Sistema de Gerenciamento de Vendas de Smartphones

Introdução

- Projeto de desenvolvimento por: Lucas Henrique, Pedro Gonçalves e Gabriel Carvalho.
- Objetivo do projeto: Controlar e analisar suas transações de venda no mercado de dispositivos móveis.

Funcionalidades Principais

- Cadastro / Remoção
- Busca
- Alteração
- Exportação / Importação de `.csv`
- Consulta de estoque
- Ordenação

Estruturas Utilizadas

- Struct
- Modularização
- Condicionais
- Estruturas de Repetição
- Getline
- SystemClear

Acertos e Erros Durante o Desenvolvimento do Projeto

Erros

- Utilização de vetor normal em vez de utilizar struct
- Esquecimento de apagar o último elemento do vetor
- Leitura incorreta do arquivo binário

Como Resolvemos

Utilização de vetor normal em vez de utilizar struct

- Foi necessário refazer o código para utilizar struct

```
struct Celular
{
  int identificador;
  char nome[30];
  float preco;
  int quantidade;
  char fabricante[10];
  int anoCriacao;
  char descricao[230];
};
```

Esquecimento de apagar o último elemento

- Quebramos a cabeça para perceber que estávamos deixando o for sem o `=`.

```
for (int i = 0; i <= estoqueReal; i++)
  {
    if (celulares[i].identificador == identificador)
    {
      posicao = i;
      contPosicao++;
    }
  }
```

Leitura incorreta do arquivo binário

- Assistimos as aulas e não entendemos nada como aplicar no nosso código
- Foi na tentativa e erro que conseguimos resolver

Passo 1

- Calculamos o tamanho do arquivo e dividir pelo tamanho da struct

```
// utilizando struct com ponteiro para armazenar os dados do arquivo
arquivoIn.seekg(0, ios::end);
int tamanhoArquivo = arquivoIn.tellg();

arquivoIn.seekg(0, ios::beg);

int estoqueMax = tamanhoArquivo / sizeof(Celular);
```

Passo 2

- Criamos um vetor de struct e utilizamos o `new` para alocar a memória

```
Celular *celulares;
celulares = new Celular[estoqueMax];
```

Passo 3

- Utilizamos o `while` para ler o arquivo e armazenar os dados no vetor

```
int estoqueReal = 0;
string linha;

while (estoqueReal < estoqueMax && arquivoIn.read((char *)&celulares[estoqueReal], sizeof(Celular)))
{
  estoqueReal++;
}
```

Acertos

- Modularização: Ajudou a deixar o código mais limpo e fácil de entender
- Utilização adequada das estruturas de controle: Fez com que o código ficasse mais organizado para o usuário ter uma opção de escolha
- Tratamento de erros: Foi utilizado para evitar que o usuário digite um valor inválido

Conclusão

- O projeto foi muito importante para o nosso aprendizado
- Aprendemos a utilizar as estruturas de controle e modularização
- Aprendemos a lidar com trabalho em grupo e ter um código grande