*O código e senha dos moddles está em um .txt separado.*

*Texto em preto: informações sobre o conteúdo*

*Texto em azul: opiniões pessoais*

*Texto em vermelho: tarefas ou trabalhos*

**01/10 (Terça-Feira)**

Gerência de Configuração de Software (GC):

Foi introduzida a GCS. Essa área auxilia na coordenação de um projeto ao auxiliar no controle e organização de mudanças e versões.

Tarefa no Moodle até Terça dia 08/10 às 19:00: Elencar os 18 mitos da GCS e elaborar sobre um deles. Um arquivo .txt com todos os mitos traduzidos e ajuda para acessar o livro está no GitHub.

Serão realizados seminários do dia 10/12 ao 18/02 sobre plataformas de GCS.

|  |  |
| --- | --- |
| Data | Tema |
| 10/12 | GitHub e GitLab |
| 17/12 | BitBucket e Bazaar |
| 04/02 | CVS e Selenium |
| 11/02 | Subversion e Mercurial |
| 18/02 | TFS e RedMine |

Banco de Dados (BD):

Foi revisada a matéria de BD. Essa área é útil para o armazenamento, organização e acesso de informações em imenso volume. Sistemas Gerenciadores de Banco de Dados (SGDBs) são usados como intermediários entre o usuário e o banco de dados, permitindo o emprego de estratégias mais avançadas para a definição, construção e manipulação de Bancos de Dados.

|  |
| --- |
| Vantagens de um SGDB |
| * Independência de dados |
| * Acesso eficiente |
| * Tempo reduzido no desenvolvimento de aplicações |
| * Segurança e integridade dos dados |
| * Administração de dados uniforme |
| * Acesso concorrente |
| * Recuperação contra crashes |

**02/10 (Quarta-Feira)**

Banco de Dados (BD):

Foi revisado o SQL, relembrando a Data Definition Language (DDL) e Data Manipulation Language (DML). Foi realizado um exercício (sem entrega) com a criação e manipulação de tabelas.

Um site com explicações e exemplos do SQL se encontra nesse link: [https://www.w3schools.com/sql/](https://www.w3schools.com/sql/default.asp)

Outros links úteis estão no Moodle da matéria.

Algoritmos e Estruturas de Dados (AE):

Foi introduzida a disciplina. Essa disciplina envolverá:

* Análise de algoritmos
* Tipos abstratos de dados
* Pilhas
* Filas
* Listas ordenadas
* Arvores (binárias)

Foi revisada a definição de algoritmos **(uma sequência finita de instruções não ambíguas para resolver um problema)**

Foi feita a análise matemática do custo de execução um algoritmo, onde cada instrução vale **1**, e instruções que repetem **n** vezes valem **n**, ou seja, um loop *for* com 3 instruções dentro vale **2**(inicialização) + **2n**(iteração) + **3n**(instruções dentro) ou seja, **2 + 5n**.

