**Aula # 1**

**Tema 1 –** Conceitos sobre Linguagens de Programação

**Sumário:** Introdução e breves considerações.

A diciplina de T.L.P tem um carácter teórico-prático, justifica-se enquanto suporte técnico-prático a análise e desenvolvimento de programas informáticos.

A área de informática é composto pelos seguintes cursos:

* Técnico de informática;
* Técnico de Gestão de Sistemas Informáticos;
* Técnico de informática e Multimídea.
* Técnico de Informática de Gestão.

Esta área de formação prepara os alunos para o exercício de profissões relacionadas com a instalação, a configuração e a manutenção de redes informáticas e dos equipamentos de apoio às estruturas de redes informáticas e dos equipamentos de apoio à estrutura de redes, procedendo à implementação de níveis de segurança adequados, assegurando a optimização do seu funcionamento e respeitando as normas de segurança, higiene e saúde no trabalho de protecção do ambiente.

Os exercícios a apresentar nas aulas têm dois objetivos fundamentais:

* Fixar as técnicas de programação;
* Adquirir estrututra logica de raciocinio cada vez mais apurada para a pogramação.

A disciplina de TLP compreende os conceitos e boas práticas para a criação de um programa de computador utilizando as técnicas e recursos como variáveis, tipos de dados, modularização, estrutura de controlo, funções, procedimentos, etc.

**ÁREA FUNCIONAL**

Alunos formados no curso de técnico de informática geralmente atuam em diversas áreas relacionadas à tecnologia da informação (TI).

**Suporte Técnico**: Prestam suporte a usuários e empresas, ajudando na resolução de problemas relacionados a hardware, software, redes e outros aspectos tecnológicos.

**Administração de Redes**: Configuram, gerenciam e monitoram redes de computadores, garantindo a segurança e o funcionamento adequado das conexões de internet e intranet.

**Desenvolvimento de Software**: Participam do desenvolvimento, teste e manutenção de aplicativos e sistemas, trabalhando com linguagens de programação, bancos de dados e outras ferramentas de desenvolvimento.

**Manutenção de Hardware**: Realizam diagnósticos, reparos e upgrades em computadores, servidores e outros equipamentos tecnológicos.

**Gestão de Sistemas Operacionais**: Instalam, configuram e mantêm sistemas operacionais, como Windows, Linux e outros, garantindo que estejam atualizados e funcionando de maneira eficiente.

**Segurança da Informação:** Trabalham com medidas para proteger informações e sistemas contra ameaças como hackers, vírus e outras vulnerabilidades.

**Suporte a Sistemas ERP (Enterprise Resource Planning)**: Auxiliam na implementação e manutenção de sistemas integrados de gestão empresarial, que automatizam processos como contabilidade, recursos humanos e logística.

**Empreendedorismo em TI**: Alguns profissionais decidem abrir seus próprios negócios, oferecendo serviços de suporte, consultoria, desenvolvimento de software, entre outros.

**EMPRESAS E ORGANIZAÇÕES**

Técnicos em informática são requisitados por uma ampla gama de empresas e organizações que necessitam de suporte em tecnologia da informação. Algumas das principais categorias de empresas que costumam contratar esses profissionais incluem:

**Empresas de TI e Consultoria**: Companhias especializadas em tecnologia, como desenvolvedoras de software, prestadoras de serviços de TI e consultorias tecnológicas, frequentemente contratam técnicos para suporte técnico, desenvolvimento de software e administração de redes.

**Empresas de Suporte Técnico e Manutenção**: Empresas que oferecem serviços de suporte técnico terceirizado ou que fazem manutenção de equipamentos de informática também são grandes empregadoras de técnicos.

**Indústrias e Manufaturas**: Empresas industriais e de manufatura muitas vezes empregam técnicos para cuidar da infraestrutura de TI, incluindo a automação industrial, sistemas de controle, e manutenção de computadores e redes.

**Setor Público**: Órgãos governamentais e instituições públicas frequentemente têm departamentos de TI responsáveis por manter a infraestrutura tecnológica, onde técnicos de informática são essenciais.

