FIAP ON

LÓGICA DE PROGRAMAÇÃO

INTRODUÇÃO

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

Sumário

INTRODUÇÃO

1 HISTÓRIA DA PROGRAMAÇÃO

Para que entendamos os conceitos relacionados à história da programação, devemos primeiramente entender o termo “Tecnologia”. Tecnologia tem origem no grego “tekhene” que significa “Técnica, arte e ofício” juntamente com “logia” que significa “estudo”.

Assim, não devemos entender tecnologia como algo informatizado ou eletrônico, mas sim um processo que visa melhorar e agilizar tarefas para o ser humano. Por exemplo, temos as tecnologias primitivas como o cálculo, a escrita, a descoberta do fogo e a invenção da roda que ajudaram o ser humano em seu dia a dia a sobreviver, a se comunicar, a se locomover, entre outras tarefas do seu cotidiano.

Partindo deste conceito de tecnologia, Joseph-Mariae Jacquard criou o Tear de Jacquard, uma máquina desenvolvida em 1801 cujo objetivo era o de substituir o braço humano em algumas tarefas. Esta máquina mecânica era programada e sua tecnologia era emaranhar os fios para que fossem criadas as suas tranças.

A história da programação está relacionada com a história da computação. Ada Augusta Byron King (ou Ada Lovelace, nome da província onde morava) foi considerada a primeira programadora de computadores.

Ela pegou o computador criado por Charles Babbage, em 1822 – que nada mais era do que algo criado com madeira e latão para a realização de cálculos baseada nos cartões perfurados do tear mecânico. Ela escreveu um programa que poderia funcionar nesta máquina: o cálculo de sequência de Bernoulli. Este foi considerado o primeiro algoritmo criado e interpretado por uma máquina (ainda não eletrônica).

Os programas de computadores nada mais são que algoritmos que podem ser escritos numa linguagem específica e funcionar em um computador.

Na década de 40, as linguagens de programação começaram a se especificar (respeitando regras de sintaxes), sendo entendidas e executadas por um computador respeitando normas de semântica.

Na década de 50. as linguagens eram às de máquina para os primeiros computadores. Nessa época, as linguagens eram às de máquina para os primeiros computadores. Os programas eram binários e inseridos por cabos nos computadores. Eles notaram que codificação binária era de difícil compreensão, manipulação e pouco versátil, então criaram o Assembly.

Com o Assembly as instruções ficaram algorítmicas, ou seja, mais parecido com o que fazemos hoje. Na década de 50 houve uma mudança significativa nos conceitos das linguagens de programação, criou-se as linguagens de alto nível como Fortran (1954), Lisp (1958) e Cobol (1959). A sua aplicação inicial era mais governamental em supercomputadores.

Na década de 60 e 70, as linguagens foram evoluindo e o Algol tornou-se popular porque interpretava mais fielmente um algoritmo, tanto que a sua última versão foi ALGOL 68 que passou a ficar defasada com a criação do PASCAL.

PASCAL é uma linguagem estruturada de fácil compreensão, criada por Niklaus With em 1970. Tanto que passou a ser uma linguagem padrão para o aprendizado de lógica de programação por ter uma similaridade com o pseudocódigo.

Basic foi uma outra linguagem importante – criada em 1964 por Eugene Kurtz e George Kemeny, nos EUA – que hoje é a linguagem nativa do Visual Basic.

A linguagem de programação Simula – Criada em 1960 por Nygaard e Dahl – foi a primeira a aplicar conceitos de linguagem de programação orientada a objetos. Ela utilizava classe, herança dentre outros conceitos que utilizamos nas linguagens de alto nível atuais.

Na década de 70 foram criadas as linguagens nativas das linguagens de programação que temos hoje. A linguagem C – criada por Dennis Ritchie e Ken Thompson em 1972 – hoje é linguagem dativa do C++, C#, JavaScript, Java e PHP, por exemplo. O Basic é linguagem nativa do Visual Basic e o Pascal do Delphi.

