UNIDADE I

Estudos Disciplinares
Lógica de Programação

Prof. Me. Marcelo Santos

Introdução

Programando um computador

- Arquitetura de computador

Código-fonte e código-objeto

- O que é o código-fonte?
- O que é código de objeto?
- Qual é a diferença entre código-fonte e código do objeto?

Fonte: https://pixabay.com/photos/coding-programming-css-1853305/

Programação em geral

- Linguagens formais e ambiente de programação
- Visão geral sobre os ambientes de desenvolvimento de *software*

Algoritmos

- Criação de Algoritmos
- Organização e Planejamento

Programando um computador

- Muitas pessoas pensam que a programação de computadores é uma arte misteriosa. Quando lhe perguntam como chegar a um local específico, você responde com um programa, mesmo que você provavelmente o chame de “instruções”.

- Suas instruções podem não funcionar porque você se esqueceu de uma curva ou não se deu conta de um beco que a pessoa iria encontrar no meio do caminho.

- Programadores de computador chamam esse tipo de erro de *bug*.

Fonte: https://pixabay.com/photos/computer-laptop-notebook-pen-table-2242264/

Programando um computador

- Uma partitura musical é outra forma de programa com símbolos especiais que devem ser colocados corretamente dentro de duas dimensões.

- Um programa de música (partitura) também pode ser convertido para algum formato mecânico ou eletrônico, para que os dispositivos, como teclados modernos, possam tocá-los automaticamente (e dispositivos como rolos de piano existem, inclusive, antes dos computadores eletrônicos serem inventados).

Fonte: https://pixabay.com/photos/keyboard-computer-keys-white-886462/

Programando um computador

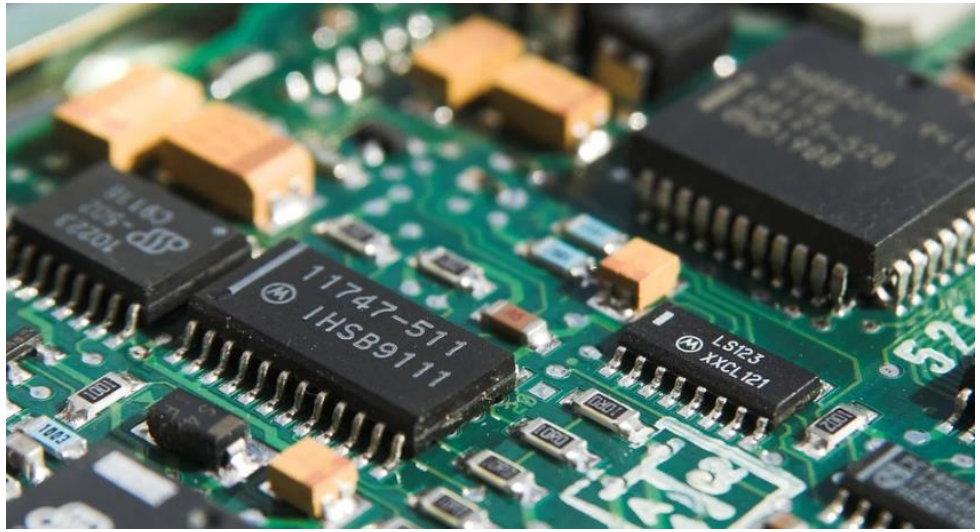
- A programação de computadores é apenas outra maneira de fornecer um conjunto exato de instruções para alcançar um determinado objetivo. Tenho certeza de que você seguiu e forneceu diversas instruções ao longo de sua vida.

Fonte: https://pixabay.com/photos/computer-pc-workplace-home-office-1185626/

Arquitetura de computador

- Se você olhar através da literatura da área da ciência da computação, parece complexo encontrar uma definição universal aceita do termo “arquitetura de computadores”. De acordo com Stallings (2010), os especialistas muitas vezes discordam sobre a diferenciação exata entre os termos “arquitetura de computadores” e “organização de computadores”.

- O termo "arquitetura" é usado para incluir a arquitetura do conjunto de instruções (a abstração do programador de computador), organização ou microarquitetura (a estrutura interna e a implementação de um computador no registro e na unidade funcional) e arquitetura do sistema (a organização do computador a partir da memória cache e no nível do barramento).

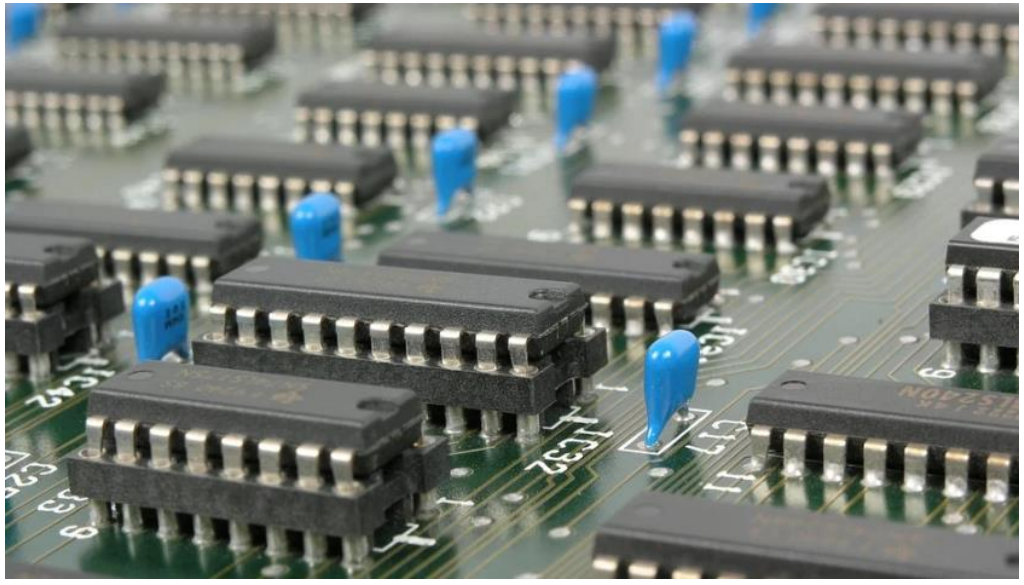


Fonte: https://pixabay.com/photos/cyber-security-network-internet-2377718/

Arquitetura de computador

- Com base nessa proposta, o termo "arquitetura de computadores" abrange todos os aspectos de um computador que você deve conhecer para entender como um computador executa um programa. Portanto, a arquitetura de computador inclui as seguintes áreas:

  - os componentes físicos fundamentais que constituem um computador e o seu sistema (o *hardware*)
  - o tipo de instruções/idioma que o computador pode entender
  - a tecnologia de computador subjacente que manipula essas instruções (às vezes chamadas de microarquitetura)



Fonte: https://pixabay.com/photos/mother-board-electronics-computer-581597/

Arquitetura de computador

- As melhorias na tecnologia de computadores têm sido tremendas desde que as primeiras máquinas apareceram. Um computador pessoal que pode ser comprado hoje por um valor acessível tem mais desempenho (em termos de, digamos, multiplicações de ponto flutuante por segundo), mais memória principal e mais capacidade de disco do que uma máquina que custava milhões nos anos 50-60. Podemos relacionar quatro linhas de evolução que surgiram dos primeiros computadores:

- ***Mainframes***: computadores grandes que podem suportar muitos usuários enquanto oferecem um grande poder de computação. É principalmente em *mainframes* que a maioria das inovações (tanto na arquitetura quanto na organização) foi feita.