**Instituições de Ensino**: Escolas, faculdades e universidades empregam técnicos de informática para manutenção de laboratórios de informática, suporte a sistemas educacionais e gestão de redes internas.

**Hospitais e Instituições de Saúde**: Na área da saúde, técnicos de informática são necessários para manter sistemas de registro eletrônico de pacientes, equipamentos médicos conectados a redes, e outros sistemas críticos.

**Comércio e Varejo**: Grandes redes de lojas, supermercados e comércio eletrônico utilizam técnicos para cuidar de suas plataformas de vendas online, sistemas de ponto de venda (POS), e redes internas.

**Bancos e Instituições Financeiras**: O setor bancário depende muito da tecnologia para suas operações diárias, como sistemas de transações, segurança da informação, e gestão de dados, fazendo dos técnicos de informática profissionais valiosos.

**Telecomunicações**: Empresas de telecomunicações precisam de técnicos para suportar a infraestrutura de rede, manutenção de servidores e segurança da informação.

**Startups de Tecnologia**: Startups tecnológicas geralmente contratam técnicos para uma ampla gama de funções, desde suporte ao usuário até desenvolvimento de software e infraestrutura.

**PITCH PESSOAL**

Um **pitch pessoal** é uma apresentação curta e impactante que você faz sobre si mesmo, com o objetivo de destacar suas habilidades, experiências, valores e o que você pode oferecer em um contexto profissional.

É uma espécie de "elevator pitch", onde a ideia é comunicar de maneira concisa e convincente por que você é a pessoa certa para uma oportunidade específica, seja uma vaga de emprego, uma parceria, ou até mesmo um networking.

Um pitch pessoal deve ser breve, normalmente entre 30 segundos e 2 minutos, e adaptado ao contexto em que será apresentado. Ele é especialmente útil em entrevistas de emprego, eventos de networking e situações onde é necessário causar uma boa impressão de maneira rápida e eficaz.

**Tarefa**

1 – Dê uma definição para a disciplina de TLP.

Compreende os conceitos e boas práticas para a criação de um programa de computador utilizando as técnicas e recursos como variáveis, tipos de dados, modularização, estrutura de controlo, funções, procedimentos, etc.

2 – Elaborar um Pitch pessoal.

**Exemplo de Pitch Pessoal**: "Olá, meu nome é Ana Silva, sou formada em Engenharia de Software e tenho mais de cinco anos de experiência em desenvolvimento de aplicações web. Trabalhei em projetos que vão desde startups até grandes corporações, sempre focada em criar soluções que melhoram a experiência do usuário. Meu principal diferencial é a capacidade de entender as necessidades dos clientes e traduzir isso em tecnologia funcional e escalável. Estou atualmente em busca de novas oportunidades que me permitam aplicar essas habilidades em projetos desafiadores e inovadores."

**Aula # 2**

**Tema 1 –** Conceitos sobre Linguagens de Programação

**Sumário:** Conceitos básicos sobre softwares.

Históricamente as linguagens de programação estão diretamente relacionadas ao surgimento das primeiras máquinas inteligentes, por meio delas, o intuito sempre foi transformar as sequências lógicas em resolução de problemas.

Diferentes llinguagens de programação para resolver diferentes tipos de problemas.

Programar é conhecer os fundamentos, a lógica, que está na base de todas as linguagens de programação. Com isso, é possível utilizar qualquer linguagem de programação.

O conceito de software refere-se à unidade lógica do computador.

**Software** é o conjunto de ordens e instruções que tornam possível ao computador realizar determinadas tarefas.

Estas instruções codificadas em uma linguagem compreensível pelo computador constituem programas, qua tal como nossos pensamentos e ideias, orientam o nosso comportamento, controlam e regulam o funcionamento do computador.

**Regras sintáticas** consistem num conjunto de normas a seguir que indicam como se deve estruturar o código, ou seja, como se deve construir o código.

**CLASSIFICAÇÃO DOS SOFTWARES**

Os softwares podem ser divididos em software de sistema, software de aplicação e softwares de programação.