A linguagem Prolog (1972) era logico-interativa. O seu conceito de construção do programa era diferente dos demais. Enquanto as outras linguagens utilizavam a metodologia algorítmica, Prolog usava o conceito de predicados, átomos, lógica e regras para aumentar a inteligência do programa.

Ainda hoje, ao explicarmos lógica de programação utilizando o Pseudocódigo, enxergamos o Pascal. O Pascal é uma linguagem de programação didática onde para todas as estruturas temos Início e Fim (BEGIN e END), conceito este que usamos no pseudocódigo para que o aprendiz saiba que tudo que for aberto deve ser fechado.

A primeira versão da linguagem C++ foi criada em 1985 por Bjarne Stroustrup. Basicamente pegou-se a linguagem nativa C e nela inseriu o conceito de Programação Orientada a Objetos (herança, classe e métodos).

A linguagem de programação Perl – criada por Larry Wall – foi a primeira linguagem de código aberto (open source). O seu conceito inicial era o de extração de relatórios complexos, mas por ter o seu código-aberto passou a ser mais explorada para outros fins. Extremamente versátil, ela funciona em diversos sistemas operacionais e conecta várias bases de dados diferentes.

Os anos 90 foi uma mudança no paradigma na programação. Com a popularização da internet surgiram linguagens para este fim. A linguagem de programação Java – criada por James Gosling em 1995 – é uma linguagem de programação multiplataforma; todo código escrito em Java pertence a uma classe e dela surgem objetos.

Também em 1995 surgiu a linguagem de programação JavaScript, criada por Brendan Eich. A Netscape – empresa responsável pelo navegador mais potente da sua época – o contratou para desenvolver esta linguagem no modo cliente (bastando apenas ter um navegador, ainda que não tivesse conexão com a internet). Todos os navegadores a tem como padrão e atualmente pode ser utilizada fora do navegador via NODEJS, por exemplo. Apesar da similaridade do nome e data de criação, Java e JavaScript não são “irmãs”, ambas são orientadas a objeto, mas JavaScript não precisa rodar dentro de uma classe e o seu código é mais simples.

A linguagem de programação Lua foi criada em 1993 na PUC-Rio. Sim, ela é brasileira. Ela é uma linguagem de programação procedural e orientada a objetos. Hoje ela é voltada mais para o mercado de criação de games.

Ela é utilizada pela Roblox, uma das principais plataformas de games para IOS, Android, PCs e vídeo-games.

A linguagem de programação PHP – Criada por Rasmus Lerdorf em 1995 – é uma linguagem voltada para a internet. Ela é complementar ao Java Script, enquanto o JavaScript roda no Cliente, PHP roda no servidor. Ambas são programáveis dentro do HTML com as suas TAGS específicas.

Delphi é uma linguagem de programação orientada a objetos que tem o Pascal como linguagem nativa. Criada pela Borland em 1995, as suas principais características são: desenvolvimento multiplataforma, tem uma IDE bastante integrada, suporta arquitetura cliente-servidor e banco de dados SQL. Também suporta integralmente APIs do Windows.

A linguagem de programação Visual Basil (Principal concorrente do Delphi em seu auge) é uma linguagem de programação criada pela Microsoft em 2000. Hoje adota o sobrenome .NET e está disponível no Visual Studio. A sua linguagem nativa é o Basic. Como no Delphi a sua IDE é robusta, podemos fazer tudo por ela, como compilar, testar, depurar e arrastar componentes.

Com o passar do tempo as linguagens de programação vão evoluindo e ficando mais fácil de entender e programar. Saiba então que cada software, aplicativo, página da internet que você utiliza está escrito em alguma linguagem de programação.

2 COMO O COMPUTADOR ENTENDE UM PROGRAMA

As linguagens de programação são linguagens usadas para a comunicação com o computador. Estas linguagens são constituídas de comandos, que quando utilizados corretamente, executam ações.

