INF1202 Trabalho Final de Algoritmos
Turmas C e D - 2020/1

Objetivo

Exercitar as habilidades e conceitos de programação desenvolvidos ao longo da disciplina através da implementação de uma aplicação em C, desenvolvida por um grupo de 2 alunos da mesma turma ou de turmas diferentes (C e D). A coordenação de um trabalho em equipe faz parte da habilidade de programar, então não serão aceitos trabalhos individuais.

O programa deve ser estruturado de forma modular em funções parametrizadas que realizam as entradas, processamento e geram as saídas da aplicação proposta.

Produto do trabalho e datas de entrega

O trabalho será entregue em 3 etapas:

a) Relatório de Andamento: dia 5 de Outubro (tarefa no Moodle)

Formalização da dupla que fará o trabalho e submissão do código inicial do trabalho no moodle. O código deve mostrar a área do jogo e o movimento da paleta controlada pelo usuário funcionando.

b) Entrega do código: dia 25/11 até as 10:20 horas pelo Moodle

Upload no Moodle em tarefa própria de um ÚNICO arquivo compactado cujo nome do arquivo é o nome dos alunos contendo:

1) Programa executável: o programa deve rodar em qualquer plataforma Windows ou Linux, não apenas na máquina do aluno. Verifique se executará sem exceções antes de entregar. Cuide para que o programa não precise de bibliotecas que não foram entregues.
2) Código documentado (arquivos .c ou .cpp). Inclua o nome dos autores no cabeçalho do programa. **Descreva no cabeçalho como comentário, os requisitos do enunciado que não puderam ser atendidos ou que não executam por erro, para ciência dos avaliadores.**
3) Bibliotecas adicionais às que estão disponíveis por padrão, exceto conio2 e ncurses. Você pode fazer upload de diferentes versões e ir aperfeiçoando o programa. Faça o upload assim que tiver uma versão executável, de modo a garantir a entrega e precaver se de problemas com servidor, redes, internet, etc.

c) Apresentação: dia 25/11

O programa será apresentado no dia 25/11 na aula teórica. O arquivo a ser apresentado será aquele carregado no Moodle. Nenhuma alteração será permitida. Ambos alunos devem dominar o código para explicá-lo. A ausência de um dos alunos na apresentação acarretará decréscimo da nota para aquele.

Avaliação

O programa deve atender todos os requisitos listados neste enunciado, não deve apresentar erros de compilação e executar sem erros, pontos serão reduzidos caso contrário.

A aplicação desenvolvida deverá demonstrar os seguintes conteúdos que serão avaliados:

1. (2 pontos) Habilidade em estruturar programas pela decomposição da tarefa em subtarefas, utilizando subprogramação para implementá-las.
2. (2 pontos) Documentação de programas (indentação, utilização de nomes de variáveis, abstração dos procedimentos para obter maior clareza, uso de comentários no código).
3. (2 pontos) Domínio na utilização de tipos de dados simples e estruturados (arranjos, estruturas) e passagem de parâmetros. Uso exclusivo de variáveis locais.
4. (1 ponto) Formatação e controle de entrada e saída, com construção de interfaces que orientem corretamente o usuário sem instruções ou intervenção adicional do programador.
5. (1 ponto) Utilização de arquivos binários e de texto.
6. (2 pontos) Atendimento aos requisitos do enunciado do programa: modelo de estrutura de dados, de interação e de relatórios, ordenação dos dados, opções do programa, etc..

A apresentação do trabalho prático, mesmo que rodando parcialmente, é pré-requisito para realizar a recuperação, caso o aluno não obtenha nota para aprovação.

Contextualização

O jogo a ser implementado é uma variação do jogo Breakout. O jogo possui camadas de tijolos posicionados próximos ao topo da área de jogo, uma palheta posicionada próxima ao fundo da área de jogo, e uma bola. Os requisitos do jogo são:

1. Cenário: O jogo é composto por um jogador, uma paleta controlada por ele, uma bola de movimento independente e um número de camadas de tijolos que valem pontos, dependendo do seu tamanho e fase do jogo.
2. O jogador tem “vidas” , cujo valor inicial é 3. Cada vez que a bola atinge a base da área de jogo, o jogador perde uma vida.
3. O jogador controla a palheta que se movimenta apenas horizontalmente para a esquerda e a direita, utilizando as setas do teclado auxiliar.
4. A bola tem movimento controlado pelo jogo, quicando entre o topo e a paleta que se movimenta na base da area de jogo, sempre num trajeto diagonal à paleta. A bola inicia parada, em cima da paleta e é lançada pelo primeiro movimento da paleta e tecla de espaço.
5. Ao atingir um tijolo, lateral ou o topo da area de jogo, a direção e sentido da da bola são invertidos. Quando a bola atingir a palheta, a bola retorna no sentido oposto e na

direção horizontal do lado da palheta que foi atingigo (p. ex., se a bola atingiu a esquerda da palheta, ela vai para a esquerda).