**TIPOS DE LINGUAGENS DE PROGRAMAÇÃO**

**Linguagem de Programação** é um conjunto de regras léxicas e sintáticas para se escrever programas.

As várias linguagens de programação podem ser classificadas em três tipos distintos:

* Linguagens Máquina;
* Linguagem Assembly (simbólica);
* Linguagens de Alto Nível.

**Linguagem de Baixo Nível e Linguagens de Alto Nível**

***Linguagem Máquina*** consiste num conjunto de números binários (sequencias de 0 e 1) que são entendidas pela Unicdad Central de Processamento (CPU). Têm a ver com carateristicas intrísecas do processador do computador.

***Linguagem Assembly*** Consiste numa linguagem muito próxima da linguagem de máquina, em que cada opcode (código de operação) se associa a uma palavra (*mnemotécnica – palavra que nos ajuda a lembrar de uma ação realizada pelo comando*).

***Linguagem de Alto Nível*** são linguagens que permitem a especificação de instruções para o processador numa forma abreviada, onde cada instrução representa várias instruções em linguagem de máquina.

São fáceis de aprender, requerem menos tempo para escrever, têm melhor documentação.

***CONCEITOS SOBRE COMPILADOR E INTERPRETADOR.***

Um programa escrito em linguagem de alto nível é um simples texto, texto escrito usando programa do tipo editor de texto, posteriormente descodificado nas correspondentes instruções em linguagem de máquina.

Existem dois percursos para execução do programa escrito em linguagem de alto nível: interpretadores e compiladores.

**Interpretador** lê no programa cada instrução de cada vez, por cada insrução lida, verifica quais são as acções a tomar em linguagem de máquina e executa-as. Converte para linguagem de máquina cada declaração ou função executada.

Um interpretador não cria nenhum outro programa.

**Compilador** lê um certo número de vezes o programa inicial em linguagem de alto nível (texto) e constrói um segundo programa equivalente ao primeiro mas escrito em linguagem de máquina.

**Programa Fonte, Programa Tradutor e Programa Objeto**

**Programa fonte** é o texto que nós escrevemos, utilizando um editor de texto e segundo as regras sintáticas utilizadas na linguagem de programação com que vamos programar.

É um programa legivel, escrito por um programador.

**Programa Tradutor** são programas que traduzem o programa fonte em linguagem de máquina, utilizando para tal o programa objeto criado a partir do programa fonte. Com a utilização dos programas tradutores obtemos um programa que pode ser executado directamente pela máquina (computador).

**Programa Objeto** é o programa que obtemos após termos feito a compilação do programa fonte, utilizando um compilador, ou seja, após ter sido traduzido o programa fonte.

Tarefa

1 – Escreva o ranking das linguagens de programação atualmente.

2 – Dê exemplos de linguagens de programação interpretadas e compiladas.

3 – Classifique as linguagens em 2 grupos, de propósito geral e linguagens específicas.

**Aula # 3**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Sintaxe de C#.

Um programa pode ser visto como algo que transforma informação.



Para desenvolvermos algoritmos computacionais utilizamos um modelo de programação. Um ***modelo de programação*** fornece idéias e conceitos para nos ajudar a criar algoritmos.

A concepção de um programa obedece uma sequencia de 4 fases:

**1º Formulação do problema:** estudar o problema de forma a encontrar um modelo que o represente corretamente.

**2º Resolução do problema:** obter um algoritmo que conduz a solução do problema.

**3º Tradução:** traduzir o algoritmo numa linguagem de programação escolhida.

**4º Teste:** efetuar conjunto de testes com o objetivo de verificar se o programa cumpre os objetivos pretendidos.

**Pseudocódigo** é uma forma de representação de algoritmos que se assemelha a linguagens de programação mas que utiliza a língua nativa do utilizador de forma a ser facilmente entendida por quem não tem quaisquer conhecimentos da sintaxe de uma linguagem de programação.

Os programadores cuja língua nativa é português, costumam referir-se ao pseudocódigo como Portugol, também conhecido por “Português Estruturado”.

