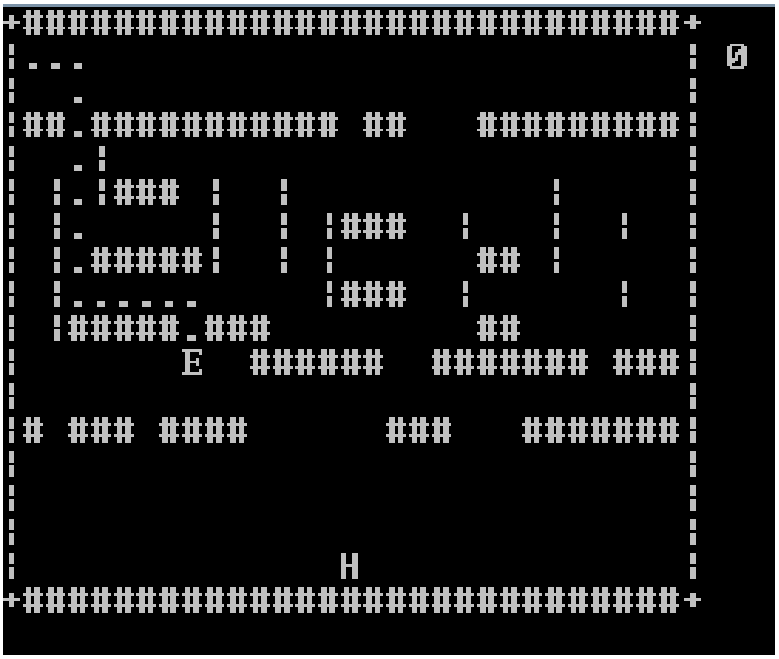
Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

1º ano, 1º semestre, 2018/2019

RELATÓRIO PRÁTICO

Programação de Computadores



Docente:

Luís Magalhães

Grupo:

Eduardo Pereira nº88279

Pedro Ranito nº88293

Resumo

Neste trabalho é efetuada a implementação do jogo de labirinto 2D “Pacman”, com algumas mudanças na versão original.

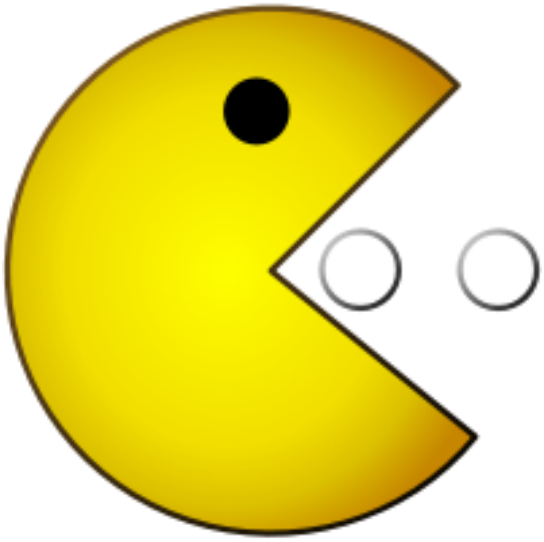
Este relatório está dividido em quatro partes:

Uma breve introdução, onde iremos explicar os nossos objetivos para este trabalho e o tipo de código que iremos utilizar.

Uma descrição do problema, onde falaremos das regras do jogo, descrevendo as características do jogo, as jogadas possíveis, e explicaremos as condições para ganhar o jogo.

A representação do estado do jogo, ou seja, tudo aquilo que foi utilizado permitindo que o jogo se torne minimamente jogável e também onde será demonstrado como são feitas.

Por fim as conclusões e perspetivas de desenvolvimento, que contêm as conclusões por nós tiradas da realização deste projeto e o que é que poderíamos ter feito para melhora-lo, caso tivéssemos mais tempo.



Índice

1.Introdução

O objetivo deste trabalho é criar em linguagem C/C++ um jogo de labirinto, tendo o nosso rupo optado por implementar uma versão do jogo “Pacman”, por se tratar de um dos maiores clássicos do mundo dos jogos.

Este trabalho revelou-se uma excelente oportunidade de aprendizagem no contexto de programação, motivando-nos o facto de poder utilizar mais tarde de forma lúdica.

Neste projeto será apresentado o jogo com uma única opção de Humano vs Computador. Para tal, foram usados ciclos, condições, matrizes e funções base das bibliotecas de linguagem c.

