FUNDAMENTOS DA COMPUTAÇÃO

Prof. Dr. Ruy Ferreira (ruy@ufmt.br / prof_ruy@hotmail.com / @profruy)

Texto produzido para apoiar o ensino da disciplina-título.

LINGUAGEM DE PROGRAMAÇÃO DE ALTO NÍVEL E DE MONTAGEM

Conceitos fundamentais da programação

Para estudar as linguagens de programação é necessária uma abordagem inicial, situada entre uma revisão de conceitos e uma tentativa de classificação das linguagens de programação. Para isso é fundamental entender alguns de seus conceitos básicos.

A automatização de tarefas é um aspecto marcante da sociedade moderna. O aperfeiçoamento tecnológico alcançado, com respeito a isto, teve como elementos fundamentais a análise e a obtenção de descrições da execução de tarefas em termos de ações simples o suficiente, tal que pudessem ser automatizadas por uma máquina especialmente desenvolvida para este fim: o **computador**.

Esta especificação dos passos a serem seguidos e suas regras é dado o nome de **algoritmo**, ou seja, para que o computador possa realizar um determinado trabalho, será necessário que este seja detalhado passo a passo, através de uma forma compreensível pela máquina para ser empregado no que chamamos de **programa**.

A programação de computadores segue uma **lógica** de programação que permite definir a **sequência** lógica para o desenvolvimento do programa. Mas, o que é lógica? É o conjunto de leis, princípios ou métodos que determinam um raciocínio coerente, induzindo a uma solução prática e eficaz do programa, com uma sintaxe definida.

Entende-se por "Seqüência Lógica" os passos executados até atingir um objetivo ou solução de um problema. Cada passo é uma instrução que o computador deve realizar. Assim, **instrução** é o que indica a um computador uma ação elementar a ser executada. Logo, para obtermos o resultado esperado, é preciso colocar em prática o conjunto de todas as instruções, na ordem correta.

Ao conjunto de todas as instruções colocadas em sequência lógica chamamos de **algoritmo**. Formalmente é uma seqüência finita de passos que levam a execução de uma tarefa. Uma receita de bolo é um algoritmo, assim como um manual de utilização de uma máquina de lavar roupa. Em computação usamos algoritmos para representar a

solução de tarefas, como modelo para programas ou como receita para um processo computacional.

Em um algoritmo manipulamos objetos por meio de uma série de operações primitivas (comandos), tais objetos modificáveis são chamados de **variáveis**. Cada algoritmo possui seu próprio nível de **abstração**, podendo destacar somente propriedades relevantes ou ir a detalhes ínfimos da solução do problema.

Com base em um algoritmo escrevemos os programas de computador. Por analogia podemos pensar o algoritmo como uma maquete de um edifício e o programa como as plantas dela originadas. Os **programas** de computadores nada mais são do que algoritmos escritos numa **linguagem de computador** (Pascal, C++, Cobol, Fortran, Visual Basic, etc.) e que são **interpretados e executados** por uma máquina.

Pode-se dizer que o programa é um conjunto de instruções, escritas em uma linguagem computacional qualquer que será interpretado e executado pela máquina visando solucionar um problema.

Linguagem de descrição de algoritmo

Para escrevermos algoritmos é preciso uma linguagem clara e que não deixe margem a ambiguidades, para isto, se deve definir uma sintaxe e uma semântica, de forma a permitir uma única interpretação das instruções quando escritas em um algoritmo.

Normalmente os algoritmos possuem a seguinte estrutura:

Nome_Do_Algoritmo
Variáveis
Declaração das variáveis
Procedimentos
Declaração dos procedimentos
Funções
Declaração das funções
Início
Corpo do Algoritmo
Fim

Os algoritmos podem ser escritos em qualquer língua, seguindo as regras de construção de uma dessas abordagens: 1) utilizar o método cartesiano quando a complexidade (variedade) não estiver totalmente absorvida; 2) aplicar o planejamento

reverso: a partir das saídas, desagregar, desmontando a informação buscando atingir os dados de entrada obtendo as ações; 3) montar uma tabela de decisão quando uma ou mais ações dependentes de um conjunto de condições assumirem determinadas combinações de valores.