Os algoritmos são representados usando 4 formas fundamentais:

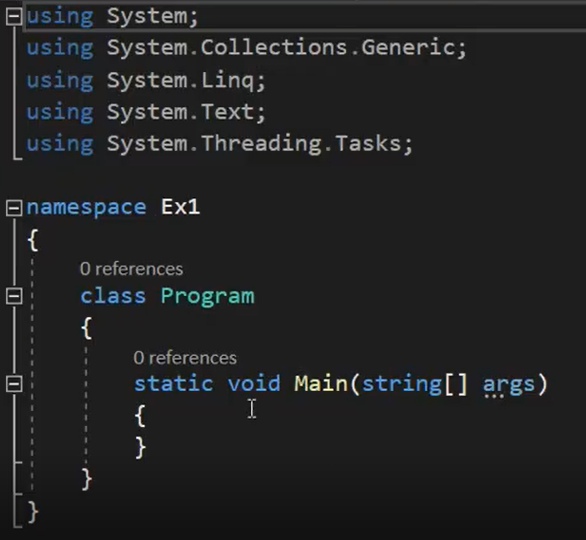
Descrição narrativa;

Fluxograma ou diagrama de fluxo;

Diagrama de Chappin;

Pseudocódigo.

Um ***comando*** é uma ordem ou instrução dada a um computador ou qualquer outra máquina automatizada.



**INSTRUÇÕES DE SAIDA**

1º Console.Write();

2º Console.WriteLine();

Usando frases: Console.Write(“ nossa frase”);

Usando variável:

1. Console.Write(variável);
2. Console.Write(variável1 +” ”+variavel2);
3. Console.Write($ “{variável}”);

**INSTRUÇÕES DE ENTRADA**

A entrada padrão é para leitura de dados do tipo string (texto).

1º Console.ReadLine();

Para outros tipos de dados, precisamos efetuar conversão de dados.

2º Convert.ToInt(Console.ReadLine());

3º int.Parse(Console.ReadLine()) e int.TryParse()

**Exercícios**

1 - Cria um programa que solicite o nome e a idade do usuário e apresenta os dados na tela.

**Aula # 4**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** IDE.

**IDE** (*Integrated Development Envoirment – Ambiente de Desenvolvimento Integrado*) é uma ferrramenta (software) usado no desenvolvimento de aplicações que combinem diferentes funcionalidades em única interface gráfica de usuários.

Seu uso pode impactar positivamente na produtividade e otimização das demandas de um negócio, maximizando sua competitividade no mercado. IDEs ajudam os desenvolvedores a programar novas aplicações de forma rápida.

É possível desenvolver aplicações sem uma IDE.

As funções principais dentro da IDE são:

**1 – Editor de código fonte:** é um editor de texto que auxilia na criação de código de software por meio de funcionalidades como destaque da sintaxe com indicadores viisuais, recurso de preenchimento automático específicos da linguagem e verificação de bugs (erros) durante o desenvolvimento.

**2 – Compilador** são tarefas como compilação de código fonte em código binário, criação de pacotes de código binário e execução de testes automatizados.

**3 – Debugger** é um programa usado paa testar ouros programas e mostrar graficamente a localização do bug no código original.

Exemplos de IDES.

C# é uma linguagem de programação orientado a objetos baseado em classes e orientado a componentes.

|  |  |
| --- | --- |
| **IDE PARA C#** | |
| Computador | Telefone |
| Visual Studio | C# Shell |
| Visual Studio Code |  |

**Aula # 4**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Tipos de dados em C#

O c# separa os tipos de dados em dois grupos, tipo valor (**value types**) e tipo referência (**reference types**).

Os tipo valor incluem tipo simples (int, float, bool, char), tipo enum (enumerações), struct (estrutura) e nullable.

Os tipo referência incluem class, interface, arrays (vetores) e delegates.

Algumas diferenças entre tipo valor e tipo referência são:

1º O tipo valor mantém os dados dentro da sua própria alocação de memória, enquanto o tipo refrência contém um ponteiro para outro local da memória que mantém os dados reais.

2º As variáveis do tipo referência são armazenadas no heap (memória dinâmica) enquanto as variáveis do tipo valor são armazenados no stack (memória estática).

3º As variáveis alocadas na stack são tratadas quando o programa é compilado enquanto as variáveis alocadas no heap são tratadas em tempo de execução.