6. Quando a bola atingir algum tijolo, o tijolo é destruído e o jogador acumula os pontos respectivos daquele tijolo e fase.
7. Alguns tijolos precisam ser atingidos várias vezes para serem destruídos. Eles devem ter uma aparência diferente dos demais, e essa aparência deve mudar a cada vez que o tijolo for atingido. Eles valem a mesma pontuação ao serem destruídos. A quantidade deste tipo de tijolo deve avançar conforme for avançando de fase.
8. Em qualquer momento do jogo (gerado aleatoriamente), alguns tijolos deixam cair bônus quando destruídos. Se o jogador conseguir coletar o bônus com a palheta, ele ganha algum benefício, que pode ser: (a) aumento do tamanho da palheta (dura até ele perder uma vida ou avançar de fase); (b) pontos extras; (c) uma vida extra (até o limite máximo de 9 vidas).
9. O jogo se desenvolve em 3 fases de dificuldade crescente. Para atingir a próxima fase, o jogador tem que destruir todos os tijolos. A cada fase, o jogador recupera suas 3 vidas, aumenta a velocidade da bola, diminui o tamanho da paleta e dobra a pontuação dos tijolos da fase anterior, que terão disposição diferente.
10. O objetivo do jogo é passar pelas 3 fases, destruindo todos os tijolos e acumulando o maior número de pontos.
11. Em qualquer momento do jogo, o jogador pode decidir PAUSAR o jogo, utilizando a tecla “P”. O jogo solicita o nome do jogador e salva o estado atual do jogo (todas as variáveis) em um arquivo binário <nome_do_jogador.bin>. Após pausar o jogador pode continuar jogando ou sair. O jogador pode retornar ao mesmo momento do jogo quando abrir o aplicativo.
12. Em qualquer momento do jogo, o jogador pode decidir encerrar o jogo sem salvar, utilizando a tecla <ESC>.
13. Ao terminar um jogo (completar todas as fases ou perder todas as vidas), o jogo deve perguntar o nome do jogador e mostrar sua colocação em uma a lista de até 5 dos jogadores mais recentes e suas pontuações em ordem decrescente.
14. Os nomes e pontuações são salvos em um arquivo texto ordenado em ordem decrescente, contendo até 5 jogadores. Se houverem mais de 5 jogadores, os jogadores de menor pontuação sao excluídos do arquivo. Este arquivo será carregado no final de cada partida, para mostrar a colocação do jogador.
15. Os elementos do jogo (paleta, tijolos, bola) não serão representados em uma matriz correspondente a área de jogo, mas como estruturas ou arranjos de estruturas.

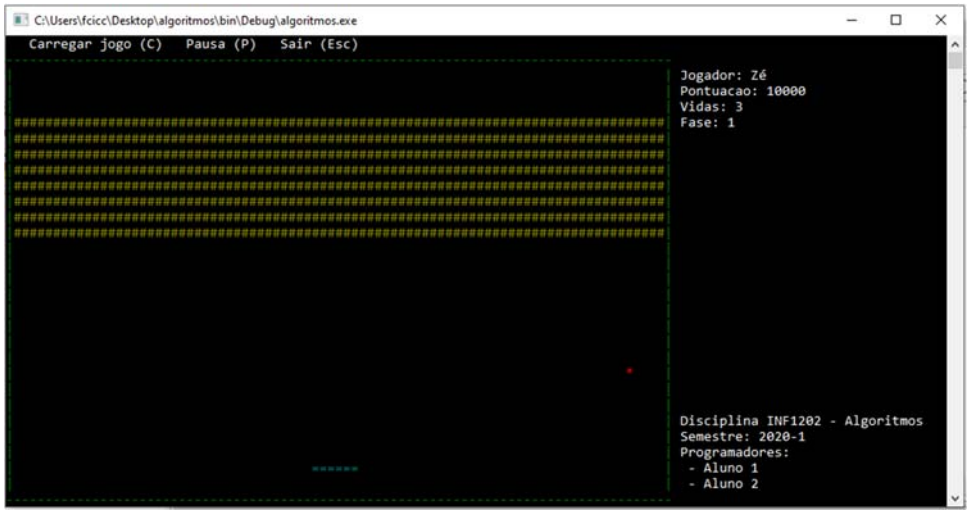
Interface do jogo

A área do jogo é assim configurada:

