```c
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <SDL2/SDL.h>

// Declarações globais:
bool esta_rodando = false;                      // status de execução do programa

SDL_Window *janela = NULL;                      // ponteiro para uma janela
int largura = 800;                              // largura da janela
int altura = 600;                               // altura da janela

SDL_Renderer *renderizador = NULL;              // ponteiro para um renderizador

uint32_t *buffer_de_cor = NULL;                 // ponteiro para o color buffer
SDL_Texture *textura_do_buffer_de_cor = NULL;   // textura para o buffer de cor


// Protótipos de funções:
bool inicializar_janela(void);          // inicializa uma janela
bool configurar(void);                  // setup inicial da aplicação
void processar_input(void);             // recebe e processa inputs do usuário
void atualizar(void);                   // atualiza o estado do programa
void renderizar(void);                  // renderiza a aplicação
void destruir_janela(void);             // faz a limpeza de estruturas da memória


// Função main:
int main(void)
{
    esta_rodando = inicializar_janela();

    if(!configurar())
    {
        destruir_janela();
        fprintf(stderr, "Erro na configuração do ambiente.\n");
        return 1;
    }

    while (esta_rodando)
    {
        processar_input();
        atualizar();
        renderizar();
    }

    destruir_janela();

    return 0;
}



// Definição das funções:
bool inicializar_janela(void)
{
    // Inicializa o ambiente SDL. Podemos inicializar os
    // gráficos, o mouse, o teclado, etc. Iremos, no momento,
    // inicializar tudo:
    if (SDL_Init(SDL_INIT_EVERYTHING) != 0)
```

```
    {
        fprintf(stderr, "Erro na inicialização do SDL.\n");
        return false;
    }

    // Agora temos que criar uma janela SDL. A função que cria a janela
    // tem 6 parâmetros: título, x, y, w, h, flags. Se o título for null,
    // a janela não terá um título.
    janela = SDL_CreateWindow(NULL,
                              SDL_WINDOWPOS_CENTERED,
                              SDL_WINDOWPOS_CENTERED,
                              largura,
                              altura,
                              SDL_WINDOW_BORDERLESS);
    if (!janela)
    {
        fprintf(stderr, "Erro na criação da janela SDL.\n");
        return false;
    }

    // Agora temos que criar um renderer para a janela que foi criada,
    // um renderer que vai acompanhar nossa janela. Temos que passar
    // o ponteiro para a janela, o display onde a janela será exibida (-1
    // significa o display padrão), e flags (0 significa que não tem
    // nenhuma flag especial).
    renderizador = SDL_CreateRenderer(janela, -1, 0);
    if (!renderizador)
    {
        fprintf(stderr, "Erro na criação do renderizador SDL.\n");
    }

    // Se chegamos aqui, conseguimos inicializar o SDL, criamos uma
    // janela e acoplamos um renderizador para essa janela.
    return true;
}


bool configurar(void)
{
    // Cria o color buffer, como uma matriz de cores de pixels com
    // tamanho largura x altura, armazenada em um array por prioridade de linha.
    // Para acessar um determinado pixel, fazer:
    //    (largura * linha) + coluna
    buffer_de_cor = (uint32_t *) malloc(sizeof(uint32_t) * largura * altura);
    if (!buffer_de_cor)
    {
        fprintf(stderr, "Erro na alocação do buffer de cor.\n");
        return false;
    }

    // Cria uma textura SDL que é usada para mostrar o buffer de cor:
    textura_do_buffer_de_cor = SDL_CreateTexture(
        renderizador,
        SDL_PIXELFORMAT_ARGB8888,
        SDL_TEXTUREACCESS_STREAMING,
        largura,
        altura);
    if (!textura_do_buffer_de_cor)
    {
        fprintf(stderr, "Erro na criação da textura SDL.\n");
```

```
        return false;
    }

    // Se tudo foi configurado corretamente, retorna true:
    return true;
}

void processar_input(void)
{
    // Structure para receber os eventos
    SDL_Event evento;

    // Recebe o evento:
    SDL_PollEvent(&evento);

    // Testa o evento recebido:
    switch (evento.type)
    {
        case SDL_QUIT:
            esta_rodando = false;
            break;
        case SDL_KEYDOWN:
            if (evento.key.keysym.sym == SDLK_ESCAPE)
                esta_rodando = false;
            break;
    }
}

void atualizar(void)
{

}

void renderizar_buffer_de_cor(void)
{
    // Copiaremos todo o conteúdo do color buffer para a
    // textura do color buffer
    SDL_UpdateTexture(textura_do_buffer_de_cor,
                      NULL,
                      buffer_de_cor,
                      (int) (largura * sizeof(uint32_t)));

    SDL_RenderCopy(
        renderizador,
        textura_do_buffer_de_cor,
        NULL, NULL);
}

void limpar_buffer_de_cor(uint32_t cor)
{
    for (int l = 0; l < altura; l++)
    {
        for (int c = 0; c < largura; c++)
        {
            buffer_de_cor[largura * l + c] = cor;
        }
    }
}

void renderizar(void)
```

```
{
    // Ajusta a cor do renderer. Tem 5 parâmetros: o renderer, os valores
    // RGB e o valor do alpha (transparência)
    SDL_SetRenderDrawColor(renderizador, 255, 0, 0, 255);

    // Agora vamos limpar tudo:
    SDL_RenderClear(renderizador);

    // Agora vamos renderizar o color buffer:
    renderizar_buffer_de_cor();

    // Precisamos ser capazes de limpar nosso color buffer pois, a cada
    // momento que precisamos mostrar um frame da animação ou jogo, precisamos
    // antes limpar o color buffer para começar a renderizar o color buffer
    // novamente. Vamos passar a cor com a qual queremos "zerar" o color buffer.
    limpar_buffer_de_cor(0xFFFFFF00);

    // E agora vamos renderizar:
    SDL_RenderPresent(renderizador);
}

void destruir_janela (void)
{
    if (buffer_de_cor)
    {
        free(buffer_de_cor);
        buffer_de_cor = NULL;
    }
    SDL_DestroyRenderer(renderizador);
    SDL_DestroyWindow(janela);
    SDL_Quit();
}
```