As instruções colocadas em um programa são compiladas (ou interpretados) por uma linguagem de programação, gerando um arquivo objeto (linguagem de montagem). O linkeditor pega este arquivo objeto e transforma em um arquivo em linguagem de máquina (executável: 0 e 1), então o computador consegue interpretar e executar o programa.

A imagem abaixo ilustra este processo:

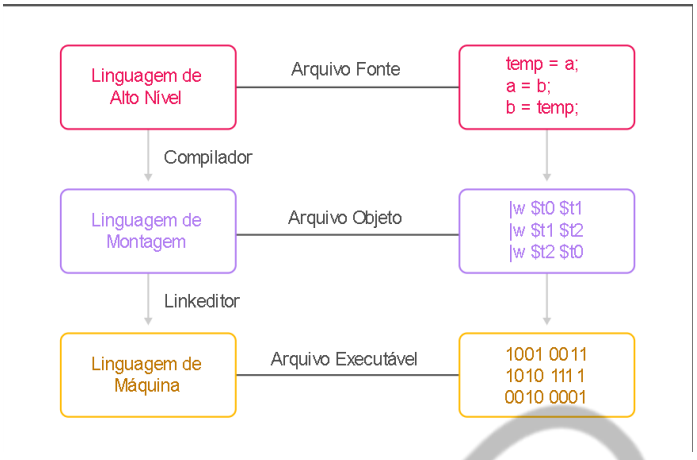

Figura 1 – Arquitetura de um programa em um computador
Fonte: Adaptado de FARIAS (s.d.)

A arquitetura do computador é lógica. Algo lógico é representado por dois estados opostos como ‘sim’ ou ‘não’, ‘verdade’ ou ‘falso’, ‘0’ ou 1. O ‘0’ e ‘1’ são notações do computador que representam ‘não passa corrente’ e ‘passa corrente’, respectivamente. Um programa deve respeitar este conceito para que as suas instruções sejam convertidas para uma linguagem de programação até a linguagem de máquina, para que sejam compreendidas pelo computador.

3 O QUE É UM ALGORITMO

Algoritmo é um termo usado em diversas disciplinas que não necessariamente são da área de programação (TI). É o processo pelo qual todo problema é submetido objetivando, por meio de passos – que respeitam uma ordem lógica – resolver o problema.

Na área de Lógica de Programação, o algoritmo tem o mesmo significado. Muitos confundem e pensam que algoritmos é apenas a solução final do problema, mas não. É a especificação dos passos em ordem lógica que visa resolver o problema proposto. Sem um problema não é possível desenvolver um algoritmo.

Exemplo: Resolução de um algoritmo do nosso cotidiano.

PROBLEMA 1 – Trocar uma lâmpada queimada:

- Pegar a lâmpada nova

- Colocar a escada debaixo da lâmpada queimada
- Subir na escada
- Retirar a lâmpada queimada
- Colocar a lâmpada nova
- Descer da escada
- Ligar o interruptor para testar a nova lâmpada

Os passos sugeridos foram colocados numa ordem lógica. Para que o algoritmo funcione adequadamente partimos do princípio de que o passo anterior obteve êxito, caso contrário há uma falha (bug) no algoritmo.

Neste exemplo podemos perceber que no passo “Colocar a escada debaixo da lâmpada queimada”, deveríamos verificar se a escada existe, foi encontrada ou está disponível, ou seja, este passo funcionará somente se a escada estiver disponível, caso contrário o algoritmo apontaria uma falha.

Partindo deste princípio, a segunda versão deste algoritmo ficaria:

- Pegar a lâmpada nova
- **Se não há uma escada disponível, providenciar**
- Colocar a escada debaixo da lâmpada queimada
- Subir na escada
- Retirar a lâmpada queimada
- Colocar a lâmpada nova
- Descer da escada
- Ligar o interruptor para testar a lâmpada nova

Colocando o passo “Se não há uma escada disponível, providenciar” resolveu o problema citado no parágrafo anterior, porém nos próximos testes poderemos encontrar outras falhas, tipo o “ligar o interruptor” estar condicionado à energia elétrica estar ligada na caixa de eletricidade do imóvel.