Alguns tipos de dados simples mais utilizados são:

Char – System. Char (carater unicode de 16bits)

Bool – System.Boolean (true ou False)

String – System.String (conjunto de caracteres unicode)

Int – System.Int32 (inteiro de 32 bits)

long – System.Int64 (inteiro de 64 bits)

short – System.Int16 (inteiro de 16 bits)

float – System.Single (ponto flutuante de 32 bits)

double – System.Double (ponto flutuante de 64 bits)

decimal – System.Decimal (inteiro de 96 bits)

byte – System.Byte (inteiro de 8 bits)

**Variável** é um nome que se dá a um local na memória que armazena um certo tipo de dados (informação).

Os tipos de informações (dados) a considerar na programação são letra, número inteiro, ponto flutuante, palavra e lógico (verdadeiro e falso).

Cada variável está associada a um tipo de dado, nome (identificador) e um valor.

Ex: int idade; System.Int32 idade = 3;

C# é case sensitive, ou seja, diferencia letras maiusculas das minúsculas.

**Constantes** são variáveis que uma vez definida o seu valor, ela não é alterada ao longo da execução do programa.

Declaramos constantes com a palavra reservada **const**.

Ex: *const bool cafe = true;*

Uma constante deve ser sempre inicializada em sua declaração, caso contrário será gerado um erro.

Para sabermos o tipo de dados de uma variável, usamos a função **GetType().**

Exercício: cria um programa que apresenta o tipo de dados de uma variável.

**CONVERSÕES DE DADOS**

Existem dois tipos de conversões (casting): explícita e implicita.

**Conversão explícita** é quando uma variável pode ser mais de um tipo.

Ex:

float x;

Double y = 4.3;

X = (float)y;

**Conversão implícita** ocorre de forma transparente nas atribuições de vaariáveis e passagem de parâemtros de métodos.

**Exercícios**

1 – Desenvolva programas para auxiliar nas seguintes tarefas:

1. Resolver equação do 2º grau.
2. Executar um caso notável a sua escolha.

**Aula # 5**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Operadores.

Na programação, os operadores são classificados em:

Operadores aritméticos;

Operadores relacionais;

Operadores lógicos.

Operadores aritméticos executam operações aritméticas entre dois ou mais operandos.

Adição

Subtração

Multiplicação

Divisão

Resto da divisão

Divisão inteira

Além destes operadores, também é utilizado o incremeto e o decremento.

Incremento: adiciona unidade ao operando. Decremento reduz unidade ao operando.

Ambos podem aparecer pré-fixado (++x), acrescenta antes da próxima instrução ou pós-fixado (y--) diminui depois da próxima instrução.

Ex:

int x = 10;

Console.Write(++x); ==== 11

Console.Write(x); ==== 11

Console.Write(x++); ==== 11

Console.Write(x); ==== 12

Os operadores aritméticos de atribuição (atribuição composta) combinam uma operação aritmética e uma de atribuição.

Ex:

int x = 10;

x = x+2; ou x+=2;

Console.Write(x);

**ESTOURO ARITMÉTICO E DIVISÃO POR ZERO**

Quando o resultado de uma operação aritmética está fora do intervalo dos valores finitos possíveis do tipo numérico envolvido, o comportamento do programa é incerto, podendo apresentar as seguintes saidas:

OverflowException – operação fora dos limites de dados admissiveis.

DividedByZeroException – erro de divisão por zero.

As operações aritmétics com os tipos de dados float e double nunca geram uma exceção. Elas geram resultados que representam valores especiais que são:

INFINITY ou ∞ – quando o divisor da operação é o numero zero.

IsNaN (is not a number ) – o valor não é número.

Ex:

double x = 1.0/0.0;

Console.Write(x); ========== INFINITY

Console.Write(double.IsInfinity(x)); ========== True

**FUNÇÕES MATEMÁTICAS ESPECIAIS**

Fazem parte da biblioteca using System, levam apalavra Math.operação().

**Math.Sqrt();** raiz quadrada de um número.

**OPERADORES RELACIONAIS (COMPARAÇÃO)**

Servem para comparar dois ou mais operandos devolvendo como resposta um valor lógico (true ou false). Elas geram uma proposição lógica (Ex: 7˃12)

Igualdade, desigualdade e diferença.

