto de regras que determina se uma determinada frase faz sentido (por exemplo, "comi uma rosquinha" faz sentido, mas "uma rosquinha me comeu" não faz).

5 - Linguagem da máquina vs. linguagem de alto nível

A IL (*lista de instruções*) é, na verdade, o alfabeto de uma linguagem de máquina. Este é o conjunto de símbolos mais simples e primário que podemos usar para dar comandos a um computador. É a língua nativa do computador.

Infelizmente, essa língua nativa está muito longe de ser a língua nativa humana. Nós dois (computadores e seres humanos) precisamos de algo mais, uma linguagem comum para computadores e seres humanos, ou uma ponte entre os dois mundos diferentes.

Precisamos de uma linguagem na qual os humanos possam escrever seus programas e de uma linguagem que os computadores possam usar para executar os programas, uma linguagem que é muito mais complexa do que a linguagem de máquina e, no entanto, muito mais simples do que a linguagem natural.

Essas linguagens são frequentemente chamadas de linguagens de programação de alto nível. Eles são pelo menos um pouco semelhantes aos naturais, pois usam símbolos, palavras e convenções legíveis para os seres humanos. Essas linguagens permitem que seres humanos expressem comandos para computadores que são muito mais complexos do que aqueles oferecidos por ILs. Um programa escrito em uma linguagem de programação de alto nível é chamado de código-fonte (em contraste com o código de máquina executado por computadores). Da mesma forma, o arquivo que contém o código-fonte é chamado de arquivo-fonte.

6 - Compilação vs. interpretação

A programação de computadores é o ato de compor os elementos da linguagem de programação selecionada na ordem que causará o efeito desejado. O efeito pode ser diferente em cada caso específico - depende da imaginação, do conhecimento e da experiência do programador.

Obviamente, essa composição precisa ser correta em muitos sentidos:

- **em ordem alfabética** - um programa precisa ser escrito em um script reconhecível, como romano, cirílico etc.;

- **lexicamente** - cada linguagem de programação tem seu dicionário e você precisa dominá-lo; felizmente, é muito mais simples e menor do que o dicionário de qualquer linguagem natural;
- **sintaticamente** - cada idioma tem suas regras e elas devem ser obedecidas;
- **semanticamente** - o programa tem que fazer sentido.

Infelizmente, um programador também pode cometer erros com cada um dos quatro sentidos acima. Cada um deles pode tornar o programa completamente inútil.

Vamos supor que você tenha escrito um programa com sucesso. Como convencer o computador a executá-lo? Você precisa tornar o programa em linguagem de máquina. Felizmente, a tradução pode ser feita por um computador, tornando todo o processo rápido e eficiente.

Há duas maneiras diferentes de transformar um programa de uma linguagem de programação de alto nível em linguagem de máquina:

- **Compilação:** o programa de origem é convertido uma vez (no entanto, esse ato deve ser repetido cada vez que você modifica o código-fonte) ao obter um arquivo (por exemplo, um arquivo .exe se o código for executado no MS Windows) contendo a linguagem de máquina. Agora você pode distribuir o arquivo em todo o mundo; o programa que executa essa conversão é chamado de compilador ou tradutor.
- **Interpretação:** você (ou qualquer usuário do código) pode traduzir o programa de origem toda vez que ele tiver que ser executado. O programa que executa esse tipo de transformação é chamado de interpretador, pois interpreta o código toda vez que se destina a ser executado. Isso também significa que você não pode apenas distribuir o código-fonte como está, porque o usuário final também precisa que o intérprete o execute.

Devido a algumas razões fundamentais, uma determinada linguagem de programação de alto nível foi projetada para se enquadrar em uma dessas duas categorias.

Existem muito poucas linguagens que podem ser compiladas e interpretadas. Normalmente, uma linguagem de programação é projetada com esse fator na mente de seus construtores - será compilada ou interpretada?

7 - O que o interpretador faz?

Vamos supor mais uma vez que você escreveu um programa. Agora, ele existe como um arquivo de computador: um programa de computador é na verdade um pedaço de texto, então o código-fonte geralmente é colocado em arquivos de texto.