Para o desenho de algoritmos o que mais se utiliza são os diagramas de fluxos. Eles são representações gráficas que mediante o uso de símbolos unidos mediante linhas de fluxo, mostram a sequência lógica que se deve suceder para a solução do problema. Uma vez elaborado o algoritmo é hora de codificá-lo em uma linguagem de programação qualquer.

Programação

Programa

É o conjunto de instruções entendíveis pelo computador que permitem realizar um trabalho ou resolver um problema. Um programa deve ser **finito**, ou seja, tem que ter um início e um fim. Tem que estar bem escrito para que, ao introduzir um dado, saia uma solução e caso se voltasse a introduzir o mesmo dado, saísse de novo a mesma solução.

Metodologia da programação

Entende-se como **metodologia da programação** ao conjunto de normas, métodos e anotações que nos indicam a forma de programar. Cada linguagem de programação segue uma metodologia diferente, podendo existir linguagem que suporte mais de uma metodologia.

Existem várias **técnicas de programação**: Refinamento Sucessivo; Programação Estruturada; Programação Orientada para Objetos; Programação Modular; etc. Cada uma delas segue um **modelo** ou método que orienta a forma de programar. Existem métodos preditivos e adaptativos de programação, como o método ágil; método iterativo; método do modelo em cascata; e a codificação denominada cowboy ou balburdia.

Linguagem de programação

É um conjunto de regras sintáticas e semânticas que os programadores usam para a codificação de instruções de um algoritmo, elaborando assim um programa de computador. Existem várias linguagens de programação.

As linguagens de programação podem ser divididas em várias categorias, para fins de estudo. Por exemplo:

- Imperativa - Linguagens projetadas em função da arquitetura de Von Neumann (Dados e programas são armazenados na memória). Os algoritmos são especificados em grandes detalhes e os comandos ou instruções devem estar em ordem de execução. Exemplo de linguagens imperativas: C, Pascal, Fortran, Algol, Cobol, etc.
- Funcionais - São baseadas em funções matemáticas. Existe aplicação de funções a determinados parâmetros. O tipo de dados básicos são as listas e os exemplos de linguagens funcionais são: ML, Miranda, Haskell, etc.
- Lógica - São baseadas em regras e símbolos. Os programas lógicos são declarativos, porque consistem em declarações em vez de atribuições e instruções de fluxo de controle. O exemplo dessa linguagem é o Prolog.
- Orientada a Objetos - Relação próxima com a imperativa, nela são usados conceitos de classes, objetos, herança, polimorfismo e encapsulamento. Exemplos de linguagens OO: Java, C+, SmallTalk, etc.

Uma linguagem pode ser implementada por meio de três métodos: **Compilação, Interpretação Pura, Implementação Híbrida**. A Compilação ocorre quando os programas são traduzidos para linguagem de máquina a qual pode ser executada diretamente no computador. Usa-se esse método em C, PASCAL, etc.

Na Interpretação Pura o interpretador executa diretamente as instruções do programa fonte, sem traduzir em linguagem de máquina. Isso é feito através de uma máquina virtual, cujo ciclo de execução entende os comandos da linguagem de alto nível. A execução desse tipo de implementação é muito lento, por isso seu uso é raro.

Por fim, alguns sistemas de implementação de linguagem estão situados entre os compiladores e os interpretadores puros, são os chamados interpretador híbrido. Eles traduzem os programas em uma linguagem intermediária, que é interpretada. Apesar de serem utilizadas também máquinas virtuais, esse método é mais rápido que a interpretação pura, pois as instruções da linguagem fonte são decodificadas somente uma vez. Exemplos de linguagens que utilizam esse método é JAVA e PERL.

Níveis de Linguagens de Programação

As linguagens de programação podem ser classificadas em níveis de linguagens, sendo que as linguagens de nível mais baixo estão mais próximas da linguagem interpretada pelo processador e mais distante das linguagens naturais.

Linguagem de Máquina

Na linguagem de máquina, a representação dos dados e das operações (instruções) que constituem um programa, é baseada no **sistema binário**, que é a forma compreendida e executada pelo hardware do sistema. Torna-se inviável escrever ou ler um programa codificado na forma de uma string de bits.