Fonte: https://www.shutterstock.com/image-photo/mainframe-computer-94202284?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fsearch%2Fmainframes%2F

Arquitetura de computador

- **Minicomputadores**: adotaram muitas das técnicas do *mainframe*, ainda sendo projetadas para vender por um valor menor, satisfazendo as necessidades de computação para grupos menores de usuários. É o grupo de minicomputadores que apresentou um ritmo mais rápido de evolução (desde 1965 quando a DEC introduziu o primeiro minicomputador, PDP-8), principalmente devido à evolução da tecnologia de circuitos integrados (o primeiro CI apareceu em 1958).

- **Supercomputadores**: projetados para aplicações científicas, apresentam o custo mais caro (mais de um milhão de dólares). O processamento geralmente é feito no modo *batch*, por razões de desempenho.

Arquitetura de computador

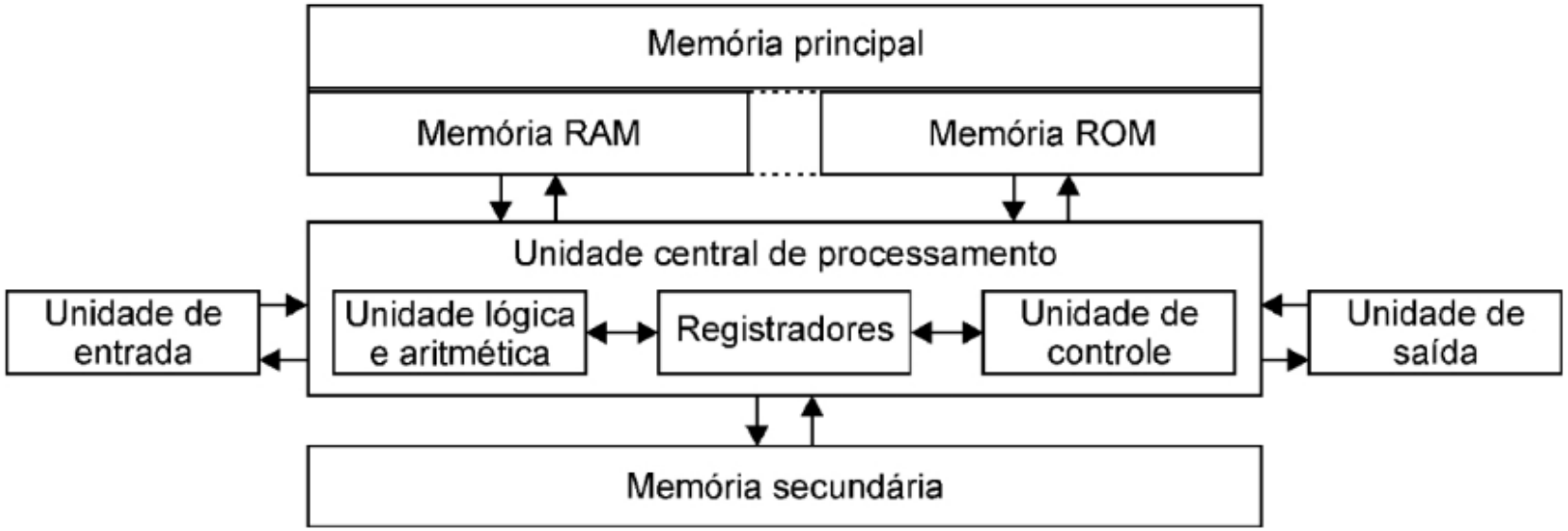
- **Microcomputadores**: apareceram na era do microprocessador (o primeiro microprocessador, Intel 4004, foi lançado em 1971). O termo micro refere-se apenas a dimensões físicas, não ao desempenho computacional. Um microcomputador típico (um PC ou uma estação de trabalho) cabe bem em uma mesa. Os microcomputadores são diretamente o produto dos avanços tecnológicos: CPUs mais rápidas, semicondutores, memórias etc. Ao longo do tempo, muitos dos conceitos anteriormente usados em *mainframes* e minicomputadores tornaram-se comuns e foram aplicados nos microcomputadores.

Fonte:https://www.shutterstock.com/image-photo/lap-top-computer-opened-281086982?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fsearch%2Fmicrocomputadores%2F

Arquitetura de computador

- O computador eletrônico, não importando seu tamanho, é uma coleção de componentes interligados com o objetivo de efetuar (processar) operações aritméticas e lógicas de grandes quantidades de dados (MANZANO, 2019, p. 17).

- A figura abaixo mostra de forma esquemática os componentes de um computador eletrônico.



Estrutura dos componentes de um computador eletrônico. (MANZANO, 2019. p. 18)

O que impulsiona o trabalho de um projetista de computadores

- Projetar um computador é uma tarefa desafiadora.
- Envolve o *software* (pelo menos ao nível de *design* do conjunto de instruções) e o *hardware*, em todos os seus níveis: organização funcional, projeto lógico, implementação.
- A própria implementação lida com o *design*/especificação de ICs (*Integrated Computer Systems*), embalagens, ruídos, energia, refrigeração etc.
    - Seria um erro terrível desconsiderar um aspecto ou outro do *design* do computador, pois o *designer* do computador precisa projetar uma máquina ideal em todos os níveis mencionados.

O que impulsiona o trabalho de um projetista de computadores

- Você não pode encontrar um aspecto que não esteja familiarizado com uma ampla gama de tecnologias, desde o compilador e o projeto do sistema operacional para o projeto lógico e embalagem.

- Um arquiteto de computador deve especificar os requisitos de desempenho de várias partes de um sistema de computador, para definir as interconexões para mantê-lo harmoniosamente equilibrado.

- O trabalho do arquiteto de computadores é mais do que projetar o conjunto de instruções, como tem sido entendido por muitos anos.

O que impulsiona o trabalho de um projetista de computadores

- Quanto mais um arquiteto é exposto a todos os aspectos do *design* de computadores, mais eficiente ele será.

