**WordSkills- Folha 2**

**Pesquisa atividade 1:** Os 3 tipos de dados usados em sistemas de informações são:

**.txt** – Texto simples, muito usado para logs, configurações básicas ou testes.

**.csv (Comma-Separated Values)** – Armazena dados tabulares em formato texto, separado por vírgulas. Muito comum para importações/exportações de sistemas.

**.json** – Leve e legível, ideal para armazenar dados estruturados, configuração de sistemas ou comunicação via APIs.

**Pesquisa 2:** Buffer, na leitura de arquivos, é uma área temporária da memória usada para ler os dados em partes ao invés de carregar o arquivo inteiro de uma vez, o que melhora o desempenho e economiza memória. Ao usar um buffer, o sistema lê blocos (como 1 KB ou 4 KB por vez) do disco e entrega ao programa, permitindo o processamento eficiente mesmo de arquivos grandes.

**Pesquisa atividade 2:** Em manipulação de arquivos, os modos Write, Append e Read definem como o arquivo será acessado:

**Read (leitura):** Abre o arquivo apenas para leitura. Você não pode escrever ou modificar. Se o arquivo não existir, ocorre erro.

**Write (escrita):** Abre ou cria o arquivo e apaga todo o conteúdo anterior, começando do zero. Ideal para sobrescrever arquivos.

**Append (acréscimo):** Abre ou cria o arquivo e mantém o conteúdo existente, adicionando novos dados ao final.

**Pesquisa atividade 3:** O conceito de stream representa a transferência sequencial de dados, como uma “corrente” que flui entre uma fonte (ex: disco, rede) e um destino (ex: programa, outro arquivo). Em C#, streams são usados para ler ou gravar dados byte a byte ou em blocos, de forma eficiente e contínua, sem precisar carregar tudo de uma vez na memória.

**Pesquisa atividade 4:** Arquivos grandes (big files) são arquivos com tamanho considerável, como logs extensos, bases de dados em CSV, ou arquivos de exportação de sistemas. Eles são grandes o suficiente para que carregar tudo de uma vez na memória (RAM) seja ineficiente ou até impossível, especialmente em sistemas com pouca memória disponível. Com File.ReadLines(), você lê uma linha por vez, mantendo baixo uso de memória e permitindo processamento contínuo sem sobrecarregar o sistema.

**Pesquisa atividade 5:** Em sistemas operacionais, permissões de pastas controlam quem pode acessar, modificar ou executar arquivos dentro delas. As permissões podem variar um pouco entre sistemas

**Pesquisa atividade 6:** Quando você exclui um arquivo normalmente (via Explorer), ele vai automaticamente pra lixeira. No código C#, se você usar File.Delete(), o arquivo não vai para a lixeira, ele é removido diretamente do disco.

**Pesquisa atividade 7:**