CLI Pokedex

Mateus Cezário Barreto[1], Nicolas Alho[1], Yuri Gabriel Delgado[1]

[1]Universidade Federal do Pará - Instituto de Ciências Exatas e Naturais

mateus.cezario.barreto@gmail.com, nicolas.alho38@gmail.com, yuri.delgado@icen.ufpa.br

Abstract. *Relatório do desenvolvimento da Pokedex CLI, um projeto brinquedo que implementa uma estrutura de dados encadeada (lista encadeada) como contêiner para registros de Pokemons em uma Pokedéx, usando a linguagem C++.*

1 Introdução

CLI Pokedéx é uma implementação de lista encadeada [Morin 2013] em C++, que foi desenvolvida como projeto final da disciplina de Programação I, ministrada pela Profa. Dra. Paula Cardoso. O projeto foi contextualizado na temática de Pokemons, onde cada bloco encadeado contém o registro de um pokemon, e a lista em si é chamada de Pokedéx.

As requisições originais propostas ao projeto incluem o armazenamento e a leitura de estado a partir de um arquivo textual que fosse manualmente editável. Essa tarefa seria substancialmente menos trabalhosa usando uma biblioteca de persistência de dados em formato textual, como o formato Json, e a própria biblioteca padrão do C++ (Standard Template Library ou STL) para obter a estrutura std::list, que implementa uma estrutura encadeada [Brokken 2014]. Nesse caso, o esforço principal seria construir uma interface de linha de comando ou gráfica para que o usuário pudesse efetuar operações de criação, leitura, atualização e remoção dentro da aplicação, conjunto esse de operações conhecido como CRUD (Create, Retrieve, Update and Delete) [Bourke 2019], bem como, adicionalmente, a operação de ordenamento baseada em, pelo menos, dois parâmetros, o que foi igualmente exigido.

No entanto, uma das restrições intencionais é justamente o não uso de estruturas como std::list (para lista encadeada) e std::vector (para

sequência baseada em cache) [Brokken 2014], no intuito de que essa tarefa fosse reimplementada no código, com a conveniente flexibilidação para que os elementos da lista sejam apenas adicionados no final. No espírito didático dessa restrição didática, escolheu-se implementar uma sintaxe básica para leitura e gravação de dados em arquivos, ao invés de usar uma biblioteca, o que só seria viável com um tipo de dado suficientemente simples, sem subtipos ou recursões.

Portanto, a simplicidade da estrutura de atributos foi um dos motivos para a escolha de Pokemons como temática do projeto. Cada registro de Pokemon inclui apenas tipos de texto e tipos numéricos, sendo o registro de um pokemon a única estrutura encadeada, isto é, tirando a necessidade de uma implementação genérica de lista para lidar com vários tipos de dados.

No que tange a escolha de soluções e de ambiente, o projeto foi ambientado no ecossistema de sistemas operacionais baseados em Linux, com uso de ferramentas de código aberto e de material teórico de licença permissiva, sempre que possível.

2 Planejamento

O projeto da CLI Pokedex foi desenvolvido usando o sistema de versionamento Git, e publicado no serviço de hospedagem de código Github desde o começo do desenvolvimento. O planejamento foi relaizado dentro do próprio repositório Git, nos arquivos "readme.md" (língua inglesa) e "readme.pt-BR.md" (língua portuguesa), iniciando com a escolha dos campos de atribuitos de cada para pokemon. Os atribuitos escolhidos abrangeram números inteiros, decimais e texto, conforme a Tabela 1 (tipos conforme a sintaxe da linguagem C++), com o tipo mais complexo sendo a sequência de string. O tamanho de cada vetor de string foi escolhido com base no número máximo de campos possívei(maior registro tem 12 posições), o suficiente para abranger as habilidades, fraquezas, forças e grupos de ovos de um pokemon, sem tornar o registro espaçoso demais, mantendo, portanto, o registro de um pokemon como único tipo de elemento encadeado.

Tabela 1: Atributos de um pokemon

Nome do atributo	Tradução	Tipo
name	nome	string
global_id	ID global	int
weight	peso	float
generation	geração	int
base stat total	total de atributos base	int
abilities	habilidades	std::string[5]
weaknesses	fraquezas	std::string[7]
resistances	resistências	std::string[12]
egg_group	grupo de ovos	std::string[6]

Nas primeiras versões a biblioteca std::vector foi usada temporariamente para focar no desenvolvimento da interface de linha de comando, e depois foi substituiída pela implementação própria de lista encadeada dentro da classe Pokedex.