- A arquitetura do conjunto de instruções se refere ao que o programador vê como o conjunto de instruções da máquina.

- O conjunto de instruções é o limite entre o *hardware* e o *software*, e a maioria das decisões relativas ao conjunto de instruções afeta o *hardware* e o inverso também é verdadeiro, pois muitas decisões de *hardware* podem afetar de maneira benéfica/adversa o conjunto de instruções.

O que impulsiona o trabalho de um projetista de computadores

- A implementação de uma máquina refere-se à lógica e às técnicas de *design* que são usadas para implementar uma instância da arquitetura.

- É possível ter diferentes implementações para alguma arquitetura, da mesma forma que existem possibilidades diferentes para construir uma casa usando os mesmos planos, mas com outros materiais e técnicas. A implementação tem dois aspectos:

    1. a organização se refere a aspectos lógicos de uma implementação. Em outras palavras, refere-se ao alto nível dos aspectos do *design*: *design* da CPU, sistema de memória, barramento etc.

    2. o *hardware* refere-se às especificidades de uma implementação.

Interatividade

Como são chamados os computadores grandes em que a maioria das inovações (tanto na arquitetura quanto na organização) foi feita e que podiam suportar muitos usuários enquanto ofereciam um grande poder de computação?

a) Mainframes.

b) Minicomputadores.

c) Supercomputadores.

d) Microcomputadores.

e) Mastercomputadores.

Resposta

Como são chamados os computadores grandes em que a maioria das inovações (tanto na arquitetura quanto na organização) foi feita e que podiam suportar muitos usuários enquanto ofereciam um grande poder de computação?

a) Mainframes.

b) Minicomputadores.

c) Supercomputadores.

d) Microcomputadores.

e) Mastercomputadores.

Código-fonte e código do objeto

- O SDLC é uma estrutura ou processo conceitual que considera a estrutura dos estágios envolvidos no desenvolvimento de uma aplicação desde o estudo inicial de viabilidade até a implantação no campo e manutenção.

- Um SDLC geralmente é usado para descrever as etapas que são seguidas dentro da estrutura do ciclo de vida do *software*.

- As etapas envolvidas no SDLC fornecem uma compreensão do código-fonte e do código do objeto. Este tópico irá apresentar a diferença entre eles.

    - A diferença entre o Código-fonte e o Código do Objeto é que o Código-fonte é uma coleção de instruções de computador escritas usando um manual legível por humanos a partir de uma linguagem de programação, enquanto o Código do Objeto (*Object Code*) representa uma sequência de instruções em uma linguagem de máquina e é a saída após o compilador converter o código-fonte.

O que é o código-fonte?

- DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE (2018) afirma que antes de desenvolver o *software*, deve haver uma compreensão dos requisitos (funcionais e não funcionais).

- Os analistas obtêm as funcionalidades necessárias do usuário e as documentam. Esse documento contém justamente a especificação dos requisitos do sistema (SRS - *System Requirement Specification*).

- Ele fornece a documentação descritiva das funcionalidades necessárias. Com base nesse documento, o sistema poderá ser projetado.

    - O projeto do sistema pode ser feito usando fluxogramas, diagramas de fluxo de dados (DFD – *Data Flow Diagrams*). As saídas da fase de *design* podem ser a produção do *design* de banco de dados, *design* de processo etc.

O que é o código-fonte?

- Após a fase de *design* ser concluída, esses projetos podem ser implementados usando uma linguagem de programação relevante por um programador.

- As linguagens de baixo nível possibilitam uma comunicação mais natural com a máquina. É uma maneira de codificação considerada por muitos como de grande dificuldade para uso. Destacam-se nessa categoria as linguagens de máquina e Assembly. A linguagem Assembly (e não assembler) é mais fácil de usar do que uma linguagem de máquina, por ser baseada em comandos de instruções em formato mnemônico (siglas com significado definido de suas ações) (MANZANO, 2019, p. 24).

Fonte: https://www.shutterstock.com/image-photo/coding-code-program-programming-compute-coder-517228516?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fsearch%2Fprograma%25C3%25A7%25C3%25A3o%2F

O que é o código-fonte?

- A linguagem Assembly não é mais a linguagem na qual novos aplicativos são escritos, embora as partes mais sensíveis continuem sendo escritas em linguagem Assembly, e isso se deve aos avanços nas linguagens de programação e na tecnologia dos compiladores.

- A obsolescência da programação em linguagem Assembly, bem como a criação de sistemas operacionais portáteis (como UNIX), reduziram os riscos de introduzir novas arquiteturas. Novas famílias de computadores estão surgindo, muitos deles híbridos de famílias "clássicas": supercomputadores gráficos, multiprocessadores, MPP (*Massively Parallel Processors*), minissupercomputadores etc.

Fonte: https://www.shutterstock.com/image-photo/pensive-programmer-working-on-desktop-pc-1569671278?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fsearch%2Fprograma%25C3%25A7%25C3%25A3o%2F

O que é o código-fonte?

- Existem muitas linguagens de programação. Algumas delas são C, C++, C#, Python, Java etc. O programador pode selecionar a linguagem de programação de acordo com o projeto do *software* e converter os diagramas e a documentação em programas de computador. As instruções foram escritas para alcançar as funcionalidades do *software* necessário usando a linguagem de programação.

- As instruções têm uma sintaxe semelhante ao idioma inglês e legível por um ser humano. Essa coleção de instruções escritas usando uma linguagem de programação legível por humanos é chamada de código-fonte.

    - As linguagens de alto nível possibilitam maior facilidade de comunicação com um computador pelo fato de serem expressadas de maneira mais próxima à comunicação humana, pois baseiam-se em palavras do idioma inglês.

O que é o código-fonte?

- Destacam-se, nessa categoria, linguagens de programação como FORTRAN, COBOL, BASIC, PASCAL, C, JAVA, Lua, C++, entre outras.

- Esse tipo de linguagem é mais facilmente assimilado por seres humanos. Assim, o número de pessoas que as conhece é bastante grande. Nesse sentido, considere o exemplo de um programa que apresenta a palavra mundo na tela, codificado nas linguagens de alto nível (Pascal, C++ e C), e observe que elas são eminentemente mais fáceis de expressar uma ideia do que suas equivalentes em baixo nível (MANZANO, 2019, p. 24).

Fonte: https://pixabay.com/photos/coding-programming-working-macbook-924920/

O que é código do objeto?

- DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE (2018) apresenta que o código-fonte é compreensível por humanos porque possui uma sintaxe semelhante à do inglês. Não é compreensível por um computador ou uma máquina. Os computadores ou máquinas compreendem uma linguagem binária que consiste em zeros e um. Portanto, é necessário converter o código-fonte em um formato compreensível pela máquina. O compilador converte o código-fonte em uma linguagem binária ou linguagem de máquina.

