



# Music Lang: A linguagem de programação para criar músicas

Aluno: Henrique Fazzio Badin

Observação Importante: Para todos os testes foi utilizado o WSL no Windows, então foi necessário fazer a instalação do clang e do VLC Media Player para execução.

# A Motivação por Trás da Criação

Para a matéria de jogos, não conseguia achar na internet músicas simples que fossem condizentes com a atmosfera do meu jogo.

Softwares de música eram complexos para iniciantes como eu.

Isso me deu vontade de criar uma linguagem de programação, que é um meio que eu tenho mais contato e mais prática na qual eu pudesse tentar gerar músicas simples.



# Características Principais da Linguagem

## Estrutura de Código

- Variáveis, condicionais (if/else), loops (repeat)
- Tipagem estática e forte para evitar erros
- Toda linha de código que não é uma declaração de variável deve estar dentro de blocos
- Todo código inicia com um setup

## Elementos Musicais

- Suporte a tipos novos criados como note e sequence
- music\_base para setup inicial de instrumentos
- Termos amigáveis: note, pause\_duration, sequence

## Execução e Compatibilidade

- Notas tocam arquivos .wav em pastas de instrumentos (Pasta guitar tem todas as notas em .wav)
- Compilada para LLVM, gerando executável nativo
- Usa VLC, PowerShell e cmd.exe para operação

# Curiosidades e Potencial



Para teste de algumas musicas foi adaptado o código já feito para arduino no repositório

<https://github.com/robsoncouto/arduino-songs/tree/master>



## Adaptação da disciplina

Foi utilizado o compilador v3.0 feito na disciplina como base para a criação do compilador musical