As versões iniciais também incluiam uso de structs e classes, misturadas. No entanto, o uso de structs foi abandonado em detrimento do uso de classes, apenas, para adotar uma idioma mais próximo de orientação à objetos.

Uma versão paralela da CLI Pokedex, escrita em linguagem C e sem orientação à objetos (structs ao invés de classes, por exemplo), foi planejada, assim como uma interface gráfica para a Pokedex, com o uso da biblioteca Raygui, mas a interface imediata de linha de comando foi o foco principal do projeto. Foram implementados modelos de entrada de linha de comando direta com o uso de flags(ex: pokedex-cli --help), e de inserção de comandos no terminal com uma interface de texto.

Já o sistema de construção escolhido para o programa foi o Meson, pela proximidade da sintaxe dos arquivos de construção do Meson com a linguagem C++, pelo menos no que tange à listas e chamada de função. Ainda assim, visando melhor portabilidade e tendo em vista o pequeno

porte do projeto, o repositório contém instruções para compilação manual apenas com g++ ou clang (comando clang++).

Em relação à padronização, o padrão ISO da linguagem C++ utilizado foi o de 2023 (C++23), conquanto os recursos da linguagem utilizados se ateram ao tradicional, com exceção da estrutura "enum class", introduzida no padrão C++11 (2011). Mesmo a entrada e saída não utilizou os modelos recentes de formatação com uso da função print(), e sim os tradicionais operadores de stream (">>" e "<<"), junto com funções oriundas da linguagem C, principalmente para ocasiões com menos formatação, como as mensagens de error. O processamento de argumentos da interface de comando foi outra ocasião em que funções oriundas de C, como strcmp() para comparação de sequências de caracteres, foram preferidas, em razão da velocidade e simplicidade.

Dentre as principais dificuldades no desenvolvimento, nota-se a organização do código, dado as grandes diferenças de estilo entre os integrantes, diversos bugs(agora corrigidos) envolvendo a atualização de campos dos pokemons, salvamento de arquivo, sort com peso e a adaptação do programa pra interface com texto.

3 Resultados

O programa é capaz de performar operações básicas de CRUD usando a interface em texto, além de poder reordenar os dados. Também é possível usar o programa com parâmetros imediatos passados via linha de comando, com exceção do salvamento de arquivo. Os argumentos de linha de comando podem ser acessados por meio do argumento "--help", como é de praxe em executáveis de linha de comando.

A interface gráfica utilizando Raygui está primitivamente presente em meio ao código, mas não apresenta as mesmas funcionalidades da interface de linha de comando ou da de texto , servindo apenas para carregar as entrada da Pokedéx a partir de um arquivo escolhido e para trafegar na lista encadeada de pokemons em duas direções, visualizando o nome de cada pokemon na tela.

Em relação às classes presentes no programa, além de uma classe para a interface de linha de comando (CommandLinePokedex) e outra para a interface gráfica (GraphicalPokedex), tem-se: uma classe para representar um pokemon e seus atributos (Pokemon); uma para

representar a estrutura encadeada (PokedexPokemonEntry); uma classe homônima para representar a Pokedex e gerenciar a lista encadeada (adicionar e remover itens, por exemplo). Já a versão em C não chegou a ser implementada.

Uma das partes mais demoradas do projeto foi a estrutura de leitura e consumo de arquivo, com a sintaxe estipulada, sendo esta também uma das partes menos testadas. Não foram estipuladas regras sintaticas claras para o que deve ou não estar presente em um registro de pokemon. Ainda assim, a utilização diante dos arquivos de teste gerados automaticamente performa relativamente bem. Esses arquivos estão presentes no repositório, em um diretório dedicado, contando, inicialmente, com um arquivo pequeno feito a mão, mas também com dois arquivos gerados com uso de scripts.

Por fim, o relatório da atividade avaliativa foi escrito dentro do repositório Git, para ser montável assim como o resto do programa. O relatório foi escrito na linguagem de programação Ruby. A implementação do processo de montagem foi implementação própria do projeto. O código autoral inteiro no repositório foi submetida à licença Unlicense, tornando o repositório, efetivamente, domínio público.

4 Referências

Morin, P. (2013). Open Data Structures. AU Press.

Bourke, C. (2019). Computer Science II. https://cse.unl.edu/~cbourke/ComputerScienceTwo.pdf.

Brokken, F. B. (2014). C++ Annotations. ISBN 90-367-0470-7. University of Groningen.