- Esse código convertido é conhecido como código do objeto que é compreensível pelo computador (as instruções dadas pelo ser humano são compreensíveis pelo computador a partir dessa conversão).

Fonte: https://pixabay.com/photos/code-code-editor-coding-computer-1839406/

Qual é a diferença entre Código-fonte e Código do Objeto?

- Os programas de computador são úteis para fornecer instruções ao computador para executar uma tarefa específica. Esses programas são escritos usando linguagens de programação.

- Existem diversas linguagens de programação, e o programador pode selecionar uma linguagem para desenvolver os seus programas ou *software*. Código-fonte e Código do Objeto são dois termos associados à área de programação.

    - A diferença entre o código-fonte e o código do objeto é que o código-fonte é uma coleção de instruções de computador escritas usando um código legível por humanos a partir de uma linguagem de programação, enquanto o Código do Objeto (*Object Code*) é uma sequência de instruções na linguagem de máquina e é a saída após o compilador converter o código-fonte (DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE, 2018).

Qual é a diferença entre Código-fonte e Código do Objeto?

Código-fonte x Código do objeto	
O Código-fonte é uma coleção de instruções do computador escritas a partir de uma linguagem de programação legível por humanos.	O Código do Objeto é uma sequência de instruções escritas em linguagem de máquina ou binário e é a saída após o compilador converter o código-fonte.
Compreensibilidade	
O Código-fonte é legível pelo programador de sistema.	O Código do Objeto é legível pelo computador (linguagem de máquina).
Geração	
O humano gera o Código-fonte.	O compilador gera o Código do Objeto.
Formato	
O Código-fonte está na forma textual (a partir da linguagem de programação).	O Código do Objeto está na forma de números binários (linguagem de máquina).

Linguagens formais e ambiente de programação

- De acordo com Slonneger & Kurtz (1995), a linguagem fornece um meio de comunicação sonora e escrita. Os seres humanos aprendem uma língua como consequência de suas experiências de vida, mas na linguística – a ciência das línguas –, as formas e os significados das línguas são submetidos a um exame mais rigoroso.

- Essa ciência também pode ser aplicada no processo de desenvolvimento de programas computacionais. Em contraste com as linguagens naturais, com as quais comunicamos nossos pensamentos e sentimentos, as linguagens de programação podem ser vistas como linguagens artificiais definidas por homens e mulheres inicialmente para fins de comunicação com computadores, mas, o que é mais importante, para comunicar algoritmos entre as pessoas.

Linguagens formais e ambiente de programação

Slonneger & Kurtz (1995) apresentam que muitos dos métodos e grande parte da terminologia da linguística se aplicam às linguagens de programação. Por exemplo, as definições de idioma consistem em três componentes:

1. A Sintaxe refere-se à maneira como os símbolos podem ser combinados para criar frases (ou programas) no idioma relacionado. A sintaxe define as relações formais entre os constituintes de uma linguagem, fornecendo, assim, uma descrição estrutural das várias expressões que a compõem. A sintaxe lida apenas com a forma e a estrutura dos símbolos em um idioma sem nenhuma consideração dada ao seu significado.

Fonte: https://www.shutterstock.com/image-photo/developing-programming-coding-technologies-website-design-699634498?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fcode-code-editor-coding-computer-1839406%2F

Linguagens formais e ambiente de programação

2. A semântica revela o significado das cadeias sintaticamente válidas em um idioma. Para as linguagens naturais, isso significa correlacionar sentenças e frases com os objetos, pensamentos e sentimentos de nossas experiências. Para as linguagens de programação, a semântica descreve o comportamento que um computador segue ao executar um programa em sua linguagem. Podemos divulgar esse comportamento descrevendo o relacionamento entre a entrada e a saída de um programa ou uma explicação passo a passo de como um programa será executado em uma máquina real ou abstrata.

3. Para a programação, as linguagens incluem questões como a facilidade de implementação, eficiência na aplicação e metodologia de programação.

   A sintaxe deve ser especificada antes da semântica, pois o significado pode ser dado apenas para as expressões que são formadas em uma linguagem em específico. Da mesma forma, a semântica precisa antes considerar as questões pragmáticas, uma vez que a interação com os usuários pode ser considerada apenas para expressões cujo significado é entendido.

Interatividade

Sabendo que o Código-fonte é uma coleção de instruções de computador escritas usando um manual legível por humanos a partir de uma linguagem de programação, selecione a alternativa que contém a definição para o termo “Código Objeto (*Object Code*)”.

a) Representa uma sequência de instruções em uma linguagem de alto nível e é a entrada de dados.

b) Representa uma sequência de instruções em uma linguagem de alto nível e é a saída após o compilador converter o código-fonte.

c) Representa uma sequência de instruções em uma linguagem de máquina e é a entrada de dados.

d) Representa uma sequência de instruções em uma linguagem de máquina e é a entrada após o compilador converter o código-fonte.

e) Representa uma sequência de instruções em uma linguagem de máquina e é a saída após o compilador converter o código-fonte.

Resposta

Sabendo que o Código-fonte é uma coleção de instruções de computador escritas usando um manual legível por humanos a partir de uma linguagem de programação, selecione a alternativa que contém a definição para o termo “Código Objeto (*Object Code*)”.

a) Representa uma sequência de instruções em uma linguagem de alto nível e é a entrada de dados.

b) Representa uma sequência de instruções em uma linguagem de alto nível e é a saída após o compilador converter o código-fonte.

c) Representa uma sequência de instruções em uma linguagem de máquina e é a entrada de dados.

d) Representa uma sequência de instruções em uma linguagem de máquina e é a entrada após o compilador converter o código-fonte.

e) Representa uma sequência de instruções em uma linguagem de máquina e é a saída após o compilador converter o código-fonte.

Visão geral sobre os Ambientes de Desenvolvimento de *Software*

- De acordo com Dart *et al.* (1992), os ambientes de desenvolvimento de *software* referem-se à coleção de ferramentas de *hardware* e *software* que um sistema desenvolvedor utiliza para construir sistemas de *software*.

- À medida que a tecnologia melhora e as expectativas do usuário aumentam, a funcionalidade de um ambiente tende a mudar.

- Ao longo dos últimos anos, o conjunto de ferramentas de *software* disponível para desenvolvedores está expandido de forma considerável. Podemos ilustrar essa mudança observando algumas distinções na terminologia.

    - O ambiente de programação e ambiente de desenvolvimento de *software* são frequentemente usados como sinônimo, porém ao longo desse tópico faremos uma distinção entre esses dois termos.

Visão geral sobre os Ambientes de Desenvolvimento de *Software*

- O termo ambiente de programação, entendemos como sendo um ambiente que suporta apenas a fase de codificação do ciclo de desenvolvimento de *software* – ou seja, apenas as tarefas pequenas de programação, como edição e compilação.