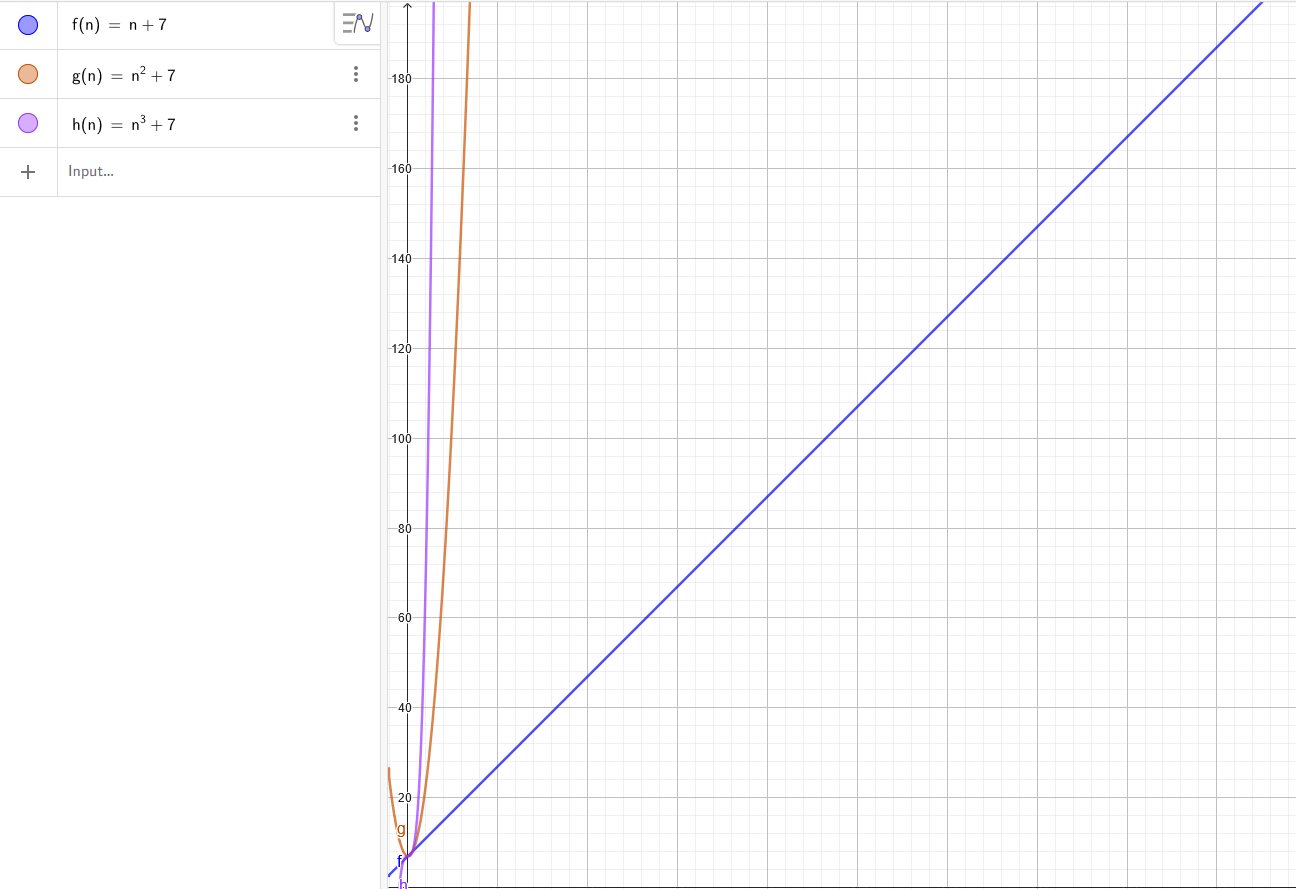
Também foi vista a possibilidade de melhores casos (execução mais rápida) e piores casos (execução mais lenta). Por exemplo, uma função que conta todas as vogais de uma String teria em seu melhor caso uma entrada vazia, e em seu pior caso com uma entrada composta exclusivamente por vogais, pois no último caso ela executaria quarto instruções (teste do for, incremento do for, teste da vogal, incremento da contagem) para cada letra.

**03/10 (Quinta-Feira)**

Algoritmos e Estruturas de Dados (AE):

Foi introduzido o conceito de comportamento assintótico. A ideia é que, com quantidades enormes de dados, variáveis constantes são inconsequentes e podem ser descartadas. Por exemplo:

|  |  |  |
| --- | --- | --- |
|  | **n = 10** | **n = 8000** |
| n + 7 | 17 | 8007 |
| n² + 7 | 107 | 64000007 |
| n³ + 7 | 1007 | 512000000007 |



Como a variável com o maior expoente é o fator determinante do custo apenas ela é usada.

A notação Big-O expande essa lógica ao assumir que, para qualquer algoritmo, ela sempre será maior ou igual ao custo, nos dando sempre o pior caso e, com ele, o limite de tempo de execução. Aplicando a notação:

|  |  |
| --- | --- |
| **Custo** | **Notação Big-o** |
| n + 7 | O(n) |
| n + n²+7 | O(n²) |
| n² + n³ + 7 | O(n³) |
| n + n² + n³ + 7 | O(n³) |

Foram mencionados outros tipos de análise assintótica:

* Notação Big-Ômega (Big-𝞨)
* Notação Big-Theta (Big-𝞡)
* Notação Pequeno-o
* Notação Pequeno-Ômega

Mais informaçõe sobre esses tipos de notações nos slides da aula.

Foram explicadas classes de complexidade de problemas comums:

* **O(1) Ordem constante:**

Independente do tamanho da entrada. Custo fixo;

* **O(log(n)) Ordem logarítmica:**

Típica de algoritmos que resolvem um problema

transformando-o em problemas menores;

* **O(n) ordem linear:**

Em geral, uma certa quantidade de operações é

realizada sobre cada elemento de entrada (um for loop);

* **O(n(log(n)) ordem log linear:**

Algoritmos que trabalham com particionamento dos dados; Resolvem um problema transformando-o em problemas menores, que são resolvidos de forma independente e depois unidos;

* **O(n²) ordem quadrática:**

Normalmente ocorre quando os dados sãoprocessados aos pares (for loop dentro de um for loop)