== igual a...

˃ maior do que...

˃= maior ou igual a...

˂ menor do que...

˂= menor ou igual a...

!= diferente de...

**OPERADORES LÓGICOS**

Servem para comparar dois ou mais proposições devolvendo como resposta um valor lógico (true ou false).

|  |  |  |  |
| --- | --- | --- | --- |
| **Operação** | **Operador** | **Símbolo** | **Condição verdadeira** |
| Conjunção | E / AND | && | Se todas as proposições forem verdadeiras. |
| Disjunção | Ou / OR | **||** | Se pelo menos uma das proposições forem verdadeiras. |
| Negação | Não / NOT | ! | Inverte o valor |
| Disjunção exclusiva | OU Excl / XOR | ^ | Se as proposições tiverem valores lógicos contrários. |

**Exercícios**

1 – Desenvolva programa para apresentar uma tabela de verdade de um dos operadores lógicos.

**Aula # 6**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Instruções condicionais.

Controlo de fluxo de um progrma gera um desvio na sequencia linear do programa permitindo escolher diferentes abordagens dependendo do resultado de uma dada proposição ou condição.

As instruções condicionais ajudam a identificar que ações a tomar dependendo do resultado da condição em análise.

As instruções condicionais são: if (if...else), switch...case, operador ternário (? : )

**Instrução condicional IF**

Sintaxe do if:

|  |  |
| --- | --- |
| If (condição){  Instruções;  } | If (condição){  Instruções;  }else{  Instruções;  } |

1º A condição é avaliada;

2º Se o resultado da condição for verdadeiro, executa-se as instruções do bloco if.

3º Se o resultado da condição for falsa, executa-se as instruçoes do bloco else, se existir.

**Exercícios**

1 – Desenvolva programa para apresentar o resultado final de um aluno (apto/não apto) caso sua media final seja superior a 9 valores.

2 – Desenvolva um programa que solicita um número ao usuário e mostra se o mesmo é par ou ímpar.

**Instrução condicional SWITCH...CASE**

Sintaxe do if:

Switch(**variavel**){

Case 1:

Instruções1;

Break;

Case 2:

Instruções2;

Break;

Default:

Instruções;

}

1º A variável indica o elemento a ser comparado nos cases;

2º Cada case representa a instrução a ser seguida se o seu valor corresponder a variável em análise.

3º Cada case deve ser terminado com a instrução break, senão todos os cases serão executados.

4º A instrução default é opcional, não termina com break e somente é executado caso nenhum dos cases for verdadeiro.

A instrução switch...case usa-se nas situações em que o numero de possibilidades para a tomada de decisão é elevada (maior que 2) de formas a reduzir a complexidade do uso do if...else consecutivos e encadeados.

**Exercícios**

1 – Cria um programa que possibilite ao usuário escolher seu estado civil (solteiro, casado, divorciado, viuvo).

**OPERADOR TERNÁRIO**

**Condição ? expressão 1: expressão 2;**

Variável = **Condição ? expressão 1: expressão 2;**

1º A condição é avaliada;

2º Se o resultado for verdadeiro, executa a primeira expressão;

3º Se o resultado for falso, é executado a segunda expressão.

**Exercícios**

1 – Cria um programa que acrescenta 5% ao salário do funcionário se este for acima de 100.000 kz do contrário acrescente 15%.

**Aula # 7**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Instruções de repetição.

Controlo de fluxo de um progrma gera um desvio na sequencia linear do programa permitindo escolher diferentes abordagens dependendo do resultado de uma dada proposição ou condição.

As instruções de repetição servem para repetir um bloco de instruções (código) o número de vezes em que a condição é verdadeira.

REPETIÇÃO = CICLO = ITERAÇÃO

As instruções cíclicas são: for, while, do...while e o foreach.

Inicialização – determina onde o ciclo começa;

Condição – determina onde o ciclo acaba (quando o valor lógico se torna falso)

Incremento – salto para próxima operação a ser feita.

**Instrução cíclcica FOR**

Se o número de iterações for conhecida (fixa) é recomendável o uso do for.

