Escolha do Jogo:  
  
Essa escolha foi feita de forma colaborativa, com a participação de todos os integrantes do grupo, considerando fatores como viabilidade de implementação, popularidade do jogo e possibilidade de aplicação de estruturas na linguagem C.  
  
Para garantir uma decisão justa e eficiente, o grupo seguiu os seguintes passos:  
Inicialmente, cada integrante sugeriu possíveis jogos que fossem viáveis (que estivesse no playstore),que pudessem ser implementados para o trabalho. Após uma breve discussão, definimos três jogos como finalistas:

Uno

Banco Imobiliário

Paciência

Para cada jogo, fizemos uma análise inicial de viabilidade, considerando:  
1- A complexidade das regras e se elas poderiam ser implementadas com os conhecimentos já adquiridos na disciplina.  
2- A necessidade de uso de estruturas como listas, pilhas, filas, structs, e manipulação de arquivos (se necessário).

Após essa análise, o grupo organizou uma enquete interna para decidir, democraticamente, qual jogo seria escolhido. Cada integrante votou em uma das três opções.

Com todos os votos, o jogo **Uno** foi o escolhido para ser o foco do nosso projeto.  
Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.  
https://abrir.link/NCCwL

Motivos para Descartar as Outras Opções:

**Banco Imobiliário:**  
**Alta complexidade das regras:** O jogo envolve regras financeiras detalhadas, como compra e venda de propriedades, pagamento de aluguel, hipotecas, cálculos bancários e diversas exceções que aumentariam consideravelmente a complexidade do trabalho.  
**Paciência:   
Menor envolvimento dos integrantes:** Por ser um jogo menos dinâmico e mais repetitivo, o grupo demonstrou menos entusiasmo em trabalhar com ele, o que também foi levado em conta no momento da decisão.  
  
Justificativa da Escolha:

**Regras bem definidas:** Uno é um jogo com regras simples e objetivas

**Popularidade e interesse do grupo:** O jogo é amplamente conhecido e apreciado pelos integrantes do grupo, o que aumenta o engajamento e facilita o entendimento das regras durante a implementação.  
  
  
Entendimento do Projeto:  
Após a definição do jogo Uno como base para o projeto, o grupo passou a estudar com mais profundidade a estrutura e o funcionamento do jogo, com o objetivo de planejar sua implementação em linguagem C, utilizando conceitos importantes como **listas**, **pilhas**, e **filas**. Além disso, organizamos um cronograma de trabalho colaborativo, com reuniões semanais para garantir que todos os membros pudessem contribuir ativamente para o desenvolvimento do projeto.  
  
Desde o início do projeto, o grupo decidiu seguir uma abordagem colaborativa e organizada. Para isso, definimos uma rotina de reuniões online semanais, realizadas **todos os domingos**, com o objetivo de revisar o progresso  
  
Durante essas reuniões, cada integrante teve a oportunidade de sugerir melhorias, discutir dúvidas e contribuir com ideias para a implementação. Essa dinâmica foi essencial para garantir:

Participação igualitária de todos os membros, Divisão justa das responsabilidades, Troca constante de conhecimento.  
  
  
Entendimento inicial do jogo:  
Antes de iniciar a codificação, o grupo discutiu e analisou como funciona a mecânica de turnos do UNO. Foi identificado que a ordem de jogadas é sequencial, ou seja, cada jogador joga na sua vez e em seguida aguarda sua próxima vez, voltando para o final da fila. Essa lógica encaixa perfeitamente no conceito de fila.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

A primeira parte do código é a inclusão das bibliotecas stdio.h e string.h. Optamos por utilizar stdio.h porque ela oferece funções essenciais para entrada e saída de dados, como printf e fgets, que usamos para exibir mensagens e ler comandos do usuário. Já a biblioteca string.h foi escolhida porque ela fornece funções para manipulação de strings, como strcpy e strncmp, que são necessárias para copiar e comparar os nomes dos jogadores dentro do vetor.

#include <stdio.h>

#include <string.h>

Em seguida, definimos duas constantes com o #define: MAX, que representa o número máximo de jogadores permitido na partida, e TAM\_NOME, que define o tamanho máximo permitido para o nome de cada jogador. Usamos essas definições para tornar o código mais organizado e fácil de ajustar futuramente, se necessário.