## Compilação em VM

O compilador gera LLVM, podendo ser rodado em diferentes sistemas e arquiteturas

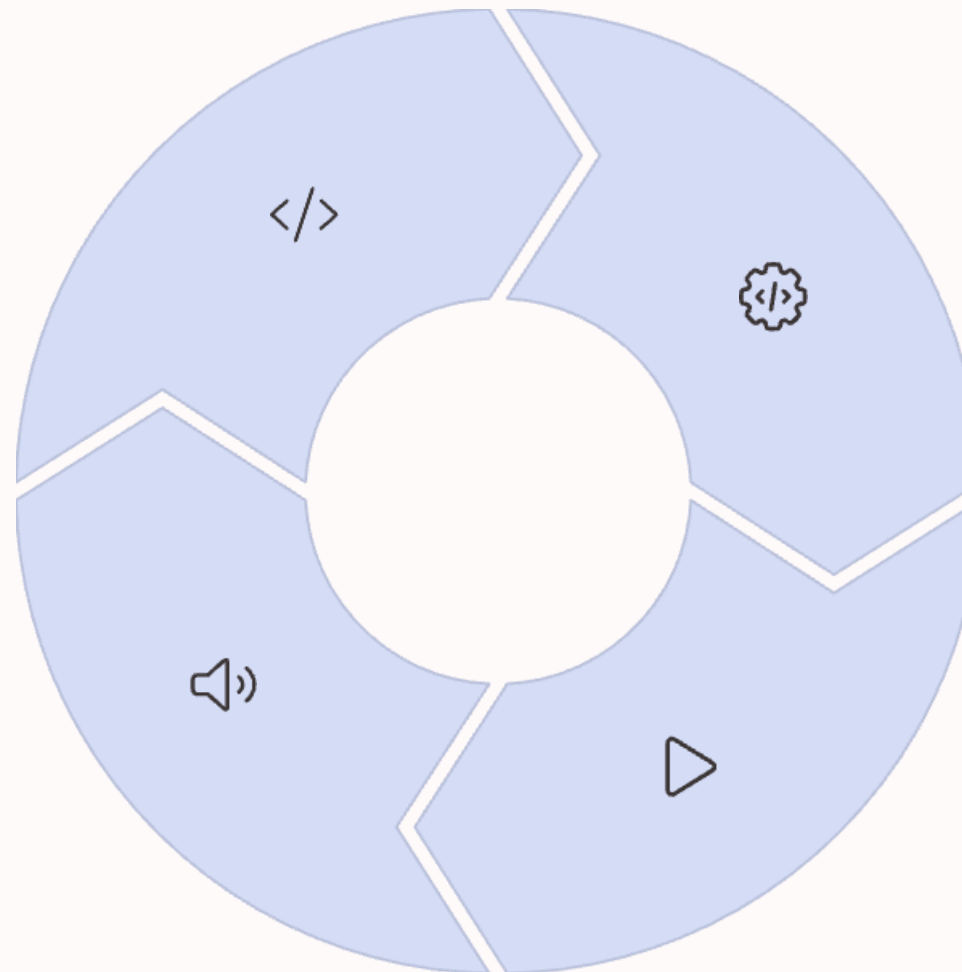
# O Ciclo da Geração Musical

## Código Musical

Escreva a codificação da sua musica em uma arquivo .mus.

## Sons Gerados

Ao rodar o executável, o computador executa uma série de comandos no Windows para tocar cada nota e cada pausa na ordem do seu código, usando arquivos .wav e o VLC Media Player



## Compilação LLVM

O compilador traduz seu código para LLVM IR (.ll)

## Criação do executável

Transforma o arquivo C de funções personalizadas em um .o

Transforma o arquivo .ll em um .o

Faz o link desses 2 arquivos e gera um executável

Observação: Para todos os testes foi utilizado o WSL no Windows, então foi necessário fazer a instalação do clang e do VLC Media Player para execução.

# Exemplo 1: Código Musical com quase todas as funcionalidades permitidas

```
music_base {  
    instrument "guitar"  
    tempo_base 120  
}  
  
var x int = 5  
var y double = 3.14  
var flag bool = true  
  
var n1 note  
var n2 note = note E4 duration 0.5  
  
var seq1 sequence = [note D4 duration 0.4 pause_duration 0.2,  
n2 pause_duration 0.6]  
var seq2 sequence = [n1 pause_duration 1, n2 pause_duration  
2]
```

```
{  
    x = x + 2  
    y = y * 2.0  
    flag = false  
    n1 = note A4 duration 1.0  
    play_note n1  
    pause_duration 1.2  
    play_sequence seq1  
  
    if x > 3 {  
        play_note n2  
        pause_duration 0.7  
    } else {  
        play_sequence seq2  
    }  
  
    repeat 3 times {  
        play_note n1  
        pause_duration 0.2  
    }  
  
    repeat 2 times while x < 10 {  
        play_sequence [note G4 duration 0.5 pause_duration 0.2, n1 pause_duration 0.6]  
        x = x + 1  
    }  
  
    y = y - 1.0  
    x = x * 2  
  
    flag = (x > 2) && (y < 10)  
  
    if flag {  
        pause_duration 2  
        play_note note D4 duration 1  
        play_note note G4 duration 1  
        pause_duration 0.5  
    }  
}
```

# Exemplo 2: Musica Tema Star Wars

Uma composição famosa, agora em código, demonstra a capacidade da linguagem de criar melodias reconhecíveis com simplicidade e clareza na prática.

```
music_base {
  instrument "guitar"
  tempo_base 120
}
var lick1 sequence = [note A3 duration 0.75 pause_duration 0.5, note E4 duration 0.75 pause_duration 0.5]
var lick2 sequence = [note D4 duration 0.5 pause_duration 0.25, note C#4 duration 0.5 pause_duration 0.25, note B3 duration
0.5 pause_duration 0.25, note A4 duration 1.0 pause_duration 0.75, note E4 duration 0.5 pause_duration 0.25]
var lick3 sequence = [note D4 duration 0.5 pause_duration 0.25, note C#4 duration 0.5 pause_duration 0.25, note D4 duration
0.5 pause_duration 0.25, note B3 duration 0.75 pause_duration 0.5]

{
  repeat 3 times {
    play_note note E3 duration 0.5
    pause_duration 0.25
  }
  play_sequence lick1
  repeat 2 times {
    play_sequence lick2
  }
  play_sequence lick3
}
```

# Link do repositório da APS e vídeo de execução

**Repositório:** <https://github.com/HenriqueFBadin/APSLogComp/tree/main>

**Vídeo de execução:** <https://youtu.be/XNod50FGCXs>