INF1202 Trabalho Final de Algoritmos
Turmas C e D - 2022/1

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. É aconselhado a criação de bibliotecas para decompor o programa, mas não obrigatório.

Produto do trabalho e datas de entrega

O trabalho será entregue em 3 etapas:

a) Relatório de Andamento: dia 15 Agosto (tarefa no Moodle)

Formalização da dupla que fará o trabalho e submissão do código inicial do trabalho no moodle até dia 15 de Agosto no Moodle. O código será mostrado na aula pratica (15 e 17 de Agosto) e deve mostrar a área do jogo e o movimento controlado pelo usuário funcionando.

b) Entrega do código: dia 05 de Outubro até as 10:00 horas pelo Moodle

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

O arquivo deve conter:

1) Programa executável: o programa deve rodar em outras máquinas, não apenas na máquina do aluno, respeitando o mesmo sistema operacional (Windows ou Linux). Verifique se executará sem exceções antes de entregar

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 as sugeridas pelo professor. Atente que, se forem usadas outras bibliotecas além das que vem no pacote do Code Block e da RayLib, estas devem ser entregues no zip do trabalho.

c) Apresentação: dia 05 de Outubro na aula teórica

O programa será apresentado na aula teórica. O arquivo a ser apresentado será aquele carregado no Moodle e rodará na máquina do professor, tanto em Linux como em Windows. Nenhuma alteração será permitida. Ambos os alunos devem explicá-lo e serem avaliados em relação ao domínio do código. A ausência de um dos alunos na apresentação acarretará decréscimo da nota para aquele.

Avaliação

O programa deve atender todos os requisitos listados nesse enunciado sem adaptações, não deve apresentar erros de compilação ou *warnings* e deve executar sem erros de execução. Pontos serão deduzidos caso contrário. Os próprios alunos vão executar e apresentar o código aos colegas.

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 (endentação, 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. Sem variáveis globais.
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 entrega e apresentação do trabalho não é requisito para aprovação na disciplina se o aluno atingir a média sem ele. Mas a entrega e a apresentação do trabalho prático, mesmo que rodando parcialmente, é pré-requisito para realizar a recuperação se o aluno ficar com média abaixo de 6.0 na dsiciplina

CONTEXTUALIZAÇÃO

O jogo a ser implementado é uma variação do jogo **Atari MILIPEDE**, cuja tela na versão original vemos na Figura 1 e iremos adaptar na nossa aplicação.

Exemplo do jogo em https://www.youtube.com/watch?v=aeYbLQAW7Qw e manual do jogo original (que não é o que iremos seguir no nosso trabalho) em https://gamefaqs.gamespot.com/atari2600/584891-millipede/faqs/41575

Figura 1 – Interface original do jogo MILIPEDE no Atari.

Neste jogo, um fazendeiro controlado pelo jogador precisa controlar duas pragas – aranhas e milípedes - que invadem sua fazenda e colher os cogumelos que aumentam sua força. O fazendeiro atira contra as pragas em movimento para matá-las e contra os cogumelos para colhê-los. Cada vez que uma aranha ou milípede toca no jogador, ele paralisa alguns (poucos) ciclos do jogo e precisa consumir cogumelos para se curar. O jogo termina quando todos os cogumelos foram colhidos ou o jogador morre por ser tocado por alguma praga sem ter mais cogumelos para se curar. Vence quem tiver guardado mais cogumelos sem consumi-los.

A seguir são numerados e listados os requisitos do jogo que são avaliados na entrega do trabalho. O número do requisito pode ser utilizado para referência caso não seja atendido.

1. CENÁRIO: O cenário do jogo será implementado em **modo texto ou gráfico.**

2. O jogo é composto por uma fazenda de cogumelos plantados aleatoriamente no terreno. As pragas sempre surgem no topo do cenário e se deslocam para baixo. O fazendeiro inicia na parte de baixo e carrega o disparador que atira e colhe. Ele se movimenta apenas na parte de baixo do cenário e acessa no máximo 25% da altura do cenário.

3. A área de jogo ocupa praticamente toda a tela sendo mantida apenas 1 linha no topo para o menu de opções
    - ESC – Sai do jogo
    - C – Carrega jogo salvo
    - P – pausa e retoma o jogo. Ao pausar, pergunta se quer salvar o jogo Sim/Não.
    - R – pausa o jogo e mostra ranking com últimos 5 jogadores.