Observação: ele precisa ser texto puro, sem nenhuma decoração, como fontes diferentes, cores, imagens incorporadas ou outras mídias. Agora você precisa chamar o intérprete e permitir que ele leia o arquivo de origem.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

O intérprete lê o código-fonte de uma forma comum na cultura ocidental: da parte superior para a inferior e da esquerda para a direita. Há algumas exceções - elas serão abordadas mais adiante neste curso.

Primeiro de tudo, o intérprete verifica se todas as linhas subsequentes estão corretas (usando os quatro aspectos abordados anteriormente).

Se o compilador encontrar um erro, ele terminará o trabalho imediatamente. O único resultado nesse caso é uma mensagem de erro.

O conceito de Interpretação

O intérprete informará onde o erro está localizado e o que o causou. No entanto, essas mensagens podem ser enganosas, pois o intérprete não é capaz de seguir suas intenções exatas e pode detectar erros a alguma distância de suas causas reais.

Por exemplo, se você tentar usar uma entidade de nome desconhecido, ela causará um erro, mas o erro será descoberto no local em que tenta usar a entidade, e não onde o nome da nova entidade foi introduzido.

Em outras palavras, o motivo real geralmente está localizado um pouco mais cedo no código, por exemplo, no lugar onde você tinha que informar o intérprete que você usaria a entidade do nome.

Se a linha parece boa, o intérprete tenta executá-la (observação: cada linha é geralmente executada separadamente, para que o "leitura-verificação-execute" do grupo possa ser repetido muitas vezes - mais vezes do que o número real de linhas no arquivo de origem, pois algumas partes do código podem ser executadas mais de uma vez).

Também é possível que uma parte significativa do código seja executada com sucesso antes que o intérprete encontre um erro. Esse é o comportamento normal neste modelo de execução.

Você pode perguntar agora: qual é o melhor? O modelo de "compilação" ou o modelo de "interpretação"? Não há uma resposta óbvia. Se houvesse, um desses modelos teria deixado de existir há muito tempo. Ambos têm suas vantagens e desvantagens.

8 - Compilação vs. Interpretação - Vantagens e Desvantagens

Compilação (Vantagens):

- a execução do código convertido é mais rápida;
- apenas o usuário precisa ter o compilador - o usuário final pode usar o código sem ele;
- o código traduzido é armazenado usando linguagem de máquina - como é muito difícil de entender, suas próprias invenções e truques de programação provavelmente permanecerão em segredo.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Interpretação (Vantagens):

- você pode executar o código assim que tiver concluído - não há fases adicionais de conversão;
- o código é armazenado usando linguagem de programação, e não linguagem de máquina - isso significa que pode ser executado em computadores que usam linguagens de máquina diferentes; você não compila seu código separadamente para cada arquitetura.

Compilação (Desvantagens):

- a compilação em si pode ser um processo muito demorado - talvez você não consiga executar o código imediatamente após fazer uma alteração;
- você precisa ter tantos compiladores quanto plataformas de hardware nas quais deseja que seu código seja executado.

Interpretação (Desvantagens):

- não espere que a interpretação aumente seu código a alta velocidade - seu código vai compartilhar a capacidade do computador com o intérprete, então não pode ser muito rápido;
- você e o usuário final precisam ter o intérprete para executar o código.

Python - como você deve ter certeza - é uma linguagem interpretada. Isso significa que ele herda todas as vantagens e desvantagens descritas. Obviamente, ele adiciona alguns de seus recursos exclusivos aos dois conjuntos. Se você quiser programar em Python, precisará do interpretador Python. Você não poderá executar seu código sem ele. Felizmente, o Python é gratuito. Essa é uma das vantagens mais importantes. Devido a razões históricas, as linguagens projetadas para serem utilizadas na maneira de interpretação são frequentemente chamadas de linguagens de script, enquanto os programas de origem codificados usando-as são chamados de scripts.

Professor: Daniel Tadeu Petinice
E-mail: daniel.petinice@sp.senai.br

Referências:

1. CISCO ”Skills For All. (s.d.). Fundamentos do Python 1”. Disponível em: https://skillsforall.com/. Acesso em: 01/03/2024.