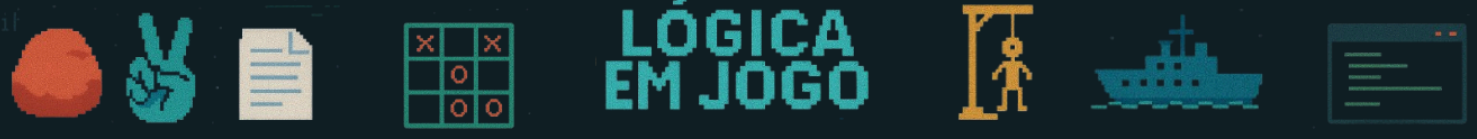

Sessão 08

Jogo da forca com IA

Professor Thiago Goveia

INSTITUTO FEDERAL
Minas Gerais

1.0 Objetivo da Sessão

Nesta sessão, daremos um salto gigantesco! Vamos conectar nosso jogo da forca em C à API do **Google Gemini**. A IA se tornará uma parte ativa do nosso jogo de duas maneiras:

1. **Versão 05: Gemini como "Mestre do Jogo"** - A IA irá gerar a palavra secreta e uma dica para o jogador humano adivinhar.
2. **Versão 06: Gemini como "Jogador"** - Nós (humanos) escolheremos uma palavra, e a IA tentará adivinhá-la, letra por letra.

2.0 Focos Conceituais Chave

1. **A "Ponte" C -> Python (popen)**: Em C, usar `popen()` é uma forma de executar um comando externo (como um script Python) e ler o que ele imprime na tela, como se estivéssemos lendo um arquivo.
2. **API (Interface de Programação de Aplicações)**: É um conjunto de regras que permite que programas diferentes conversem entre si. Pense na API como um "**garçom**" do Gemini: nosso script Python (o cliente) faz um pedido, e a API (o garçom) leva o pedido ao Gemini (a cozinha) e traz a resposta.
3. **Engenharia de Prompt**: A qualidade da resposta da IA **depende 100% da qualidade da nossa pergunta** (o "prompt"). Aprenderemos a dar instruções claras, definindo Papel, Tarefa e Restrições.
4. **Variáveis de Ambiente**: Chaves de API (senhas) **NUNCA devem ser escritas diretamente no código**. Vamos aprender a armazená-las de forma segura em variáveis de ambiente, que nosso script Python pode ler.

3.0 Preparação: O Lado Python (A "Ponte")

Antes de o C poder *chamar* a IA, precisamos construir a "ponte" que ele irá usar.

3.1. Instalar a Biblioteca do Google

No terminal, instale a biblioteca do Gemini para Python: `pip install google-generativeai` . Verifique se o Python está instalado!

3.2. Configurar a API Key

1. Obtenha sua API Key no Google AI Studio.
2. Defina-a como uma **Variável de Ambiente**. **No terminal PowerShell do VSCode:** `$env:GEMINI_API_KEY = 'sua_chave_aqui'`
   Atenção: Este mesmo terminal deverá ser utilizado para executar o código

3.3. Criar o Script `gemini_bridge.py` (Ponte para o Gemini)

Crie este arquivo na mesma pasta do seu projeto C. Este script receberá o *prompt* do nosso programa C, enviará ao Gemini e imprimirá a resposta.

```
import google.generativeai as genai
import os
import sys

# Função para registrar mensagens de log, isto é, mensagens recebidas do C e do Gemini
def registrar_no_log(mensagem):
    with open("gemini_bridge_log.txt", "a") as log_file:
        log_file.write(mensagem + "\n")

try:
    # 1. Configura a API Key a partir da Variável de Ambiente
    genai.configure(api_key=os.getenv("GEMINI_API_KEY"))

    # 2. Inicializa o modelo
    model = genai.GenerativeModel('gemini-2.5-flash')

    # 3. Pega o prompt que o C enviou (via argumento de linha de comando)
    if len(sys.argv) > 1:
        prompt_do_c = sys.argv[1]
        registrar_no_log(f"Prompt recebido do C: {prompt_do_c}")
    else:
        raise ValueError("Nenhum prompt fornecido pelo C.")

    # 4. Envia o prompt para a API e recebe a resposta
    response = model.generate_content(prompt_do_c)
    registrar_no_log(f"Resposta recebida do Gemini: {response.text}")
```

```
    # 5. Imprime a resposta para o C poder ler
    # Remove quebras de linha da resposta da IA para facilitar a leitura
    print(response.text.strip())

except Exception as e:
    # Se algo der errado (ex: API Key errada), informa o C
    print(f"ERRO;{str(e)}")
```

A lógica está lançada!