- O termo ambiente de desenvolvimento de *software*, entendemos que está relacionado com um ambiente que aumenta ou automatiza as atividades que compreendem o ciclo de desenvolvimento de *software*, incluindo as grandes tarefas de programação, como o gerenciamento de configuração e a programação de diversas tarefas, como o gerenciamento de projetos e equipes; significa ainda o ambiente que suporta uma manutenção em larga escala e a um longo prazo.

Fonte: https://pixabay.com/photos/code-programming-hacking-html-web-820275/

Visão geral sobre os Ambientes de Desenvolvimento de *Software*

- A evolução dos ambientes também exige a distinção dos recursos básicos do sistema operacional – serviços fundamentais como memória, dados e o gerenciamento de vários programas – a partir da funcionalidade aprimorada que caracteriza os ambientes de última geração.

- Essa funcionalidade aprimorada é normalmente obtida por meio de ferramentas como navegadores, gerenciadores de janelas, gerenciadores de configuração e gerenciadores de tarefas.

    - Em certo sentido, os ambientes têm evoluído de acordo com o entendimento da comunidade de engenharia de *software* sobre as tarefas envolvidas no desenvolvimento de sistemas de *software*.

Visão geral sobre os Ambientes de Desenvolvimento de *Software*

- Dart *et al.* (1992) citam em sua obra uma taxonomia dessas tendências. Segundo os autores, a taxonomia compreende quatro categorias, cada uma representando tendências com um impacto significativo nos ambientes – em suas ferramentas, interfaces de usuário e arquiteturas. As quatro categorias são:

1. **Ambientes centrados na linguagem**: Eles são criados em torno de um idioma, fornecendo um conjunto de ferramentas adequadas para esse idioma. Esses ambientes são altamente interativos e oferecem recursos limitados e um suporte para programação em geral.

2. **Ambientes orientados à estrutura**: Eles incorporam técnicas que permitem ao usuário manipular estruturas de forma direta. A independência linguística das técnicas levou à noção dos geradores para os ambientes.

Visão geral sobre os Ambientes de Desenvolvimento de *Software*

3. **Ambientes do kit de ferramentas**: Eles fornecem uma coleção de ferramentas que incluem o suporte independente da linguagem para tarefas de programação de forma ampla, como o gerenciamento de configuração e controle de versão.

4. **Ambientes baseados em métodos**: Eles incorporam suporte para uma ampla gama de atividades no processo de desenvolvimento de *software*, incluindo tarefas como gerenciamento de equipe e projeto. Esses ambientes também incorporam ferramentas para o desenvolvimento de métodos específicos e d*esign*.

Fonte: https://pixabay.com/photos/coding-programming-working-macbook-924920/

Programação em geral

- Conforme Dart *et al*. (1992), os ambientes centrados na linguagem oferecem ao usuário um universo de diversas linguagens de programação.

- Esses ambientes são adequados para a fase de codificação do ciclo de desenvolvimento de *software*. Eles fornecem técnicas incrementais de compilação ou interpretação para ajudar a reduzir o impacto de pequenas alterações de código durante a manutenção.

- O estilo exploratório de programação que eles suportam ajuda ao usuário a experimentar protótipos de *software*.

    - Ferramentas como navegadores, além de serem extremamente úteis para o usuário durante o desenvolvimento do programa exploratório, podem ser bastante eficazes para a manutenção de grandes sistemas de *software*.

Programação em geral

- Devido às técnicas especializadas usadas para implementá-las, esses ambientes geralmente não oferecem suporte a várias linguagens e, em alguns casos, não facilitam a portabilidade de programas e aplicativos. Além disso, os ambientes centrados na linguagem podem se tornar grandes demais para uma pessoa compreender e estender.

- Os ambientes para linguagens imperativas suportam recursos de programação, como o controle de versão. Mas eles atualmente não suportam o processo de implementar as muitas tarefas, como o gerenciamento de projetos, nem fornecem suporte para as tarefas de desenvolvimento que não sejam a codificação.

    - Isso não está claro se esses ambientes podem ampliar suas instalações para atender a esses requisitos, mas provavelmente formarão um componente de ambientes futuros que darão suporte a todo o ciclo de vida do *software*.

Programação em geral

- Os desenvolvedores de sistemas comerciais de *software* estão tentando refinar suas técnicas de implementação para melhorar o desempenho.

- Eles estão construindo ambientes centrados na linguagem para linguagens imperativas como C e estão tentando ampliar esses ambientes para dar suporte à fase de *design* e incorporar algumas técnicas de gerenciamento de projetos.

- Dart *et al.* (1992) afirmam que a motivação inicial para ambientes orientados para a estrutura era dar aos usuários uma ferramenta interativa.

    - Esse recurso foi estendido para fornecer aos ambientes de programação de usuário um suporte à semântica interativa, o processo de execução de programa e a sua depuração.

Programação em geral

- O editor é o componente central de tais ambientes; é a interface pela qual o usuário interage e pela qual todas as estruturas são manipuladas.

- Os esforços foram continuados para possibilitar o suporte à programação, pois os ambientes orientados à estrutura fizeram várias contribuições para a tecnologia do ambiente, como o fornecimento da manipulação direta do programa, a verificação incremental da semântica estática e diversas informações acessíveis ao usuário e, mais importante, a capacidade de descrever formalmente a sintaxe e a semântica de uma linguagem a partir do qual uma instância de um editor de estrutura pode ser gerada.

Fonte: https://www.shutterstock.com/image-photo/coding-code-program-programming-compute-coder-517228516?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fcode-code-editor-coding-computer-1839406%2F

Algoritmos

Charntaweekhun & Wangsiripitak (2006) afirmam que o processo de programar centrado apenas na linguagem de programação é o caminho que muitos programadores utilizam para desenvolver programas e *softwares*. O processo para a programação usando uma linguagem de computador possui 3 etapas centrais:

1. Digitar o código de acordo com o algoritmo
2. Utilizar um código-fonte de compilação
3. Verificar e validar a partir de um programa de teste e depuração

Fonte: : https://pixabay.com/photos/whiteboard-writing-man-presentation-849814/

Algoritmos

- O primeiro e o último passo estão atrelados com as tarefas que os programadores devem realizar, porém a segunda etapa está relacionada com uma demanda do *complier* (compilador).
- A maioria dos programadores investe muito tempo na primeira etapa, pois converter a ideia (problema) em um código-fonte é muito difícil quando estamos diante de algoritmos complexos.
- Às vezes é possível inclusive finalizar a codificação do projeto, porém são enfrentados diversos erros de compilação. Encontrar esses erros e corrigi-los é uma tarefa muito árdua.
    - Por esse motivo, uma das possibilidades que temos disponíveis é desenvolver os algoritmos a partir da criação dos fluxogramas, que nada mais é do que a base do modelo que os desenvolvedores podem utilizar para redigir suas ideias em papéis (de forma manual). Desta forma, é fácil encontrar os erros no processo de criação sem ao menos implementar uma linha de código.