Linguagem Hexadecimal

Para simplificar a compreensão e a programação de computadores, foi adotada a notação **hexadecimal** para representar programas em linguagens de máquina. Mas a programação usando a linguagem hexadecimal continuou sendo impraticável para uso comercial.

Linguagem Assembly

A linguagem de máquina de cada processador é acompanhada de uma versão "legível" da linguagem de máquina que é a chamada linguagem simbólica **Assembly**. Simbólica, pois esta linguagem não é composta de números binários ou hexadecimais, como nas duas linguagens anteriores. A linguagem **Assembly** é na realidade uma versão legível da linguagem de máquina. Ela utiliza palavras abreviadas, ou mnemônicos, indicando a operação a ser realizada pelo processador. Exemplos de instruções Assembly:

MOV R1, R2

Mnemônico MOV (abreviação de MOVE) e dois registradores como parâmetro: R1 e R2. Quando o processador executa essa instrução, ele comanda o movimento do conteúdo de R2 para R1.

A passagem de um programa escrito em Assembly para a linguagem de máquina é quase sempre direta, não envolvendo muito processamento. Essa passagem é chamada de **Montagem**, e o programa que realiza esta operação é chamado de **montador** (**Assembler**). A linguagem Assembly é orientada para a máquina (ou melhor, para processador), é necessário conhecer a estrutura do processador para poder programar em Assembly. Essa linguagem utiliza instruções de baixo nível que operam diretamente com registros e memórias, ou seja, as instruções são diretamente executadas pelo processador.

São aplicações da Linguagem Assembly:

- Controle de processos com resposta em tempo real - Nesse tipo de aplicação, o processador deve executar um conjunto de instruções em um tempo limitado;
- Comunicação e transferência de dados - Nesse tipo de aplicação é utilizada a linguagem Assembly, devido à possibilidade de acessar diretamente o hardware;
- Otimização de sub-tarefas da programação de alto nível - Um programa não precisa somente ser escrito em linguagem Assembly ou linguagem de alto nível. Podemos ter programas de alto nível com sub-tarefas escritas em Assembly, para o caso de tarefas tempo-real ou para a programação do hardware do computador.

Linguagem de Alto Nível

As linguagens de alto nível são assim denominadas por apresentarem uma sintaxe mais próxima da linguagem natural, fazendo uso de palavras reservadas extraídas do vocabulário corrente (com READ, WRITE, TYPE, etc.) e permitem a manipulação dos dados nas mais diversas formas (números inteiros, reais, vetores, etc.), enquanto a linguagem Assembly trabalha com bits, bytes, palavras, armazenadas em memória.

A passagem de um programa escrito em linguagem de alto nível para o programa em linguagem de máquina é bem mais complexa comparado com a linguagem Assembly. Essa passagem é feita utilizando interpretadores, compiladores e linkadores. Um programa escrito em linguagem de alto nível pode, teoricamente, ser usado em qualquer máquina, bastando escolher o compilador correspondente, o que não acontece com um programa escrito em Assembly.

Linguagem Estruturada

Nessa classe estão as linguagens de programação de alto nível, criadas pela necessidade da produção de código de programa de forma clara, aparecendo o conceito de estruturação do código (endentação, utilização de letras maiúsculas ou minúsculas nos identificadores, eliminação de instruções com “go to”, etc.).

Entre as décadas de 60 a 80 surgiram diversas linguagens que podem ser organizadas da seguinte forma para fins de estudo:

- Linguagens de uso geral: Pascal, Modula-2 e C;
- Linguagens especializadas: Prolog, Lisp e Forth;
- Linguagens orientadas a objetos: Smalltalk, Eiffel, C++ e Delphi.

Conclusão

Foge ao escopo da disciplina o estudo de cada uma dessas linguagens de programação. Entretanto, substituindo uma pretensa conclusão é possível fechar essa nota de aula afirmando que o desenvolvimento de programas está associado ao uso de ferramentas ou ambiente de desenvolvimento que acompanham o programador desde a etapa de codificação até a geração e teste do código executável.