4. Uma linha na base da área de jogo mostra andamento do jogo e seu tatus:
    - Número de cogumelos colhidos e não comidos (inicializa com 0. Esse valor é que determina o ganhador)
    - Número de cogumelos restantes (inicializa com 60)[1]
    - Número de vidas restantes (inicializa com 3)
    - Número de tiros restantes (inicializa com 200)

Elementos do jogo: Os elementos do jogo serão modelados como estruturas e vetores de estruturas. **Não** implemente a área de jogo como matrix.

5. FAZENDEIRO: estrutura de dados que contém o nome do jogador, número de tiros que podem ser atirados nas pragas ou colher cogumelos, número de cogumelos colhidos e se o fazendeiro está paralisado, em movimento ou morto.

    5.1 O Fazendeiro so se movimenta no ¼ inferior do cenário.

    5.2 Quando atingido por alguma praga o fazendeiro paralisa alguns ciclos do jogo e tem que comer cogumelos para se curar.

6. COGUMELO: estrutura de dados que contêm as coordenadas atuais do envelope do cogumelo, um campo para denotar “plantado”, “colhido” ou “comido” e um campo com a cor atual do cogumelo. Outros campos podem ser acrescentados para controlar o jogo.

    6.1 Os cogumelos são plantados aleatoriamente em toda a área do jogo, exceto a primeira linha superior e inferior, não se movimentam e desaparecem quando colhidos.

    6.2 Os cogumelos são modelagos em um arranjo LISTA DE COGUMELOS com tamanho definido pela constante NUMERO DE COGUMELOS e composto por estruturas do tipo cogumelos.

7. MILIPEDE: estrutura de dados que descreve a praga do tipo milípede.

    7.1 Deve incluir no mínimo, as coordenadas da cabeça da praga, seu tamanho (número de segmentos que devem ser plotados, a direção (esquerda ou direita) do movimento.

    7.2 A milípede pode ser desenhada com uma sequência de círculos ou usar o sprite dessa biblioteca (https://spritedatabase.net/file/9025) e mostrar o desenho com o número de segmentos

8. ARANHA: estrutura de dados que descreve a segunda praga.

    8.1 É modelada como um vetor de estruturas com duas aranhas que não precisam estar sempre no cenário.

    8.2 Deve incluir, no mínimo, as coordenadas da aranha, seu status (oculta, no cenário, morta), a direção (8 possibilidades, porém sempre preferencialmente para baixo) e a velocidade que é maior que do fazendeiro e da milípede.

    8.3 As aranhas são geradas no topo da área de jogo em posições e direções aleatórias e de modo infinito.

    8.4 Se a aranha bater em um cogumelo destrói o mesmo e muda de direção. Também muda de direção se bater

[1] Número tentativa. Talvez o número de cogumelos precise ser maior.

na parede.

8.5 Se bater no fazendeiro, ele precisa comer 2 cogumelos para não morrer e a aranha sai do cenário na mesma direção que vinha no movimento. A aranha pode ser representa por um elemento gráfico (quadrado, círculo colorido) ou por um sprite.

9. STATUS do jogo. Estrutura de dados que modela: número de cogumelos colhidos e não comidos, Número de cogumelos restantes, número de vidas restantes (inicializa com 3), número de tiros restantes (inicializa com 200)

FUNCIONAMENTO DO JOGO:

10. O jogo inicia diretamente na área de jogo, apenas com os cogumelos e a barra inferior com o status do jogo inicializado.

11. ABERTURA : Ao abrir o jogo, o usuário tem acesso a área de jogo e ao menu superior de opções com 5 opções possíveis:

11.1 Carregar Jogo: abre uma interface onde é solicitado o nome do jogador e um arquivo binário <nomedojogador>.bin é carregado posicionando os elementos do jogo onde estavam quando o jogo foi pausado.

11.2 Pausa: abre uma interface onde é solicitado o nome do jogador. Um arquivo binário <nomedojogador>.bin com todos os dados do jogo é salvo. No início do jogo, estes dados serão os mesmos de inicialização do jogo. 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 <nomedojogador.bin>. Após pausar o jogador pode continuar jogando ou sair com ESC. O jogador pode retornar ao mesmo momento do jogo quando reabrir o aplicativo, com a opção *Carregar jogo.*

11.3 ESC: dá uma mensagem e encerra o jogo. Se esta opção do menu for utilizada durante o jogo, deve salvar a pontuação do jogador no final do arquivo de ranking. Se o arquivo de ranking já tiver 5 linhas, procura o jogador com a menor pontuação insere os dados deste jogador na posição deste jogador no arquivo.