Algoritmos

- Olhe a sua volta, os computadores e as redes estão por toda parte, permitindo diversas atividades humanas complexas, como: a educação, o comércio, o entretenimento, a pesquisa, a fabricação, a gestão da saúde, a comunicação e até a guerra.

- Dos dois principais fundamentos tecnológicos dessa incrível proliferação, um é o ritmo veloz com o qual avança a microeletrônica, que possibilitou um *hardware* cada vez mais rápido. O outro é a possibilidade de construir algoritmos eficientes que estão alimentando a revolução do computador.

Fonte: https://pixabay.com/photos/whiteboard-writing-man-presentation-849815/

Algoritmos

- Em termos de ciência da computação, um algoritmo é um resumo, uma descrição formalizada de um procedimento computacional. Os algoritmos se dividem em tipos diferentes, de acordo com as suas propriedades ou domínios (como, por exemplo, o processo de lidar com algoritmos combinatórios com contagem e enumeração), além de poder variar em termos de seus critérios analíticos (como, por exemplo, como características generalizadas de desempenho, como o desempenho médio).

- Em matemática, ciência da computação e assuntos relacionados, um algoritmo é uma sequência finita de etapas expressa para resolver um problema. Um algoritmo pode ser definido como sendo um processo que executa algumas sequências de operações para resolver um problema.

    - Os algoritmos são usados para a execução de cálculo, processamento de dados e muitos outros campos. Na computação, os algoritmos são essenciais porque servem como o procedimento sistemático exigido pelos computadores.

Algoritmos

- O termo algoritmo normalmente causa certa estranheza a algumas pessoas, pois muitas acreditam que está escrito ou pronunciado de forma incorreta. A palavra algoritmo vem do latim, dos termos *algorismos* ou *algorithmos*, que estão associados à ideia de algarismos por influência do idioma grego a partir do termo *arithmós*, que remete a números.
  A palavra algoritmo é aplicada e empregada, segundo o dicionário Houaiss, em matemática e computação.

- Na esfera matemática, está associada a um processo de cálculo; encadeamento das ações necessárias ao cumprimento de uma tarefa; processo efetivo, que produz uma solução para um problema em um número finito de etapas.

  - Na ciência da computação (informática), está associada a um conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas (MANZANO, 2019, p. 30).

Interatividade

O termo algoritmo normalmente causa certa estranheza a algumas pessoas, pois muitas acreditam que está escrito ou pronunciado de forma incorreta. A palavra algoritmo vem do latim, dos termos *algorismos* ou *algorithmos*, que estão associados à ideia de algarismos por influência do idioma grego a partir do termo *arithmós*, que remete a números.

Selecione a afirmação que explica o que seria o algoritmo:

a) Um algoritmo é um esboço, uma descrição descontextualizada de um procedimento matemático.

b) Um algoritmo é um resumo, uma descrição formalizada de um procedimento computacional.

c) Um algoritmo é um resumo, uma descrição descontextualizada de um procedimento computacional.

d) Um algoritmo é um resumo, uma descrição formalizada de um procedimento geométrico.

e) Um algoritmo é um resumo, uma descrição formalizada de um procedimento matemático.

Resposta

O termo algoritmo normalmente causa certa estranheza a algumas pessoas, pois muitas acreditam que está escrito ou pronunciado de forma incorreta. A palavra algoritmo vem do latim, dos termos *algorismos* ou *algorithmos*, que estão associados à ideia de algarismos por influência do idioma grego a partir do termo *arithmós*, que remete a números.

Selecione a afirmação que explica o que seria o algoritmo:

a) Um algoritmo é um esboço, uma descrição descontextualizada de um procedimento matemático.

b) Um algoritmo é um resumo, uma descrição formalizada de um procedimento computacional.

c) Um algoritmo é um resumo, uma descrição descontextualizada de um procedimento computacional.

d) Um algoritmo é um resumo, uma descrição formalizada de um procedimento geométrico.

e) Um algoritmo é um resumo, uma descrição formalizada de um procedimento matemático.

Algoritmos

- De acordo com Charntaweekhun & Wangsiripitak (2006), na programação de computadores, muitas vezes existem diferentes algoritmos para realizar qualquer tarefa. Cada algoritmo possui diversas vantagens e desvantagens em diferentes situações. É possível incentivar a utilização da construção dos algoritmos a partir de três razões: eficiência, abstração e reutilização.

- A eficiência é um dos elementos fundamentais para auxiliar no processo de resolução de problemas.

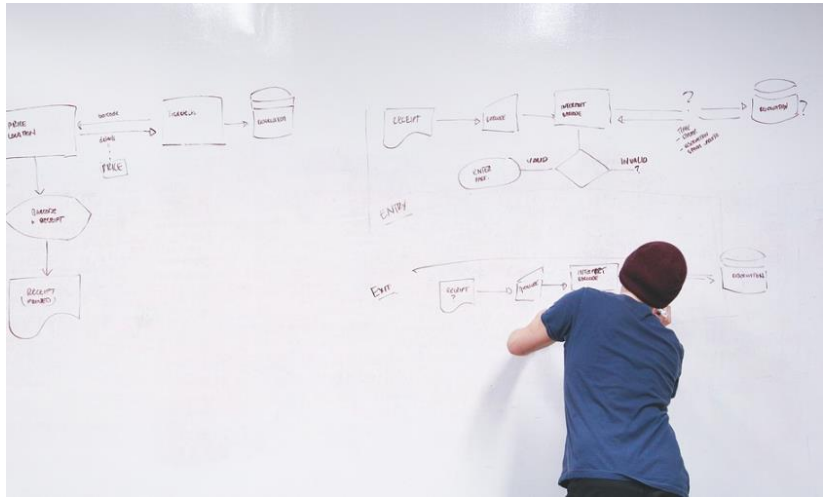
    - Os algoritmos eficientes devem ser usados para resolver tais problemas considerando o tempo e o fator de custo envolvidos em cada algoritmo.

    - A abstração é a análise sistemática dos algoritmos que podem fornecer um nível de abstração para resolver problemas.

Algoritmos

- Alguns elementos são aparentemente complicados e, quando são abstraídos (divididos em partes menores), os problemas podem ser destilados em outros mais simples, para os quais existem algoritmos conhecidos.