A seguir é apresentado as principais etapas de geração de um programa, além das ferramentas utilizadas.

- Geração do código fonte (codificação) - A codificação é escrita, utilizando uma linguagem de programação, com as instruções que o computador deve realizar para alcançar um resultado desejado. Para realização dessa tarefa são utilizados os chamados editores. O editor é a primeira ferramenta que o programador utiliza na etapa de codificação, pois é através dela que será gerado o arquivo (ou o conjunto de arquivos) que vai conter o código-fonte do programa. É possível utilizar qualquer editor para gerar o arquivo de programa. Hoje existem ambientes que oferecem ferramentas de edição mais poderosas que um simples editor.

    - Tradução do código-fonte (código-objeto) - Independente da linguagem de programação utilizada e da arquitetura do sistema computacional, o código-fonte não é executável diretamente pelo processador. O código-fonte permite apenas que o programador consiga definir o programa em uma forma legível aos humanos. Para que o programador consiga definir o programa executável, é necessário que o código-fonte seja traduzido para o código de máquina do processador que compõe a arquitetura do sistema. A tradução é feita de forma automática graças a existência de ferramentas como os **Montadores** (ou Assembler, para programas escritos em Assembly) e os **Compiladores**, construídos para gerar o código de programas escritos em linguagens de alto nível. O código gerado por essas ferramentas é representado dentro do sistema de numeração binária e é denominado **código-objeto**. Ele é o código produzido pelo compilador, é uma forma intermediária similar a linguagem de máquina do computador. Apesar de estar representado em binário, não é executável diretamente pelo processador, pois normalmente, o código-objeto referencia partes de programa que não estão necessariamente definidas no mesmo arquivo que o gerou, por exemplo, arquivos de bibliotecas de sub-rotinas.

    - Editores de ligação - A tarefa do editor de ligação ou linker é rearranjar o código do programa, incorporando a ele todas as partes referenciadas no código original, resultando num código executável pelo processador. Essa tarefa pode ser realizada também pelos chamados carregadores.

    - Depuradores ou debuggers - Os depuradores possuem uma função essencial que é auxiliar o programador a eliminar (ou reduzir) a quantidade de erros (bugs) de execução no programa. Possibilita uma análise efetiva do código devido à:

        - Execução passo-a-passo (instrução por instrução) de partes do programa;

- Visualização dos valores das variáveis e dos conteúdos dos registros internos do processador;
- Alteração em tempo de execução de conteúdos de memória, de variáveis ou de instruções; etc.

A evolução das linguagens de programação pode ser dividida em 5 etapas ou gerações.

- Primeira geração: Linguagem máquina.
- Segunda geração: Criaram-se as primeiras linguagens assembly (assembler).
- Terceira geração: Criam-se as primeiras linguagens de alto nível. Ex: C, Pascal, Cobol.
- Quarta geração: São linguagens capazes de gerar código por si só, são os chamados RAD, com o qual pode-se realizar aplicações sem ser um expert na linguagem. Aqui também se encontram as linguagens orientadas a objetos, tornando possível a reutilização de partes do código para outros programas. Ex: Visual, Natural Adabas.
- Quinta geração: São as linguagens orientadas à inteligência artificial. Estas linguagens ainda estão pouco desenvolvidas. Ex: LISP

Não existe uma linguagem universal, capaz de substituir todas. Cada tipo de problema requer uma linguagem específica. Logo, devemos nos acostumar a aprender a codificar em várias linguagens de programação..

REFERÊNCIAS

CAPRON, H. L.; Johnson, J. A. **Introdução à Informática**. 8ª Ed. São Paulo: Pearson Prentice Hall, 2004.

LEWIS, Harry R; Christos H. Papadimitriou. **Elementos de teoria da computação**. 2 ª Ed.Bookman,2000.

BROOKSHEAR, J. G**. Ciência da computação** – uma visão abrangente. Porto Alegre: Bookman, 2000.

VIEIRA, N. J. **Introdução aos fundamentos da computação**: linguagens de máquinas. São Paulo: Thompson Pioneira, 2006.