#define MAX 5

#define TAM\_NOME 20

O vetor jogadores foi criado para armazenar os nomes de cada participante, simulando uma fila. Ele é bidimensional porque armazena várias strings (nomes), e cada string tem um tamanho limitado por TAM\_NOME. Também criamos a variável total\_jogadores, que é responsável por contar quantos jogadores foram adicionados à fila até o momento.

char jogadores[MAX][TAM\_NOME];

int total\_jogadores = 0;

Para adicionar jogadores, criamos a função entrarNaFila, que recebe o nome como parâmetro. Dentro dela, usamos strcpy para copiar o nome do jogador para o vetor de jogadores na posição correta. Só adicionamos o jogador se ainda houver espaço na fila, o que é verificado pela condição total\_jogadores < MAX. A escolha dessa estrutura simples permitiu que o grupo compreendesse bem o funcionamento da fila sem complicações desnecessárias.

void entrarNaFila(const char\* nome) {

if (total\_jogadores < MAX) {

strcpy(jogadores[total\_jogadores], nome);

total\_jogadores++;

}

}

A função proximoTurno representa a parte mais importante da lógica de turnos. Quando chamada, ela imprime na tela o nome do jogador que está na vez (sempre o primeiro da fila), e depois espera o usuário pressionar ENTER ou digitar "uno" para encerrar o jogo. Para isso, usamos fgets para capturar a entrada e strncmp para verificar se a palavra "uno" foi digitada. Caso sim, o jogo termina com exit(0). Após o jogador encerrar seu turno, seu nome é armazenado temporariamente e os demais jogadores são "empurrados" uma posição para frente no vetor. Por fim, o jogador que acabou de jogar é colocado no final da fila. Essa movimentação de nomes foi escolhida para simular o comportamento real de uma fila de maneira clara e direta.  
  
Aqui o jogador pode digitar tanto “Uno” ou “uno” que o jogo se encerra.  
foi um problema discutido durante o trabalho, se o jogador digitar qualquer outra coisa o jogo continua.

void proximoTurno() {

printf("\nVez de: %s\n", jogadores[0]);

printf("Pressione ENTER para encerrar o turno (ou digite 'sair' para encerrar o jogo)...\n");

char input[10];

fgets(input, sizeof(input), stdin);

if (strncmp(input, "uno", 3) == 0 || strncmp(input, "Uno", 3) == 0) {

printf("Jogo encerrado.\n");

exit(0);

}

Esse bloco de código é o que faz a fila dos jogadores girar. O jogador que joga sai da frente e vai para o final da fila, mantendo a ordem dos turnos. É como uma roda onde todos esperam sua vez e voltam para o fim da fila depois de jogar.

char temp[TAM\_NOME];

strcpy(temp, jogadores[0]);

for (int i = 0; i < total\_jogadores - 1; i++) {

strcpy(jogadores[i], jogadores[i + 1]);

}

strcpy(jogadores[total\_jogadores - 1], temp);

}

No main, adicionamos quatro jogadores para a simulação e iniciamos uma variável rodada para contar as rodadas. Em vez de limitar o número de rodadas, optamos por um while (1), que cria um laço infinito. Isso permite que o jogo continue até que o jogador decida manualmente encerrá-lo digitando "uno". Esse comportamento foi pensado para tornar a simulação mais próxima de uma partida real de Uno, onde as jogadas continuam indefinidamente até que alguém ganhe ou deseje parar.

int main() {

entrarNaFila("Você");

entrarNaFila("Jesse");

entrarNaFila("Rafaela");

entrarNaFila("Fabio");

int rodada = 1;

while (1) { // Laço infinito até o jogador digitar "Uno"

printf("\n======= RODADA %d =======\n", rodada);

proximoTurno();

rodada++;

}

return 0;

}

Todas as escolhas no desenvolvimento desse código foram feitas com o objetivo de simplificar a compreensão da estrutura de filas e mostrar de forma visual e didática como o jogador atual é reposicionado ao final após seu turno, algo que pode ser visualizado facilmente a cada rodada.