* **O(n³) ordem cúbica:**

É caracterizado pela presença de três estruturas de repetição (for

loop) aninhadas;

* **O(2n) ordem exponencial:**

Geralmente ocorre quando se usa uma solução de força bruta. Não são úteis do ponto de vista prático;

* **O(n!) ordem fatorial:**

Geralmente ocorre quando se usa uma solução de força bruta. Não são úteis do ponto de vista prático. Possui um comportamento muito pior que o exponencial.

|  |  |  |  |
| --- | --- | --- | --- |
| f(n) | n = 10 | n = 50 | n = 100 |
| n | 1.0E-05 segundos | 5,0E-05 segundos | 6,0E-05 segundos |
| n log n | 3,3E-05 segundos | 2,8E-04 segundos | 3,5E-04 segundos |
| n² | 1,0E-04 segundos | 2,5E-03 segundos | 3,6E-03 segundos |
| n³ | 1,0E-03 segundos | 0,13 segundos | 0,22 segundos |
| 2n | 1,0E-03 segundos | 35,7 anos | 365,6 séculos |
| 3n | 5,9E-02 segundos | 2,3E+08 seculos | 1,3E+13 séculos |

**04/10 (Sexta-Feira)**

Programação Orientada a Objetos (PO):

Foi introduzido o Java, uma tecnologia para o desenvolvimento de Softwares Multiplataforma, através da Máquina Virtual do Java (jvm).

O código-fonte é salvo no formato .java, após compilado pelo Java Compiler (java), ele se torna o bytecode com formato .class, finalmente, a jvm é chamada para executar o programa.

JDK é o uma coleção de binários, bibliotecas etc para necessária para desenvolvedores de Java. Contido dele está o JRE (também baixável separadamente) que é uma coleção de APIs e contém a máquina virtual para a execução de programas Java.

IDEs, idealmente instalados depois do JDK, são programas para o desenvolvimento em java, que facilitam a criação e gerenciamento de projetos e a compilação e execução de programas. É recomendado o [NetBeans](https://netbeans.apache.org/front/main/download/index.html) ou o [IntelliJ IDEA](https://www.jetbrains.com/idea/) (Ctrl + Click para acessar o link).

Tarefa no Moodle até dia Quinta 17/10 às 23:59: Prática 1 do Java

**07/10 (Segunda-Feira)**

Gestão de Pessoas (GP):

Não houve aulas. Serão repostas a partir de um horário de aula extra nos dias 14/10 e 21/10. Isso significa que as aulas passarão do horário do ônibus.

Modelagem de Software (MS):

Modelos de Software são uma importante ferramenta para a concepção, planejamento e construção de qualquer software, facilitando a visualização de como o Software funciona em geral, fornecendo um guia para o desenvolvimento e documentando o processo. Essas matéria se relaciona diretamente com a Programação Orientada a Objetos e seus conceitos.

É diferenciada Análise de Projeto:

Análise é o primeiro passo e se trata de uma investigação feita com e para o cliente. Nela indentificaremos os problemas e os objetos (da perspectiva de PO).

Projeto é a parte mais técnica e se trata diretamente com o programador. O projeto será usado como uma base direta para a programação.

Note que os modelos UML que criaremos serão bem parecidos com os que fizemos para Banco de Dados. Em relação a Objetos, nós iremos desenvolver uma compreensão mais intuitiva deles conforme programamos com linguagens OO.

**08/10 (Terça-Feira)**

Gerência de Configuração de Software (GC):

Como os slides foram, para mim, incompreensíveis, meus resumos do conteúdo das próximas aulas serão, primairamente, traduções ou resumos do livro *SCM Handbook Third Edition* por Alexis Leon. Recomendo ler esse livro em caso de qualquer dúvida com a matéria.

Já aprendemos que GCS lida com as mudanças que ocorrem durante o processo de desenvolvimento de Software. Esse processo produz os seguintes itens que, coletivamente, são chamados de **Configuração de Software**:

* Programas (código-fonte, programas executáveis, bibliotecas, etc.)
* Documentação (requisitos do software, análise de sistemas, design de sistemas, planos, scripts e especificações para testes, manuais de instalação e de usuários e *release notes)*
* Dados (dados de teste e do projeto)

Alguns desses itens são desconhecidos para nós neste atual momento do curso.

O objetivo da gerência de configuração de software é a gravação das propriedades e relacionamentos desses itens, o que seria fácil se eles não estivem sujeitos a constante mudanças.