11.4 Ranking: carrega um arquivo texto onde cada linha contém <nome do jogador>.txt e sua pontuação. Ordena os dados carregados em ordem decrescente de pontuação. Abre uma interface e mostra os dados ordenados para o usuário.

11.5 Iniciar jogo. Qualquer tecla de setas do teclado auxiliar começa o jogo. O fazendeiro se movimenta na direção da seta (exceto para baixo no primeiro movimento) e o tempo de jogo começa a ser contado. A barra de espaço dispara os tiros.

12. COMPORTAMENTO DO FAZENDEIRO:

12.1 O fazendeiro sempre inicia aparecendo no meio da primeira linha do cenário.

12.2 O jogador controla o fazendeiro e pode movimentar em qualquer das 4 direções, dentro dos limites do cenário, buscando desviar das pragas e atingir tiros nos cogumelos.

12.3 O fazendeiro atira nos cogumelos para colhê-los. Se ele está ferido (houve colisão com aranhas ou milípede) ele comerá os cogumelos colhidos até se curar ou morrer.

12.4 O fazendeiro atira nas aranhas e na cabeça da milípede para exterminá-las

12.5 O fazendeiro que morre é substituído por outro no mesmo local inicial até o limite de 3 vidas.

13. COMPORTAMENTO DOS COGUMELOS

13.1 Os cogumelos são plantados aleatoriamente em todo o cenário do jogo, exceto a primeira linha da base.

13.2 Cogumelos não se movimentam.

13.3 Cogumelos desaparecem quando colhidos, comidos ou houver colisão com as pragas.

13.4 O jogo termina quando não houver mais cogumelo.

14. COMPORTAMENTO DA MILÍPEDE:

14.1 Apenas uma milípede aparece no cenário por vez.

14.2 A milípede é gerada em alguma posição e direção aleatórias do topo do cenário, com um número de segmentos também aleatório que varia entre 4 e 10.

14.3 As milípedes são geradas infinitamente enquanto durar o jogo.

14.4 A “cabeça” da milípede indica a direção do movimento e deve ser diferente dos demais segmentos

14.5 O tiro contra a milípede é verificado apenas contra a cabeça. Cada tiro acertado elimina um segmento. Quando não houver mais segmentos a milípede morre e desaparece.

14.6 A milípede se movimenta em velocidade constante para a esquerda ou direita.

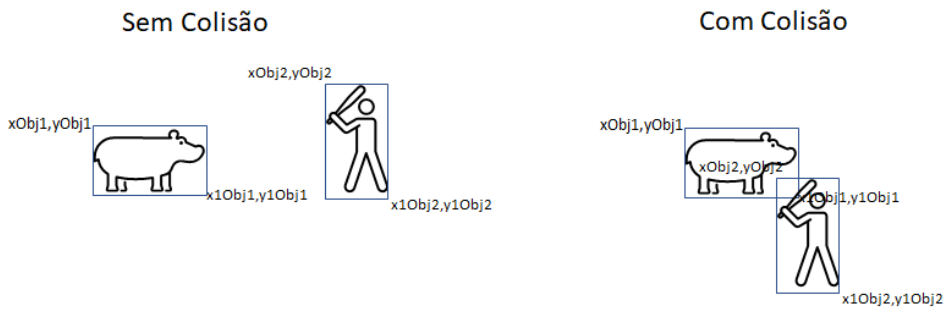
14.7 Cada vez que ela bate em um cogumelo, ele desaparece e ela ganha um segmento e desce para a linha de baixo.

14.8 Se a milípede bater na parede, desce para a linha de baixo.

14.9 Se esbarrar no fazendeiro, ele precisa comer o número de cogumelos colhidos igual ao número de segmentos da milípede para não morrer.