For (inicialização; condição; incremento){

Lista de instruções;

}

**Exercícios**

1 – Usando o ciclo for, cria um programa que lista os números de 1 a 20.

Se o número de iterações não for conhecido (fixoa) é recomendável o uso do while, do...while e foreach.

**Instrução cíclcica WHILE**

Se precisarmos que o loop executa apenas se a condição for verdadeira.

Inicialização;

while (condição){

Lista de instruções;

Incremento/decremento;

}

**Instrução cíclcica DO...WHILE**

Se precisarmos que o loop executa pelo menos uma vez antes de avaliar se a condição é verdadeira.

Inicialização;

do {

Lista de instruções;

Incremento/decremento;

}while(condição);

**Aula # 8**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Indentação e comentários.

Indentação serve para organizar o código deixando explícito as dependências que as instruções têm uma das outras.

Comentários é um texto que escrevemos no código para melhor explicar as instruções aplicadas.

Ambos não têm qualquer efeito na compilação e execução do programa. É uma característica pessoal de cada programador.

**Aula # 11**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Funções.

**Aula # 12**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Armazenamento em ficheiros.

PORTUGOL

inclua biblioteca Arquivos --> a

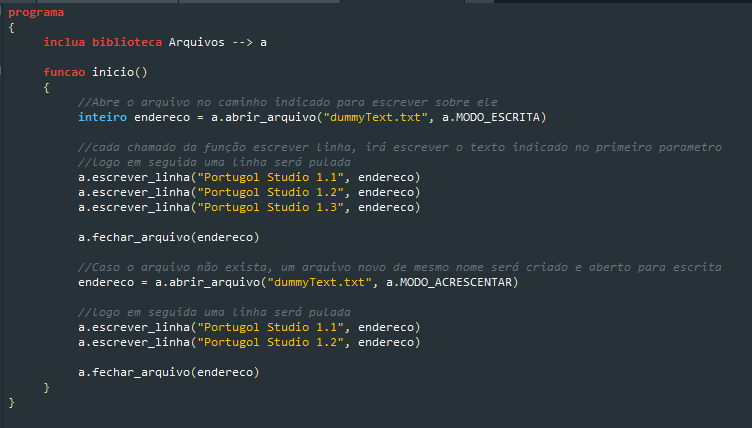
Modos de Abertura:

- MODO\_ESCRITA

- MODO\_LEITURA

- MODO\_ACRESCENTAR

Após ler os dados desejados, é necessário fechar o arquivo. É necessário fazer isso para liberar a memória que está sendo utilizada pelo arquivo e também para que outros programas possam acessá-lo.





VISUAL G

arquivo “caminho”

**Aula # 9**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Arrays (vetores e matrizes).

Array é um grupo de elementos de tipo similar reservadas numa posição de memória.

Para declar um arrayusa-se a seguinte sintaxe:

Tipo[] nomeArray = new Tipo[quantidade];

Ex: char[] letras = new char[5];

Existe três formas de inicializar os arrays ao longo da sua declaração:

1º Forma completa: Tipo[] nomeArray = new Tipo[quantidade]{elem1, elem2};

2º Omitindo a quantidade de elementos: Tipo[] nomeArray = new Tipo[]{elem1, elem2};

3º Omitindo o operador new: Tipo[] nomeArray = {elem1, elem2};

Os elementos do array são identificados a partir da sua posição de memória. A primeira posição equivale a posição 0.

Array[0] – 1ª posição

Array[1] – 2ª posição

Array[2] – 3ª posição

A inicializaçao, também pode ser feito diretamente nas suas posições.

Array[0] = 2;

A leitura de dados de um array geralmente é feito usando um ciclo, normalmente usando o for e o foreach.

**Exercícios**

1 – Cria um programa que apresenta os elementos de um array sendo estes 10, 20, 30, 40 e 50.

**VETORES E MATRIZES**

Os arrays podem ser unidimensionais ou multidimensionais.

Os arrays multidimensionais são declarados da seguinte forma:

Tipo[ , ] nomeArray = new Tipo[nº linhas, nº colunas];

A inicialização dos arrays multidimensionais (matrizes) obedece a sintaxe usada para arrays unidimensionais (vetores).

