

Universidade Federal do Rio Grande do Sul

João Kenji Suwa - 587808

Felipe Pasinato Rossoni - 587631

Otávio Rosa de Oliveira - 588807

Fase I do Trabalho Prático

Disciplina: Técnicas de Construção de Programas

Porto Alegre

2025

Introdução

O trabalho propõe a implementação de um sistema que funcione como um gerador de música. A partir de uma entrada de texto, o software irá produzir notas correspondentes a caracteres específicos, seguindo uma série de parâmetros previamente definidos. Os integrantes do grupo decidiram utilizar a linguagem Python para a construção do programa, com auxílio de bibliotecas como (MIDIUtil 1.21, pygame 2.6.0, entre outras).

Requisitos do Sistema

Algumas das funcionalidades e a sua respectiva ideia de implementação estão descritas na lista abaixo:

Funcionalidades:

Cada caractere do texto será lido e interpretado pelo programa, a fim de realizar a conversão para uma ação musical. Classes diferentes de ações terão módulos próprios implementados no código. Alguns exemplos:

Leitor de texto:

O texto do usuário é lido e convertido para uma lista de caracteres que podem ser utilizados pelos programa posteriormente para tocar notas e assim criar uma música.

Dicionário de mapeamento:

A implementação de um dicionário (pode ser alterado, com adição, exclusão, update) que mapeia um caractere para uma ação musical específica como, por exemplo:

Ler(Letra A) → Toca(Nota Lá);

Ler('!') → TrocaInstrumento(MIDI 24);

Ler(SPACE) → AlteraVolume(DobraVolume/VolumeMáximo).

Tocar notas: (Letras A,B,C,D,E,F,G,H)

O caractere é lido e a nota respectiva a uma das letras é tocada. Consulta o dicionário de mapeamento fornecido para realizar essa ação.

Troca de instrumento: (!, O/o, I/i, U/u, NL, “;” , pares e ímpares, “,”)

O programa analisa o símbolo e chama uma função que controla o instrumento atual (General MIDI) que está tocando e verifica se é possível realizar uma alteração para um outro.

Controle de volume e tonalidade: (Espaço, “?”, “.”)

Ao verificar esses caracteres, realiza uma verificação para dobrar o volume, caso isso não ocorra, coloca o volume no máximo. Também pode aumentar uma oitava ou voltar para a oitava default.

Exportar arquivo de resultado:

Após tocar a música correspondente a conversão do texto do usuário, também é possível salvar esta em um arquivo para uso futuro em outras aplicações. A lista de caracteres convertida e formatada também está disponível neste arquivo.

Classes previstas

Serão definidas classes para ajudar na organização e legibilidade do código. Essas serão divididas em módulos para guardarem diferentes dados e executarem múltiplas operações, além de ajudar a separar funcionalidades diferentes dentro do software.

GeradorMusica:

A classe GeradorMusica guarda informações sobre o programa de uma forma mais generalizada, coordenando ações a serem feitas, rotinas a serem chamadas e controle de erros da operação. Essa classe terá uma visão ampla do programa e será responsável pelo seu funcionamento correto, na ordem especificada e armazenando dados importantes sobre a execução.

Atributos: textoEntrada(string), textoConvertido(string), estadoLeitor(int), estadoExecucao(int).

Métodos: iniciarExecucao(), interromperExecucao(), tratarErros(), atualizaEstado().

LeitorTexto:

A classe LeitorTexto é responsável pela leitura do arquivo de texto fornecido pelo usuário no início do programa. Ela será encarregada de transformar a entrada bruta de dados em informações manipuláveis pelo programa, mantendo todos os caracteres do texto original, como letras, espaços, pontuações, etc. Desse jeito, o software poderá manipular mais eficientemente os dados fornecidos.

Atributos: textoInput(string), listaCaracteres(lista de char).

Métodos: lerTexto(), formatarTexto(), obterCaractere(), armazenaCaractere().

TransformaMusica:

A classe TransformaMusica recebe caracteres já formatados e prontos para serem convertidos para comandos musicais, realizando um mapeamento pré-definido de objetos como notas, pausas, alteração de instrumento/volume para gerar uma lista completa de quais destes serão reproduzidos em forma de sons para o usuário do programa.

Atributos: listaEntrada(lista de char), listaEventos(lista de eventos), mapaTransformacao(MapaCaracteres).

Métodos: converteCaracteres(), transformaEvento(), obterEvento().

MapaCaracteres:

A classe auxiliar MapaCaracteres ajuda a classe TransformaMusica a fazer um mapeamento coerente dos caracteres para as ações musicais. Nela, está contido o dicionário de caracteres e suas respectivas correspondências de notas, assim esse pode ser facilmente alterado no caso de novas implementações de notas ou de remoção/atualização destes.

Atributos: dicionarioMapeamento(lista caracteres/ lista eventos).

Métodos: obterAcao(), atualizarCaractere(), excluirCaractere(), adicionarCaractere(), mostraMapa().

ControleMusica:

A classe ControleMusica gerencia o estado atual dos parâmetros dos sons, como o volume, a oitava e o instrumento. Também controla a possibilidade de alteração desses valores, retornando ‘true’ em caso positivo e ‘false’ em caso negativo. Assim, a consistência da música será mantida independentemente das mudanças de características solicitadas.

Atributos: instrumentoAtual(int), volumeAtual(int), oitavaAtual(int).

Métodos: alterarInstrumento(), alterarVolume(), verificaVolume(bool), dobrarVolume(), alterarOitava(), verificaOitava(bool), setOitavaDefault().

TocadorNotas:

A classe TocadorNotas tem como ação principal tocar a nota que foi convertida do texto para o usuário poder ouvi-la. Também recebe informações de ControleMusica de modo que possa alterar as características necessárias dessa nota/som. Com isso, o usuário poderá finalmente transformar seu texto de entrada em uma música.

Atributos: playerAudio(biblioteca), estadoMusica(int).

Métodos: selecionaEvento(), selecionaSom(), exportaMusica(arquivo).

Croqui da interface

Abaixo, está um croqui que representa uma ideia da implementação da interface no software proposto no trabalho.