14.10 Após acertar no fazendeiro ou atingir a base do jogo, a milípede desaparece na base e outra é criada.

15. FUNÇÃO DE COLISAO: A colisão de objetos de jogo é tratada em computação gráfica através da noção de “envelope”. Um envelope de um elemento do jogo é o menor retângulo que envolve aquele elemento, alinhado com os eixos x e y da área de tela. Esse retângulo é representado pelas coordenadas superior esquerda e inferior direita do retângulo, como no exemplo abaixo, que mostra os envelopes do animal e do homem. No modelo mais simples de colisão, que implementaremos neste trabalho, qualquer superposição de envelopes é considerada uma colisão. Esta função que recebe as coordenadas dos envelopes e verifica se acontece colisão será desenvolvida na aula prática de funções void com parâmetros da disciplina. Outra alternativa de envelope utiliza a coordenada central do elemento e seu tamanho para calcular a colisão. O uso vai depender de qual informação é fornecida pelo sprite do personagem.



16. O jogo encerra quando não houver mais vidas ou cogumelos ou quando o jogador pressionar ESC.

17. AVANÇO DE FASE: se o jogador conseguir colher todos os cogumelos, ele troca de fase. Configure sua fase inicial de jogo para ser seja possível existir uma fase 2 ainda “jogável”. Para facilitar, parametrize suas funções e variáveis de jogo de modo que a fase 2 seja implementada com as mesmas funções com parâmetros ajustados. A fase 2 se caracteriza por, no mínimo:
    17.1 Tem mais cogumelos a serem colhidos;
    17.2 Ter mais aranhas no cenário (preveja o vetor de aranhas com tamanho 4 para prever a segunda fase.
    17.3 As milípedes variam entre 6 e 14 segmentos.

Os requisitos não definidos nesta lista e aspectos de visualização dos elementos do jogo podem ser tratados como desejado pelos programadores.

Dicas

- Implementar uma função "gameloop()", que possui um laço que encerra ao pressionar a tecla ESC ou quando número de (vidas == 0 || cogumelos == 0) .
- Não utilizar funções como "clrscr()" dentro do laço (para evitar o efeito de tela piscando). Cada elemento do jogo deve ser apagado individualmente dentro do loop.

- Caso o movimento fique muito rápido, colocar um intervalo (*sleep*) entre cada iteração para evitar que o programa utilize 100% do *core* de CPU o tempo todo.
- Iniciar implementando o básico: cenário do jogo, movimentação do fazendeiro e colisão do tiro com cogumelos. Depois, ir adicionando os demais requisitos.

/* Programa principal MILIPEDE

```
/* Main */
/* Essa estrutura de programa é uma sugestão que pode ser alterada e não um requisito do jogo */
{
    Inicializa cenário do jogo
    Laço de jogo
    Contabiliza pontuação
    Carrega score
    Ordena e mostra score do jogo
    Encerra
}

/* Fluxo geral do jogo */

Inicializa estrutura de dados do jogo
Desenha cenário
Desenha fazendeiro no centro da area inferior
Gera milípede e aranhas
Laço do jogo : repete
    Le teclas
    Move fazendeiro
    Move Milípede
    Move aranha
    Testa se houve tiro
    Testa colisão do tiro com vetor de cogumelos
    Testa colisão do tiro com milipede
    Testa colisão do tiro com aranha
    Refresh do cenário
    Se PAUSA
        Salva jogo no arquivo binario
        Le entrada do jogador  (segue o jogo) ou ESC (sai do jogo)
Até ESC ou VIDAS == 0 ou COGUMELOS restantes == 0
Da mensagem de Morreu, Ganhou ou Saiu
Pede o nome do jogador e verifica número de cogumelos colhidos pelo jogador (escore)
Carrega arquivo scores
Inclui jogador
Ordena e mostra os escores
Salva arquivo de escores
Sai do jogo

/*    Tipos definidos pelo programador   */
/* Essa estrutura de dados é uma sugestão que pode ser alterada e não um requisito do jogo, mas os
elementos e suas posições no jogo devem ser tratados como estruturas*/


Tipo COORDENADA
X int
Y int
-----
Tipo  FAZENDEIRO
    COORDENADA posição
```

string nome do jogador
int vidas
Int direção (pode ser cima, baixo , esq , dir)
int numero de cogumelos colhidos
int numero de tiros restantes
int doente (numero de cogumelos para melhorar)
int status Paralisado Livre ou Morto

Tipo MILIPEDE
COORDENADA posição
Int tamanho (numero de segmentos)
Int direcao (pode ser esq ou dir)

Tipo ARANHA
COORDENADA posição
Int direcao (8 direcoes diferentes)
Status : no cenário, oculta

COGUMELO lista _cogumelos[NUM_COGUMELOS]