Regras

Inicialização

- Seu personagem será o Pac man representado por um P amarelo
- Os fantasmas serão representados pela letra F e serão verdes
- Seu nome de jogador terá 4 letras e será registrado no inicio de cada jogo
- o Pacman não pode atravessar # pois são paredes

pontos

- Cada ponto . adicionara 5 pontos a sua pontuação
- Cada ponto “o” dará 10 ponto
- ao pegar o S o Pacman ganha um tempo de imunidade e capas de matar fantasmas
- ao pegar a fruta “S” de imunidade o pacman se tornara azul indicando sua capacidade de matar fantasmas
- sua pontuação será mostrada no canto inferior esquerdo do mapa
- ao finalizar a partida caso sua pontuação esteja entre as 5 maiores, ela sera salva no txt rank
- fantasmas não podem consumir pontos nem frutas, logo apenas o jogador pode consumidas
- uma vez coletado pontos e frutas serão eliminados do mapa em sua casa da matriz correspondente

Movimentação

- o Jogador pode se mover com AWSD ou com as setas do teclado
- tanto o pacman pode não se mover durante frames
- Os fantasmas sempre estão se movendo, a não se quando seu movimento for para uma posição invalida
- o Pacman pode s mover apenas uma casa por frame
- Os fantasmas só se movem 1 casa pro frame assim como o pacman
- O Pacman não pode se mover em diagonal
- Os fantasmas não podem se mover na diagonal

Fim do jogo

- ao finalizar o jogo por game over o Pacman ira se transformar em um M pois estara morto
- o jogo acaba quando não houver mais pontos para se coletar no mapa
- o jogo acaba dando game over se o pac estiver na mesma casa que um fantasma na matriz
- ao finalizar o jogo caso sua pontuação esteja entre as 5 maiores ela sera salva junto ao seu Nome de jogador no rank txt

Pseudocódigo

Struct do pacman, armazena a posição, seu formato e seu status

typedef struct

{

    posição em x;

    posição em y;

    Formato do pac ;

    status;

} PACMAN;

Struct fantasma, armazena sua posição e status

typedef struct

{

   posição em x;

    posição em y;

    status;

} FANT;

Struct do jogador, armazena sua pontuação, nome de 4 caracteres e seu mapa apos o jogo

typedef struct

{

    pontuação do jogador;

    nome[4 letras];

```
    mapa do jogador
} PLAYER jogador;



Função Movimentação do Pacman
void move(PACMAN *p,char **tabuleiro)
{
    ponteiro de x é igual ao x
   ponteiro sw y é igual ao y
    if (_kbhit())
    {
        //V qual tecla est sendo tocada
        char move = _getch();


        depende do(move)
        {
        caso 'w':
        caso 72: (seta baixo)
          desce um y de altura
            break;
        caso 's':
        caso  80:(seta cima)
            sobre um y
            break;
        caso 'a':
        caso 75: (seta esquerda)
          um x para a esquerda
```

```
   break;

caso 'd':

caso 77: (seta diteira)
   um X para a

   break;

}

// verifica se não está bloqueado

se(px<COLUNAS && py<LINHAS && px>=0 && py>=0)

{

   se posição nao for ‘'#')

   {
              px sera x
            py sera y

   }

}

se não

{

   //teleporta pro outro lado


   se(pac estiver na extremidade igual a zero do mapa em x)



   {

      pac ira para o numero de colunas

   }

   se(altura de pac menor que zero em y)

   {

   Pac ira para linhas-1 ( outra extremidade da linha)

   }
```

```
            se(pac estiver na extremidade positiva do mapa em x)
            {
                Pac ira para X 0
            }
            se(pac estiver na altura limite do mapa em y)
            {
               pac ira para altura y 0
            }
        }
    }
}
```

movimentação do fantasma

```
void moveFantasma(FANT *f, char **tabuleiro)
{
    se(f->status>0)
    {
        return ;
    }
    int py = f->y;
    int px = f->x;
    int movef;
    int c=1;
    enquanto(c)
    {
        movef = um numero aleatorio entre 0 e 4
```

```
    //0 subir, 1 direita, 2 descer, 3 esquerda
  depende do(movef)
    {
    caso 0:
       if(caso não seja uma parede)
       {
         fantasma desce um em y
           c=0;
       }
       break;
    caso 1
       if(caso não seja uma parede)

       {
         fantasma vai um x para a direita
           c=0;
       }
       break;
    caso 2:
se(caso não seja uma parede)

       {
       fantasma sobre um y
                  c=0;
       }
       break;
```

```
caso 3:
     se(caso não seja uma parede)
   {
   fantasma vai um x para a esquerda
       c=0;
   }
   break;
}
se(px<0)

   se(fantasma estiver na extremidade igual a zero do mapa em x)

   {
     fantasma ira para o numero de colunas
   }
   se(altura de pac menor que zero em y)
   {
   fantasma ira para linhas-1 ( outra extremidade da linha)
   }
   se(fantasma estiver na extremidade positiva do mapa em x)
   {
      fantasma ira para X 0
   }
   se(fantasma estiver na altura limite do mapa em y)
   {
     fantasma ira para altura y 0
```

```
        }
```