Este exemplo mostra que um algoritmo complexo sempre terá uma falha que deverá ser retificada uma hora ou outra. Da mesma forma funciona um programa.

Agora vamos usar um exemplo de algoritmo mais consistente na área de programação.

Exemplo: Resolução de um algoritmo aplicável no computador.

PROBLEMA 2 – Solicitar três notas ao usuário que executará o algoritmo e calcular a média simples:

- Ler a primeira nota
- Ler a segunda nota
- Ler a terceira nota
- Calcular a média simples
- Exibir a média

Assim como no exemplo anterior, se todos os passos obtiverem êxito, a execução do algoritmo funcionará normalmente.

Existem falhas neste algoritmo? Várias! A mais importante seria a de verificar se a nota fornecida é consistente, ou seja, estar entre zero e dez. Para resolvermos este problema, acrescentaríamos novos passos criando, assim, outras versões.

Outra informação importante que devemos considerar na construção de um algoritmo é saber que **nunca** em um algoritmo são colocados em valores (quem o digita é o usuário), apenas elencamos os passos necessários. Por exemplo, o passo correto “Ler a primeira nota” não poderia ser substituído por “Ler a primeira nota como sendo 9”. Perceba que quem constrói o passo do algoritmo é o programador e quem fornece o que é sugerido no passo é o usuário do algoritmo, logo quem acrescenta os valores são os programadores.

4 PSEUDOCÓDIGO E FLUXOGRAMA

Como vimos, um algoritmo é uma sequência de passos em ordem lógica que objetiva mostrar como resolvemos um problema.

O algoritmo tem formas diferentes de representação. Eles são representados por intermédio de descrição narrativa (ou narrativa), fluxograma (ou diagrama de blocos) e escrito em uma linguagem de programação escolhida.

O objetivo destas formas diferentes de representação primeiramente é para facilitar o aprendizado na construção do algoritmo, depois, uma ou outra destas formas de representação poderão servir como documentação do sistema.

Pseudocódigo: no processo de aprendizagem de programação, usamos uma linguagem de programação hipotética – normalmente em português para um melhor entendimento – para a padronização do algoritmo, esta linguagem se chama pseudocódigo. As regras, sintaxes e critérios empregados são similares aos das linguagens de programação reais com o objetivo de aprimorar a disciplina e a organização que uma linguagem de programação exige do aprendiz de programação.

Didaticamente para o aluno, o fato de a linguagem ser em português facilita o aprendizado do algoritmo, forçando o aprendiz a adquirir a disciplina necessária para aprender uma linguagem de programação real.

Estrutura de um pseudocódigo:

```
programa <nome_programa>
var
    <lista_variáveis> : <tipos>
inicio
    <corpo_do_algoritmo>
fim
```

Código-fonte 1 – Estrutura e sintaxe de um pseudocódigo
Fonte: Elaborado pelo autor (2022)

O pseudocódigo tem um corpo que é dividido em três partes:

Nome do programa: Local onde colocamos o nome que daremos ao algoritmo.

Ambiente de declaração de variáveis: Local onde declararemos as variáveis com os seus nomes e tipos.

Corpo do algoritmo: Local onde escreveremos as instruções do algoritmo.

Fluxograma: Para falar de fluxograma usaremos uma analogia. Imagine uma notícia qualquer em um site. As notícias (que representam o algoritmo) são compostas de textos e de imagens. Só de olharmos uma imagem (foto) geralmente sabemos do que se trata a notícia, ou seja, uma imagem dá uma visão geral. Para termos um detalhamento, teríamos que ler o texto da notícia. A leitura seria o pseudocódigo.

Da mesma forma funciona o fluxograma, ele seria a imagem do algoritmo, só de olhar temos uma visão macro do algoritmo, enquanto o pseudocódigo (ou o código-fonte aplicado em uma linguagem de programação) é o detalhamento de como foi desenvolvido o algoritmo.