- Embora esta seja uma sessão de **aplicação**, na qual a lógica já está implementada, é fundamental que você **leia** o código, **compreenda** e tente **fazer modificações**.
- Após executar e testar, faça o commit no **Github**, descrevendo as funcionalidades e limitações de cada versão

4.0 Versão 05: Gemini como "Mestre do Jogo" (Gerando a Palavra)

Vamos modificar nosso `main.c` para que, em vez de ler de um arquivo, ele pergunte ao Gemini.

4.1. Engenharia de Prompt (em C)

O prompt é a parte mais importante. Precisamos que o Gemini nos dê a resposta em um formato que o C possa quebrar facilmente (ex: `PALAVRA;DICA`).

```
// 1. Escolher o tema
char tema[] = "Animais"; // (Experimente pedir ao usuário com scanf)

// 2. Criar o prompt
char prompt[512];
snprintf(prompt, sizeof(prompt),
    "Voce eh um mestre do jogo da forca. "
    "Sua tarefa eh gerar uma palavra e uma dica. "
    "RESTRICOES: "
    "1. O tema deve ser '%s'. "
    "2. A palavra deve ter entre 6 e 10 letras. "
    "3. A palavra deve ser UMA UNICA PALAVRA, sem espacos e sem acentos. "
    "4. A dica deve ser curta (max 5 palavras). "
    "5. Responda APENAS no formato: PALAVRA;DICA (tudo maiusculo). "
    "Exemplo: CACHORRO;Melhor amigo do homem.",
    tema
);
```

INSTITUTO FEDERAL
Minas Gerais

4.2. Esqueleto de Código C (Chamando a Ponte)

Usamos popen() para executar o script Python e fgets() para ler sua resposta.

```
// Em main.c, substitua a lógica de escolher palavra (V3/V4)

char palavra_secreta[20];
char dica[50];
//Buffer para ler a saída do Python (resposta no formato PALAVRA;DICA).
char buffer_saida_python[512];
//Buffer para montar o comando shell que será executado.
char comando_shell[768];
FILE *pipe; //Pipe em inglês é cano. Um “cano” que liga o Python ao C

// 1. Montar o comando que o C vai executar
// Lembre-se de colocar o prompt entre aspas!
// Monta o comando de forma segura usando snprintf.
// Caso gemini_bridge.py não seja encontrado, substitua-o pelo caminho
completo. Exemplo: "python \"C:\\Users\\..\\gemini_bridge.py\" \"%s\""
snprintf(comando_shell, sizeof(comando_shell), "python gemini_bridge.py
\"%s\"", prompt);

// 2. Executar o comando e abrir um "pipe" para ler a saida
pipe = popen(comando_shell, "r");
if (pipe == NULL) {
    printf("Erro ao executar a ponte Python!\n");
    return 1;
}

// 3. Ler a resposta do pipe (o que o Python imprimiu)
if (fgets(buffer_saida_python, sizeof(buffer_saida_python), pipe) != NULL) {
    // 4. Quebrar a resposta "PALAVRA;DICA" usando strtok
    char* token = strtok(buffer_saida_python, ";");
    if (token!= NULL) {
        strncpy(palavra_secreta, token, sizeof(palavra_secreta)-1);
        palavra_secreta[sizeof(palavra_secreta)-1] = '\0';
    }
    token = strtok(NULL, "\n"); // Pega o resto (a dica)
    if (token!= NULL) {
        strncpy(dica, token, sizeof(dica)-1);
        dica[sizeof(dica)-1] = '\0';
    }
} else {
    printf("Erro ao ler a saida da IA.\n");
    return 1;
}
```

```
// 5. Fechar o pipe
pclose(pipe);

printf("IA Gerou a Palavra! Dica: %s\n", dica);

// O JOGO COMECA AQUI
// O resto do seu código (inicializar display, loop while, etc.)
// funciona exatamente como antes
```

5.0 Versão 06: Gemini como "Jogador" (Adivinhando a Palavra)

Esta é a inversão de papéis. O jogador humano digita a palavra secreta, e a IA tenta adivinhar.