- Por exemplo, imagine tentar encontrar o caminho mais curto para rotear um pacote entre dois *gateways* em uma internet. Uma vez que percebemos que esse problema é apenas uma variação do problema que apresenta o caminho mais curto, podemos resolvê-lo usando a abordagem generalizada. Já a reutilização é a aplicação das soluções que foram implementadas anteriormente em diferentes situações.



Fonte: https://pixabay.com/photos/whiteboard-writing-man-presentation-849815/

Algoritmos

- Walia (2020) afirma que a palavra *algorithm* (algoritmo) refere-se ao nome do matemático Al-khowarizmi, que significa um procedimento ou uma técnica. O engenheiro de *software* geralmente usa um algoritmo para planejar e resolver os problemas.

- Um algoritmo é uma sequência de etapas para resolver um problema específico ou pode ser definido como sendo um conjunto ordenado de etapas inequívocas que produz um resultado e termina em um tempo finito.

    - Muitos profissionais da área de programação de computadores (principalmente os mais experientes, cautelosos e cuidadosos) preferem elaborar seus programas com base em um projeto que aborde todos os aspectos técnicos do desenvolvimento, com atenção especial sempre à parte do projeto lógico.

Algoritmos

- O projeto de desenvolvimento de sistemas (conjunto de programas de computador com o mesmo propósito e interligados) segue diversas etapas de um processo normalmente conhecido como análise de sistemas.

- O foco dessa obra é o projeto lógico, a parte do desenvolvimento do programa de um sistema em si. Assim, apenas os aspectos relacionados ao desenvolvimento e à escrita de rotinas de programas e seu projeto serão abordados (MANZANO, 2019, p. 34).

Fonte: https://pixabay.com/photos/whiteboard-man-presentation-write-849812/

Algoritmos

- A partir desse princípio, o algoritmo possui as seguintes características:
- *Input* (Entrada): um algoritmo pode ou não exigir a entrada de dados.
- *Output* (Saída): espera-se que cada algoritmo produza pelo menos um resultado (processo).
- *Definiteness* (Definitividade): cada instrução deve ser clara e inequívoca.
- *Finiteness* (Finitude): caso as instruções de um algoritmo sejam executadas, o algoritmo deve terminar após um número finito de etapas.

Algoritmos

- **Como escrever algoritmos**

- **Etapa 1 – Defina a entrada de seus algoritmos**: muitos algoritmos recebem dados para serem processados, por exemplo, para calcular a área de entrada do retângulo pode ser a altura e a largura do retângulo.

- **Etapa 2 – Definir as variáveis**: as variáveis do algoritmo permitem que você as utilize por mais de um lugar. Podemos definir duas variáveis para altura e largura do retângulo como HEIGHT e WIDTH (ou H & W).

Fonte: https://www.shutterstock.com/image-photo/businessman-executive-manager-hand-filling-paper-1687192501

Algoritmos

- **Como escrever algoritmos**

- **Etapa 3 – Descreva as operações do algoritmo**: use a variável de entrada para fins de cálculo, por exemplo. Para encontrar a área do retângulo, multiplique as variáveis HEIGHT e WIDTH e armazene o valor em uma nova variável que poderíamos nomear como sendo AREA. As operações de um algoritmo podem assumir a forma de várias etapas e dependendo do valor das variáveis de entrada.

- **Etapa 4 – Saída dos resultados das operações do seu algoritmo**: no caso da área do retângulo, a saída será o valor armazenado na variável AREA. Caso as variáveis de entrada descrevessem um retângulo com uma altura de 5 e uma largura de 2, o algoritmo produziria o valor de 10.

Algoritmos

- O algoritmo é uma representação passo a passo de uma solução para um determinado problema, o que facilita para entender todo o contexto, utiliza um procedimento definido e não depende de nenhuma linguagem de programação, por esse motivo é fácil compreender, mesmo sem um conhecimento prévio de programação.

Fonte: https://www.shutterstock.com/image-photo/document-management-data-system-business-internet-1416401996

Fluxograma

- De acordo com Walia (2020), o primeiro *design* do fluxograma foi desenvolvido em 1945, por John Von Neumann. Diferentemente de um algoritmo, o fluxograma utiliza símbolos para projetar uma solução para um problema. Ao olhar para um fluxograma, você pode entender as operações e a sequência de operações que serão executadas em um sistema. O fluxograma é frequentemente considerado como sendo a planta de um projeto usado para resolver um problema específico.

- Os símbolos de identificação gráfica representam sempre uma operação ou conjunto de operações similares, podendo ser identificados por um rótulo relacionado à própria ação do símbolo em uso, somente quando necessário.

    - Os símbolos devem ser conectados uns aos outros por linhas de setas que mostrem explicitamente a direção do fluxo a ser executado pelo programa.

Fluxograma

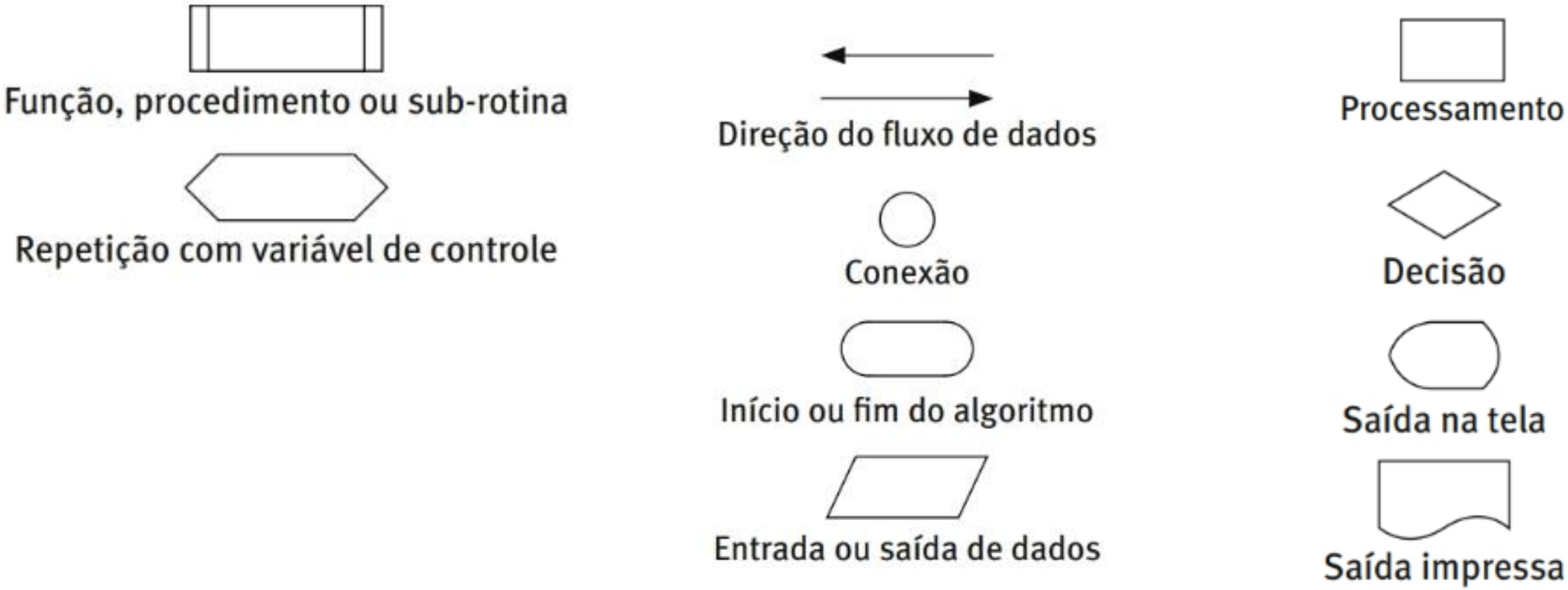
- A estrutura visual do diagrama deve, a princípio, estar orientada no sentido de cima para baixo, da direita para a esquerda e ser desenhada no centro da folha de papel. No entanto, dependendo da situação, esse critério pode ser alterado, o que leva à necessidade de manter o uso das linhas de seta indicando a direção do fluxo.