O objetivo do fluxograma é o de desenhar a ideia do algoritmo, mostrando assim o fluxo (sentido) dado entre os processos.

Fluxograma é a forma de representação gráfica dos passos do algoritmo através de figuras que representam estas ações. Nesta forma de representação do algoritmo usamos regras para cada uma das figuras/instruções.

Vamos exemplificar estes conceitos:

Problema: Dado um número pelo usuário, verificar se ele é um número primo ou não.

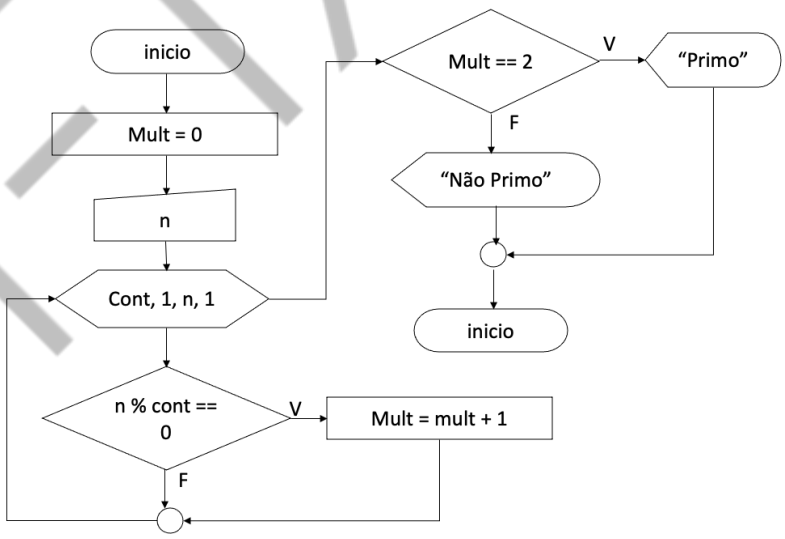

Figura 2 – Representação do algoritmo por meio de fluxograma
Fonte: Elaborado pelo autor (2022)

```
Programa exercicio_primo
var
    mult, n, cont: inteiro
inicio
    mult = 0
    leia n
    para cont de 1 até n inc 1 faça
        se n % cont == 0 então
            mult = mult + 1
        fim_se
    fim_para
    se mult == 2 então
        escreva "Primo"
    senão
        escreva "Não primo"
    fim_se
fim
```

Código-fonte 2 – Representação do algoritmo por meio de pseudocódigo
Fonte: Elaborado pelo autor (2022)

Não se preocupe se sabe ou compreendeu como foi feito este algoritmo, ensinaremos durante o curso. A ideia é apenas apresentar a você a transição do problema até a sua construção lógica, ou seja, como a partir de um problema proposto por intermédio de um texto, em uma situação interpretativa, nós conseguimos montar um algoritmo e representá-lo a partir do fluxograma e pseudocódigo para, a partir daí, expressá-lo em alguma linguagem de programação resolvendo assim o problema do usuário.

REFERÊNCIAS

FARIAS, Max Santana Rolenberg. Organização e Arquitetura de Computadores. **Univast**, s.d. Disponível em: <http://www.univasf.edu.br/~max.santana/material/aoc-i/Aula03-AOC-I.pdf>. Acesso em: 13 maio 2022.

História da Programação. Programador, 2013. Disponível em: <https://www.programador.com.br/historia-da-programacao.html>. Acesso em: 12 maio 2022.

GLOSSÁRIO

Bug	Falha na execução do algoritmo
Sintaxe	Regras para utilização de uma instrução do programa
Código-Fonte	Arquivo do tipo texto onde digitamos o programa
Arquivo-Objeto	Arquivo gerado após a compilação do código-fonte escrito sem erros de sintaxe
Arquivo-Executável	Arquivo que faz o programa funcionar (ser executado) para o usuário