Programação em C/C++

Prof. Márcio Miguel Gomes
marciomg@unisinos.br

Ano/Semestre: 2023/2

Trabalho do Grau B

Apresentação. Os trabalhos serão apresentados pelos grupos diretamente ao professor na aula do dia **27/11/2023**. O tempo máximo para apresentação por grupo será de 20 minutos. A ordem de apresentação será definida no dia da apresentação.

Instruções para envio do trabalho. Enviar somente os arquivos-fonte do projeto para a atividade aberta no **Moodle até às 19h30min do dia 27/11/2023**. Apenas um integrante do grupo precisa enviar os arquivos.

Grupos: devem ser formados grupos de no máximo 2 integrantes.

Contextualização

CVE (Common Vulnerabilities and Exposures) é uma lista de vulnerabilidades de cibersegurança publicamente conhecidas. Os dados do CVE são usados em vários produtos e serviços de segurança cibernética de todo o mundo, incluindo o Banco de Dados de Vulnerabilidades Nacionais dos EUA (NVD). O site www.cvedetails.com disponibiliza uma lista das vulnerabilidades conhecidas, códigos de classificação e demais informações, conforme arquivo em anexo na atividade. Este arquivo está em formato texto, com colunas separadas por um <TAB>.

Programa

Desenvolva um programa que implemente a classe **Registro** descrita abaixo.

```
class Registro {
private:
	int idCVE;
	int idCWE;
	string vulnerabilityTypes;
	Data publishDate;
	Data updateDate;
	float scoreCVSS;
	string gainedAccessLevel;
	string access;
	string complexity;
	string authentication;
	string confidentialy;
	string integrity;
	string availability;
	string description;
public:
	// Construtor
	Registro(string linha);

	// Destrutor
	~Registro();

	// Métodos getter

	// Métodos adicionais
}
```

Programação em C/C++

Prof. Márcio Miguel Gomes
marciomg@unisinos.br

Ano/Semestre: 2023/2

A classe “**Registro**” deve receber no seu construtor uma linha do arquivo de vulnerabilidades, realizar a separação e conversão dos dados e alimentar os atributos do objeto. Os atributos “**publishDate**” e “**updateDate**” devem ser objetos da classe “**Data**”. Essa classe deve receber no seu construtor a string da data e armazenar individualmente o ano, mês e dia como números inteiros. Além disso, deve implementar uma função “**int compare(Data date)**”, que recebe uma data como parâmetro, devolve 0 se “this” e “date” forem iguais, devolve -1 se “date” for anterior a “this” e devolve 1 se “date” for posterior a “this”.

O programa deve ser desenvolvido em uma classe chamada “**Sistema**”. O objeto “**Sistema**” deve receber em seu construtor o nome do arquivo de vulnerabilidades a ser processado. Em seguida, deve abrir o arquivo, percorrê-lo do início ao fim e instanciar um objeto “**Registro**” para cada linha do arquivo (cada linha representa um registro de vulnerabilidade). Os objetos “**Registro**” devem ser armazenados em um **vector** chamado “**dados**” (obrigatoriamente o vector deve ser de ponteiros para objetos “**Registro**”). Todo o trabalho de procura, filtragem, visualização e exportação de dados deve ser feito sobre os objetos do vector “**dados**”. Adicione novos atributos e métodos nas classes “**Registro**”, “**Sistema**” e “**Data**” se necessário. O destrutor das classes “**Registro**” e “**Sistema**” devem liberar os recursos alocados dinamicamente.

Menu

Crie um menu para interação com o usuário com as seguintes funcionalidades:

1. **Localizar por “CVE ID”**: Mostrar na tela todos os dados de um “CVE ID” informado pelo usuário, cada informação em uma linha.
2. **Localizar por “Description”**: Mostrar na tela o “CVE ID”, “Vulnerability Types”, “CVSS Score”, “Publish Date” e “Update Date” de todos os registros que possuam no campo “Description” a *substring* informada pelo usuário. Cada linha visualizada na tela deve apresentar um único registro.
3. **Histograma “CWE ID”**: Gerar na tela um histograma (contagem agrupada por código) listando cada código “CWE ID”, a quantidade e o percentual de registros por código e uma barra gráfica. Solicitar ao usuário a quantidade mínima e máxima da contagem que deve aparecer no histograma.
4. **Histograma “Score”**: Gerar na tela um histograma (contagem agrupada por grupo de código) listando cada grupo de código, a quantidade e o percentual de registros por grupos de código e uma barra gráfica. Os grupos são definidos de 0.0 a 0.9, 1.0 a 1.9, 2.0 a 3.9 ... até 9.0 a 10.0.
5. **Exportar dados**: Solicitar ao usuário um intervalo de “Publish Date” (inicial e final) e um intervalo de “CVSS Score” (inicial e final). Em ambos os casos, garantir que o filtro final seja maior ou igual ao inicial. Aplicar a filtragem e salvar os registros selecionados em um arquivo texto, no mesmo formato do arquivo original. Pedir para o usuário digitar o nome do arquivo a ser salvo.
6. **Sair**: Encerra o programa e libera os recursos alocados.

Observação 1: Todas as opções do menu devem ser implementadas. A não-implementação de alguma opção acarretará um desconto na nota final do grupo.

Observação 2: Os nomes de classes, atributos e métodos especificados acima **devem ser mantidos** na implementação do código, ou seja, não os renomeie. Novos métodos e atributos devem ser nomeados de acordo com a sua respectiva função.