5.1. A Lógica do Jogo (Invertida)

Vamos criar um **novo modo de jogo**. O *loop* principal (`while`) agora irá:

1. Mostrar o estado atual para a IA (`__ST_` e letras erradas).
2. Construir um *prompt dinâmico* com esse estado.
3. Chamar a ponte Python (`popen`) para obter o palpite da IA.
4. Processar o palpite da IA e atualizar o estado.
5. Repetir.

5.2. Esqueleto de Código C (IA como Jogador)

```
// Em uma nova função ou modo de jogo
char palavra_secreta[20];
char display[20];
char letras_tentadas_ia[20] = ""; // Letras que a IA já tentou
int num_tentativas_ia = 0;
int erros_ia = 0;
int max_tentativas = 6;
int acertou = 0;
int enforcou = 0;

printf("Modo Invertido! Informe a palavra secreta: (Maiusculo, sem
acentos)\n");
scanf("%s", palavra_secreta); // (Use fgets para palavras com espaço)

// Inicializa o display
int tamanho = strlen(palavra_secreta);
for (int i = 0; i < tamanho; i++)
    display[i] = '_';
```

```
display[tamanho] = '\0';

while (!acertou && !enforcou){ //Laço do jogo
    printf("\n--- Turno da IA ---\n");
    printf("Palavra: %s\n", display);
    printf("Tentativas da IA: %s\n", letras_tentadas_ia);
    printf("Erros: %d de %d\n", erros_ia, max_tentativas);

    // 1. Engenharia de Prompt Dinâmica
    char prompt_ia[512];
    snprintf(prompt_ia, sizeof(prompt_ia),
            "Voce eh um jogador de forca. "
            "TAREFA: Adivinhe UMA letra. "
            "ESTADO ATUAL: A palavra eh '%s'. "
            "LETRAS JA TENTADAS: '%s'. "
            "RESTRICOES: "
            "1. Responda APENAS com a proxima letra que voce quer chutar."
            "2. Nao repita letras de '%s'. "
            "3. Responda apenas UMA LETRA MAIUSCULA. "
            "Qual sua proxima letra?",
            display, letras_tentadas_ia, letras_tentadas_ia);

    // 2. Montar comando shell
    char comando_shell[768];
    snprintf(comando_shell, sizeof(comando_shell), "python
../gemini_bridge.py \"%s\"", prompt_ia);

    // 3. Chamar a IA para obter o chute
    FILE *pipe = popen(comando_shell, "r");
    if (pipe == NULL){
        printf("Erro ao executar a ponte Python!\n");
        return 1;
    }

    char buffer_chute_ia[512];
    fgets(buffer_chute_ia, sizeof(buffer_chute_ia), pipe);
    pclose(pipe);

    char chute_ia = buffer_chute_ia[0]; // Pegamos só a primeira letra
    printf("IA chutou a letra: %c\n", chute_ia);

    // Adiciona o chute da IA ao histórico de letras tentadas
    letras_tentadas_ia[num_tentativas_ia] = chute_ia;
    num_tentativas_ia++;
    letras_tentadas_ia[num_tentativas_ia] = '\0';

    // 4. Processar o chute (lógica normal da forca)
}
```

6.0 Conclusão da Sessão 06

Você integrou com sucesso um programa C a uma API de IA moderna!

O que aprendemos:

- **Comunicação C -> Python:** Como usar **sprintf** para construir um comando de shell dinâmico, passando um prompt complexo como um argumento de linha de comando para o script Python.
- **Engenharia de Prompt (Geração):** Como instruir o Gemini a atuar em um papel (Mestre do Jogo ou Jogador), executar uma tarefa (gerar palavra e dica ou adivinhar palavra) e, o mais importante, aderir a restrições de formato (PALAVRA;DICA) ou Letra de chute, essenciais para o parsing em C.
- **Comunicação Python -> C:** Como usar popen no modo "r" (leitura) para executar o script e capturar sua saída.
- **Parsing em C:** Como usar **fgets** para ler a resposta da IA e **strtok** para dividir a string formatada em palavra_secreta e dica.