Atividade integrativa: Comparação dos Métodos de Alocação de Memória.

Tipos de Alocação de Memória e Suas Aplicações:

1. Memória Estática

- Exemplo no código: static char categorias[3][20]
- Quando usar: Para dados que não mudam durante a execução e permanecem alocados o tempo todo.
- Impacto no desempenho: Acesso rápido e eficiente.
- Segurança: Pode desperdiçar memória se mal dimensionado.

2. Memória Automática

- Exemplo no código: char linha[100] dentro de ver_estoque().
- Quando usar: Para variáveis locais dentro de funções.
- Impacto no desempenho: Melhor que memória dinâmica por ser gerenciada automaticamente.
- Segurança: Menor risco de vazamento, mas pode causar stack overflow.

3. Memória Dinâmica

- Exemplo no código: Uso de malloc() para opcao, quantidade, sku, etc.
- Quando usar: Quando o tamanho dos dados pode variar em tempo de execução.
- Impacto no desempenho: Pode ser mais lento devido ao gerenciamento manual.
- Segurança: Risco de vazamento de memória se free() não for chamado corretamente.

Erros e Melhorias:

1. Verificação de fopen()

- O arquivo pode não existir ou ter permissões erradas. Use:

```
FILE *arquivo = fopen("estoque.txt", "r");
if (!arquivo) {     printf("Erro ao abrir
o arquivo!\n");     return; }
```

2. Liberação de memória em ver_estoque()

- fclose() não foi chamado para fechar o arquivo corretamente.

3. Uso excessivo de malloc()

- Variáveis como opcao poderiam ser automáticas para evitar alocação desnecessária.
- Melhor uso de memória e melhor desempenho.

4. Uso inseguro de scanf()

- scanf("%s", nome_produto); pode causar overflow. Melhor usar:

```
scanf("%19s", nome_produto);
```

5. Correção do nome da função

- cadastro_praduto() deve ser cadastro_produto().

6. Validação da entrada do usuário

- Selecione a categoria com um número válido antes de acessar categorias[].

```
if (*opcao_categoria < 0 || *opcao_categoria >= 3) {
printf("Categoria inválida!\n");     continue;
}
```