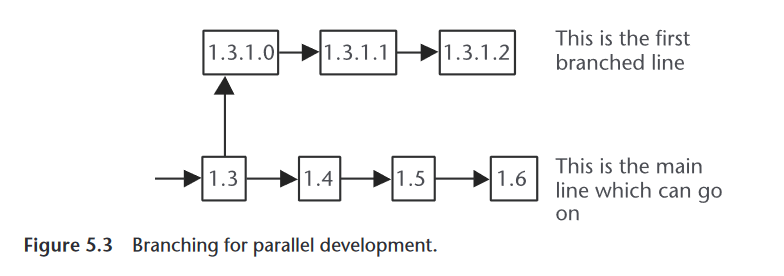
**Informações úteis para a tarefa:**

Existem várias páginas de informações no livro que foram omitidas por propósito de brevidade. Observe a pasta “Leitura Obrigatória” no Moodle para mais informações.

* Item de Configuração (CI): é o elemento fundamental da configuração de software. Um CI pode ser um grupo de programas, uma biblioteca, uma função, documentação do projeto, manuais de usuário, dados de teste, entre outros.
* Versão: quando os CI atingem um estado onde eles condizem com as especificações do projeto, eles são revisados, aprovados e movidos para um ambiente controlado, produzindo uma versão. Se e esses itens forem alterados, as mudanças serão testadas, revisadas e aprovadas novamente, produzindo uma nova versão. Versões tipicamente são criadas através de mudanças para versões passadas. Versões que são equivalentes em funcionalidade mas funcionam em ambientes diferentes (como uma para Windows e uma para Linux) são chamdas de *Variantes*.
* Baseline: a configuração de software em um momento específico é chamado de baseline. Novas baselines tipicamente são estabelecidas no fim de cada fase no Ciclo de Vida do Software. Elas são fundamentais para a GCS, e servem como um ponto de partida e de referência para mudanças futuras. Mudar um programa fora da baseline é fácil e rápido, mas mudar um programa que faz parte da baseline requer procedimentos mais demorados.

Formaremos uma compreensão mais completa de baselines conforme praticamos a GCS.

* Branching, Codeline, Mainline e Merging: Na GCS, um objeto pode ser duplicado, e modificado separadamente e em paralelo com o objeto principal. O processo em si é chamado de *branching* e os objetos duplicados normalmente são chamados de *branches* ou *codelines*, e são considerados a *child* (filhos)do original que é considerado o *parent* (parente ou pai). Mainline se refere à linha principal, ou seja, a que não tem nenhum filho. A figura abaixo demonstra o processo.



*Branches* normalmentes são temporários e acabam sendo reintegrados no programa principal. Esse processo se chama *merging*. *Branches* que não serão reintegrados são chamados de *forks*. É comum que programas tenham uma *branch* exclusivamente para desenvolvimento, chamada de *trunk*.

* Release: *release* é uma coleção de arquivos permitindo a instalação e setup do programa, também contendo o número da release ou versão, os requisitos para a instalação, um guia para a instalação, instruções para atualização, uma chave ou senha se necessário, uma lista dos problemas conhecidos da versão, uma lista de todas as mudanças entre essa versão e anteriores e instruções para contato com suporte técnico caso existam problemas. Hoje em dia a grande maioria dessas informações estão presentes no programa de instalação.

Banco de Dados (BD):

Nessa aula observamos comandos básicos de SQL para a criação e manipulação de tabelas. Scripts disponíveis no Moodle.

**09/10 (Quarta-Feira)**

Essa é a primeira de várias aulas que trataram com o uso de Views. Foi realizada uma ativdade prática para a criação, manipulação e acesso de Views. Scripts disponíveis no Moodle.

Algoritmos e Estruturas e Dados (AE):

Foram realizados vários exercícios envolvendo a análise de algoritmos básicos.

**10/10 (Quinta-Feira)**

Todas as aulas foram canceladas devido ao Festival da Canção.

**11/10 (Sexta-Feira)**

Programação Orientada a Objetos (PO):

Reintrodução dos conceitos básicos de programação, como declaração de Variáveis e Métodos (*chamados de funções no C*), assim como um começo na PO com a criação de Classes.

16 exercícios no Moodle. Entrega até dia 01/11.

**14/10 (Segunda-Feira)**

Gestão de Pessoas (GP):

Foi introduzida a matéria e conceitos básicos de Gestão de Pessoas, e previsto que faremos vários trabalhos ao longo do semestre.

Provavelmente veremos os trabalhos em mais detalhe semana que vem.

Modelagem de Software (MS):

Aula assíncrona, exercícios no Moodle.

**15/10 (Terça-Feira)**

Gerência de Configuração de Software (GC):

Temos um novo professor. Reintroduzida a matéria e os conceitos de GC, como branches, Mainline, etc.