1º Forma completa: Tipo[,] nomeArray = new Tipo[nº linhas, nº colunas]{{elem1, elem2}, ]{{elem1, elem2}};

2º Omitindo a quantidade de elementos: Tipo[,] nomeArray = new Tipo[]{{elem1, elem2}, ]{{elem1, elem2}};

3º Omitindo o operador new: Tipo[,] nomeArray = {{elem1, elem2}, ]{{elem1, elem2}};

Os elementos do array multidimensionais são identificados usando como referência o número da linha e da coluna.

Ex: nomeArray[nº linha, nº coluna] = elemento;

A leitura de dados de um array geralmente é feito usando um ciclo, normalmente usando o for e o foreach.

**Exercícios**

1 – Cria um programa que preencha uma matriz 3x3 com valores a sua escolha.

**Aula # 10**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** ArrayList.

Array são coleções de dados de um mesmo tipo cujo tamanho não podemos aumentar depois de declarado, são estáticos. Ela utiliza o namespace System que já vem declarado por padrão.

ArrayList é uma coleção de dados não-genérica em que o tamanho aumenta dinâmicamente. Ela é utilizada quando se deseja adicionar dados em que desconhecemos o tipo e o tamanho de dados.

Para usar arrayList devemos incluir o namespace System.Collections;

Sintaxe para declaração de ArrayList:

ArrayList nomeArray = new ArrayList();

ou

var nomeArray = new ArrayList();

Para inicializar ArrayList podemos utilizar as seguintes formas:

1º var nomeArray = new ArrayList(){elem1, elem2, elem3};

2º Usando referência das posições. M[1] = 9;

3º Usando a função Add(). Ex: M.Add(5);

Um ArrayList pode ter dados do tipo NULL e valores duplicados.

Para aceder os elementos do ArrayList podemos usar a referência das posições e/ou instruções de repetição.

**Exercícios**

1 – Cria um programa que tenha um arrayList declarado e inicializado com valores a sua escolha e um ciclo para apresentar os mesmos dados a tela.

Alguns métodos de manipulação de ArrayList são:

1º Add() – adiciona um valor na lista.

2º AddRange() – adiciona uma coleção dentro da lista.

3º Insert() – insere um elemento numa posição específica da lista.

4º InsertRange() – adiciona uma coleção numa posição específica da lista.

5º Remove() – remove um elemento específico na lista.

6º RemoveRange() – remove uma quantdade de elementos pelo intervalo ed endereço.

7º RemoveAt – remove um elemento numa posição específica.

8º Contains – verifica se um elemento existe na lista.

9º Clear() – Remove todos os elementos da lista.

10º IndexOf() – retorna o valor da posição de um elemento da lista, senão retorna -1.

**Aula # 11**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** List.

Array são coleções de dados de um mesmo tipo cujo tamanho não podemos aumentar depois de declarado, são estáticos. Ela utiliza o namespace System que já vem declarado por padrão.

ArrayList é uma coleção de dados não-genérica em que o tamanho aumenta dinâmicamente. Ela é utilizada quando se deseja adicionar dados em que desconhecemos o tipo e o tamanho de dados.

Para usar arrayList devemos incluir o namespace System.Collections;

Sintaxe para declaração de ArrayList:

ArrayList nomeArray = new ArrayList();

ou

var nomeArray = new ArrayList();

Para inicializar ArrayList podemos utilizar as seguintes formas:

1º var nomeArray = new ArrayList(){elem1, elem2, elem3};

2º Usando referência das posições. M[1] = 9;

3º Usando a função Add(). Ex: M.Add(5);

Um ArrayList pode ter dados do tipo NULL e valores duplicados.

Para aceder os elementos do ArrayList podemos usar a referência das posições e/ou instruções de repetição.

**Exercícios**

1 – Cria um programa que tenha um arrayList declarado e inicializado com valores a sua escolha e um ciclo para apresentar os mesmos dados a tela.

**Aula # 11**

**Tema 2 –** Programação numa linguagem de alto nível.

**Sumário:** Versionamento e repositórios.