escreve a matriz

```
void printaMatriz(char **matriz, PACMAN pac, FANT *fant, int pont)
{
    int x,y;
    char printa;
    int desc;

    cls( GetStdHandle( STD_OUTPUT_HANDLE ));
    //espera(VELOCIDADE);
    para(y=0; y<LINHAS; y mais 1)
    {
        para(x=0; x<COLUNAS; x mais 1)
        {
            printa = matriz[y][x];
            se(pac.x==x && pac.y==y)
            {
                printa = pac.carac;
                cor(amarelo);

                se(pac.status<=0)
                {
                    cor(amarelo);
                }
                se não
```

```
            {
                cor(azul);
            }
            escreve("%c", printa);
        }
        se não
        {
            para(int i=0; i<qtdIni; i mais um
            {
                se(fant[i].x == x && fant[i].y==y&& fant[i].status<=0)
                {
                    printa = CHAR_FANT;
                    cor(verde);
                    break;
                }
                se não
                {
                    cor(cinza);
                }

            }
            escreve("%c",printa);
        }
    }
    escreve("\n");
}
```

```
escreve


}


 le o mapa
void carrega_mapa(char **matriz)
{
    ARQUIVO *arq;
    int lin, col;
    numeros i,j;
    char linha[COLUNAS];
    lin =0;
    col=0;


    //abrindo arquivo
    arq=abre arquivo("mapa2.txt","r");
    se(nao for !arq)
    {
        escreva("\nErro de entrada na abertura do arquivo!");
        saia(1);
    }


    /*escreva("linha: %d", lin);
    escreva("COl: %d", col);*/
    lin  =0;
    col=0;
```

```
/*se(lin >LINHAS && col>COLUNAS)
{
    escreva("\n##ERRO: Arquivo de entrada incompativel! \n\n");
    exit(1);
}*/
repita(i=0; i<LINHAS; i mais 1)
{
    pegue(linha,COLUNAS+1,arq);
    linha[strcspn(linha, "\n")] = 0;
    repita(j=0; j<COLUNAS; j mais 1)
    {
        matriz[i][j]=linha[j];
        se(matriz[i][j] =='.')
        {
            se(aleatorio()%50 == 0)
            {
                matriz[i][j]='S';
            }
        }
        se(matriz[i][j] =='.')
        {
            se(aleatorio()%50 == 0)
            {
                matriz[i][j]='O';
            }
        }
```

```
        }

    }



    fecha arquivo(arq);
```

```
função rank
void rank(int pont)
{
    escreve arquivo("\n%s\n",nome arquivo);
    FILE* file = abre arquivo(nome arquivo, "r");
    se(file == NAO EXISTE) {
        printf("Erro %s\n", nome arquivo);
        retorna;
    }

    int rank[MAX_SCORES+1];
    int numScores = 0;
    int score;
    enquanto(escaneia arquivo (file, "%d", &score) != FIM DO ARQUIVO) {
        rank[numScores] = score;
        numScores mais 1
    }
    fecha arquivo(file);
    numScores++;
    rank [numScores]= pont;

    // organiza o rank
    numeros i, j;
    for (i = 0; i < numScores - 1; i mais 1) {
        for (j = 0; j < numScores - i - 1; j mais 1) {
            se (rank[j] < rank[j + 1]) {
                int temp = rank[j];
                rank[j] = rank[j + 1];
                rank[j + 1] = temp;
            }
        }
    }

    // novos top 5
    file = abre arquivo(nome arquivo, "w");
    se (file == NAO EXISTE) {
```

```
        escreve("Error opening file %s\n", nome arquivo);
        retorna;
    }

    para(i = 0; i < MAX_SCORES && i < numScores; i mais 1) {
        escreve arquivo(file, "%d\n", rank[i]);
    }
    fecha arquivo(file);
}
```