Banco de Dados (BD):

Aprendemos a linguagem PL/pgSQL, uma linguagem de programação feita para ser usada dentro do SQL. Praticamos a criação e execução de funções (que retornam dados) e procedimentos (que não retornam dados). Realizamos alguns exercícios envolvendo a linguagem. Scripts e exercícios no Moodle.

Funções e procedimentos possuem várias outras diferenças que não foram abordadas na aula.

**16/10 (Quarta-Feira)**

Banco de Dados (BD):

Continuamos os exercícios sobre PL/pgSQL.

Algoritmos e Estruturas e Dados (AE):

Realizamos mais exercícios sobre análise de algoritmos.

**17/10 (Quinta-Feira)**

Algoritmos e Estruturas e Dados (AE):

Realizamos a implementação da Pilha Estática em C. Código completo no Moodle.

Modelagem de Software (MS):

Foi feita uma apresentação de slides introduzindo a linguagem UML (Unified Modeling Language), que também usaremos para o trabalho futuramente.

**18/10 (Sexta-Feira)**

Programação Orientada a Objetos (PO):

Revisamos a **Classe *String***e seus métodos relevantes, como ***String.charAt(x)*** que retorna o caractere na posição do int *x*, ***String.concat(x)*** que combina a string com outra String *x*, ***String.equals(x)*** que retorna um *boolean* (*true* ou *false*) da compração entre essa string e um qualquer Objeto *x*, ***String.length()*** que retorna um int com o tamanho da *String*, ***String.replace(x, y)***, que substitui todo *char x* por *char y,* ***String.substring(x)*,** que retorna a *String* a partir da posição *int x*, ***String.toLowerCase()* e *String.toUpperCase()*** que retornam a mesma *String* com todas as letras maíusculas e minúsculas e ***String.indexOf(x)*,** que retorna um *int* com a posição do *char x* ou *String x* que você desejar.

Note que esses métodos poussiem retornos e entradas diferentes. Para uma lista completa dos métodos das Strings, visite esse link: <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html> A combinação **Ctrl + Click Esquerdo** em qualquer String em um IDE também te leva para esse link ou um muito similiar.

**21/10 – 25/10 (Semana Acadêmica)**

**28/10 (Segunda-Feira)**

**Aulas canceladas.**

**29/10 (Terça-Feira)**

Gerência de Configuração de Software (GC):

Aula prática sobre a ferramenta Git. Usamos os comandos *git init, git add e git commit*. Um guia completo para as ferramentas Git está disponível nesse link: <https://git-scm.com/docs>

Banco de Dados (BD):

Introdução aos Triggers (Gatilhos), que executam funções de trigger antes (**BEFORE**) ou depois (**AFTER**) as operações de **INSERT**, **UPDATE** e **DELETE** em uma tabela. Elas podem ser executadas para cada linha **(for each row**) ou para cada comando (**for each statement**). As Triggers possuem duas variáveis, **OLD** e **NEW**, que se referem aos dados antes e depois da operação. Funções de Trigger não recebem nenhum parâmetro e retornam o tipo Trigger. Realizamos exercícios com Triggers. Códigos no Moodle.

**30/10 (Quarta-Feira):**

Banco de Dados (BD):

Aula teórica sobre armazenamento de dados em arquivos físicos e introdução à indexação. Na indexação associamos dados com chaves para uma busca mais eficiente. A Referência é um Pointer para o endereço no Disco onde o dado está localizado.

|  |  |
| --- | --- |
| Chave | Referência |

Observamos a organização sequencial, onde cada chave é ordenada de acordo com algum fator, como ordem alfabética ou ordem crescente.

Observamos a organização em Hash onde a posição da chave é determinada usando uma função matemática a partir de valores relacionados à chave. Essa organização pode causar conflitos (duas chaves em posições iguais).

Algoritmos e Estruturas de Dados (AE):

Realizamos a implementação em C da Pilha Dinâmica. Código no Moodle.

**31/10 (Quinta-Feira):**

Algoritmos e Estruturas de Dados (AE):

Realizamos a implementação em C das Filas Estática e Dinâmica. Códigos no Moodle.

Modelagem de Software (MS):

Explicação do Diagrama de Caso de Uso, que será usado na próxima entrega do trabalho.

**01/11 (Sexta-Feira):**

Programação Orientada a Objetos (PO):

Explicação aprofundada de Classes em PO. Conceitos como instanciação, atributos, métodos e métodos estáticos e funções de get e set. Exercícios no Moodle, entrega até dia 08:11 às 23:59.