16. O jogo vai operar com a tela em modo texto e a interface será similar a abaixo.
17. Linha 1 da tela contem um menu de interação com o usuario com as opcoes: Carregar jogo (C), Pausa (P), Sair (Esc). A opcao de iniciar jogo não aparece no menu porque será a default de inicializacao do jogo. A segunda e penultima linha tem desenhada a moldura do jogo.
18. Aproximadamente um quarto da largura da tela será utilizado para mostrar informação e o status do jogo: Nome dos programadores, identificacao da disciplina (INF1202) ,

ano e semestres (2020-1) nome do jogador (se ele tiver jé informado) quantos pontos e vidas, em qual fase ele está, mostrar mensagem quando o jogo acabar, etc.

19. Demais configurações da área de jogo são de escolha do aluno: cores e aspectos dos tijolos, da bola e paleta, vidas, pontos, fase, etc.
20. Os requisitos não definidos nesta lista podem ser tratados como desejado pelos programadores.



Dicas

- Implementar uma função "gameloop()", que possui um laço que encerra ao pressionar a tecla ESC ou ao final da partida. Não utilizar funções como "clrscr()" dentro do laço (para evitar o efeito de tela piscando). Para apagar um objeto da area de jogo, escreva sobre ele com a cor de fundo.
- Colocar um intervalo (*sleep*) entre cada iteração para evitar que o programa utilize 100% do *core* de CPU o tempo todo.
- Implementar função de colisão para tratar as colisões da bola.
- Iniciar implementando o básico: cenário do jogo, movimentação da palheta e da bola, paredes, tijolos, colisões. Depois, ir adicionando os demais requisitos.

Funções e bibliotecas auxiliares

Funções e bibliotecas do C podem ser utilizadas para fazer a interface somente. Todas as funções de operação do jogo e manipulação de arquivos devem ser codificadas em C puro.

Uma biblioteca bastante útil é a *conio2 e windows.h* para Windows ou ncurses para o Linux. . Abaixo há alguns exemplos de funções úteis da conio2. Informações sobre as bibliotecas sao facilmente encontradas no StackOverflow ou diversos outros sites.

Exemplos e sugestões de funções auxiliares:

```
/* FUNCOES E DEFINICOES UTEIS DA CONIO2 */
/** * Colors which you can use in your application. */
typedef enum
{
```

```
        BLACK, /**< black color */
        BLUE, /**< blue color */
        GREEN, /**< green color */
        CYAN, /**< cyan color */
        RED, /**< red color */
        MAGENTA, /**< magenta color */
        BROWN, /**< brown color */
        LIGHTGRAY, /**< light gray color */
        DARKGRAY, /**< dark gray color */
        LIGHTBLUE, /**< light blue color */
        LIGHTGREEN, /**< light green color */
        LIGHTCYAN, /**< light cyan color */
        LIGHTRED, /**< light red color */
        LIGHTMAGENTA, /**< light magenta color */
        YELLOW, /**< yellow color */
        WHITE /**< white color */
} COLORS;

/* FUNCOES DE POSICIONAMENTO EM TELA E CORES */
void clrscr(); // limpa a tela
void gotoxy (int x, int y); // posiciona o cursor em (x,y)
void cputsxy (int x, int y, char* str); //imprime str na posicao x, y void putchxy (int x, int y, char ch);
//imprime char na posicao x, y void textbackground (int cor); // altera cor de fundo ao escrever na tela
void textcolor (int cor); // altera cor dos caracteres ao escrever na tela

/* FUNCOES E DEFINICOES UTEIS DA CONIO.H */
char getch(); // devolve um caractere lido do teclado, sem eco na tela int kbhit(); // devolve true se
alguma tecla foi pressionada, sem eco na tela

  /* FUNCOES E DEFINICOES UTEIS DA WINDOWS.H */
void Sleep (int t); // paralisa o programa por t milissegundos
void GetKeyState (int tecla); // pode ser usada para ler as setas do teclado

//Exemplo de uso (trecho):
if (GetKeyState (VK_RIGHT) & 0x80) // se a tecla “seta para direita” está pressionada {
        // (…)
}

/* FUNCOES E DEFINICOES UTEIS DA TIME.H */
clock();

//Exemplo de uso (trecho):

double tempo;
clock_t inicio, fim;
inicio = clock(); // salva tempo ao iniciar
// (…) // trecho do programa
fim = clock(); // salva tempo ao terminar
tempo_total_segundos = (int) ((fim – inicio) / CLOCKS_PER_SEC); /* calcula o número de segundos decorridos
durante a execução do trecho do programa*/
```