- A definição de inicialização e finalização de um diagrama ocorre com o uso do símbolo “terminal” devidamente identificado com um dos rótulos: início, fim, retorno ou a definição de um nome particular, quando for necessário, desde que seguidas as regras de utilização de sub-rotinas (a serem apresentadas em momento oportuno) (MANZANO, 2019, p. 38).

- O fluxograma é muito importante para desenvolver a compreensão de como um processo será realizado, além de melhorar a comunicação com os membros da equipe (todas as pessoas envolvidas no mesmo processo) e documentar os processos que serão implementados.

Fluxograma

- De acordo com Soffner (2013), os algoritmos podem ser representados de forma gráfica, por meio de símbolos padronizados (fluxogramas, também chamados de diagramas de blocos). Segue abaixo uma imagem que representa a forma como os algoritmos são representados.



Fonte: Simbologia para diagramas de blocos. (SOFFNER, 2013., p. 22)

Interatividade

O fluxograma é muito importante para desenvolver a compreensão de como um processo será realizado, além de melhorar a comunicação com os membros da equipe (todas as pessoas envolvidas no mesmo processo) e documentar os processos que serão implementados.

De que forma o fluxograma é frequentemente considerado?

a) É considerado como sendo o mapa mental de um projeto.
b) É considerado como sendo o resumo gráfico do projeto.
c) É considerado como sendo a programação de um projeto.
d) É considerado como sendo a planta de um projeto usado para resolver um problema específico.
e) É considerado como sendo uma etapa que pode ser facilmente eliminada.

Resposta

O fluxograma é muito importante para desenvolver a compreensão de como um processo será realizado, além de melhorar a comunicação com os membros da equipe (todas as pessoas envolvidas no mesmo processo) e documentar os processos que serão implementados.

De que forma o fluxograma é frequentemente considerado?

a) É considerado como sendo o mapa mental de um projeto.
b) É considerado como sendo o resumo gráfico do projeto.
c) É considerado como sendo a programação de um projeto.
d) É considerado como sendo a planta de um projeto usado para resolver um problema específico.
e) É considerado como sendo uma etapa que pode ser facilmente eliminada.

Referências

- ART, Susan; ELLISON, Robert; FEILER, Peter; EDITED, A.; FRITZSON, Peter. *Overview of Software Development Environments*. 1992. Disponível em: https://www.ics.uci.edu/~andre/ics228s2006/dartellisonfeilerhabermann.pdf. Acesso em: 25 abr. 2020.
- BEALE, E. M. L. Matrix Generators and Output Analyzers. *In* KUHN, H. W. (ed.). *Proceedings of the Princeton Symposium on Mathematical Programming.* Princeton, NJ: Princeton University Press, p. 25-36, 1970.
- BROOKSHEAR, J. G. *Computer Science*: An Overview. Boston, Mass.: Pearson, 2009.
- CHARNTAWEEKHUN, Kanis; WANGSIRIPITAK, Somkiat. *Visual Programming using Flowchart*. 2006. Disponível em: 10.1109/ISCIT.2006.339940. Acesso em: 25 abr. 2020.
- CREEGAN, J. B. *DATAFORM, a Model Management System*. Ketron, Inc., Arlington, VA, 1985.

Referências

- DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE. 24/01/2018. Disponível em: https://www.differencebetween.com/wp-content/uploads/2018/01/Difference-Between-Source-Code-and-Object-Code.pdf. Acesso em: 25 abr. 2020.
- MANZANO, José Augusto N. G. *Algoritmos*: lógica para desenvolvimento de programação de computadores. 29. ed. São Paulo: Érica, 2019.
- SLONNEGER, Kenneth; KURTZ, Barry L. *Formal syntax and semantics of programming languages*: a laboratory. 1995. Disponível em: https://www.mobt3ath.com/uplode/book/book-26246.pdf. Acesso em: 25 abr. 2020.
- SOFFNER, R. *Algoritmos e Programação em Linguagem C*. São Paulo: Saraiva, 2013.
- SOICHER, Leonard; VIVALDI, Franco. *Algorithmic Mathematics*. University of London, 2004. Disponível em: http://www.maths.qmul.ac.uk/~lsoicher/ambook.pdf. Acesso em: 28 abr. 2020.

Referências

- STALLINGS, W. *Computer Organization and Architecture, Designing for Performance*. Boston, Mass.: Pearson. 2010.
- UNIVERSITY OF CRETE. 2020. *Introduction to Data Types and Structures*. Disponível em: https://www.csd.uoc.gr/~hy252/html/References/Introduction%20to%20Data%20Types%20and%20Structures.pdf. Acesso em: 28 abr. 2020.
- WALIA, Ravi K. *ALGORITHM & FLOWCHART MANUAL for STUDENTS*. University of Horticulture & Forestry – Índia, 2020. Disponível em: http://www.yspuniversity.ac.in/cic/algorithm-manual.pdf. Acesso em: 26 abr. 2020.
- WIRTH, N. *Algorithms and Data Structures*. 1985. Disponível em: https://inf.ethz.ch/personal/wirth/AD.pdf. Acesso em: 28 abr. 2020.
- YANG, Kuo-pao. *Chapter 6 - Data Types*. 2017. Disponível em: https://www2.southeastern.edu/Academics/Faculty/kyang/2017/Fall/CMPS401/ClassNotes/CMPS401ClassNotesChap06.pdf. Acesso em: 27 abr. 2020.

ATÉ A PRÓXIMA!