2.Descrição do Problema

O objetivo deste trabalho é a implementação do jogo labirinto “Pacman”. Este é um jogo eletrônico criado pelo Tōru Iwatani para a empresa Namco, sendo distribuído para o mercado americano pela Midway Games. Produzido originalmente para Arcade no início dos anos 1980, tornou-se um dos jogos mais jogados e populares no momento, tendo versões para diversas consolas, inclusive na atualidade. A mecânica do jogo é simples: a personagem do jogador tem de apanhar todos os pontos existentes do mapa, evitando ser apanhado pelos inimigos. Ao fim de apanhar todos os pontos o jogador completa o jogo.

3.Arquitetura do Sistema

O nosso programa é composto por várias funções, existindo uma para cada situação.

Sendo elas:

- **main:**
- **preenche:** gerar os campos dos labirintos e criar uma personagem;
- **carrega:** carregar o labirinto a partir do ficheiro criado no bloco de notas;
- **imprimeMapa:** imprimir o mapa, a personagem, os pontos ganhos, as regras do jogo, as instruções e os inimigos;
- **controla_personagem:** movimentar a personagem;
- **controlo_inimigos:** movimentar os inimigos de forma a dificultar o jogador;
- **main:** definir as posições dos jogadores;
- **load e gravar jogo:** carregar e gravar jogo;

4.Implementação do jogo

4.1.Representação do Estado do Jogo

O estado do labirinto será representado como uma aplicação em modo de texto, criando o aspeto de um labirinto, constituído por uma personagem (jogador) e pontos por si conquistados, inimigos e as suas localizações no mapa. O labirinto é constituído por 47 colunas e 17 linhas.

A iniciação do estado do jogo é feita através da **preenche** chama a função **main**. Através do while na função preenche, o sistema vai gerar os campos do labirinto e posicionar a personagem no mapa.

Excerto do código utilizado:

```
void preenche (char* file, char grelha [maxX][maxY], int posicao[2]){ // gerar os campos do labirinto
```

```
int linha=0;
int coluna=0;
int iterador=0;
while(file[iterador]!='\0'){
if (file[iterador]!='\n'){
grelha[linha][coluna]=file[iterador];
coluna++;
}
else{
linha++;
coluna=0;}
iterador++;
}}
```

4.2.Inicialização do Estado do Jogo

Através da função **carrega,** o labirinto vai ser carregado a partir do ficheiro criado, por nós chamado “Labirinto”. Caso não encontre nenhum ficheiro, através do printf vai aparecer uma mensagem a constar: "Erro a carregar labirinto". Caso encontre o ficheiro, a função **preenche** é chamada para gerar o labirinto e posicionar a personagem no labirinto.

Excerto do código utilizado:

```
void carrega (char *ficheiro, char grelha [maxX][maxY], int posicao[2]){  //abrir o labirinto previamente gerado pelo bloco de notas
        char labirinto[1000];
        FILE *fp = fopen(ficheiro,"r");
        if(fp==NULL)
        {printf("Erro a carregar labirinto");
return}
char c;
```

```
int i=0;

while(fscanf(fp,"%c",&c)!=EOF){

labirinto[i]=c;

i++;}

labirinto[i]='\0';

preenche(labirinto, grelha, posicao); //atribuir o labirinto gerado no bloco de notas ao preenchimento do labirinto

fclose(fp);
```

4.3.Visualização do Estado do Jogo

Na função **main** temos definido as posições iniciais dos monstros e de seguida imprimimos o mapa já com tudo preenchido .Através da função **imprimeMapa**, é definido o número de linhas e de colunas que o labirinto vai ter. É nesta função que temos as regras e as instruções do jogo, assim como o inserir da nossa personagem e dos monstros que a seguem. Também temos sempre indicada a posição atual do jogador. Através de printfs definimos as regras, os monstros e os pontos. As regras do Jogo consistem em evitar ser apanhado pelos monstros e apanhar todos os pontos. Os monstros são "&--B--$" e o jogador é " @".

A posição do jogador é visualizada no ecrã e as teclas "A-esquerda; W-cima; S-baixo; D-direita;G-gravar; C-carregar jogo" são expressas nas instruções de jogo.

A função **controlo_inimigos** que controla os monstros. Os monstros verificam a posição atual do jogador e fazem um conjunto de movimentos que seguem ao seu encontro tendo como objetivo alcançar a mesma posição, acabando assim o jogo.

Excerto do código utilizado:

```
void imprimeMapa(int pontos, char grelha [maxX][maxY], int posicao_atual_m1[2],int

posicao_atual_m2[2],int posicao_atual_m3[2],int posicao_atual_m4[2], int posicao[2]){

int i,j,dim;

system("cls");

printf("Regras do Jogo: Evitar ser apanhado pelos monstros e apanhar todos os pontos\n");
```

```
printf("ATENCAO!---> Monstros: &--B--$--?d\n");

 printf("Pontos: %d\n",pontos);

for (i=0;i<17;i++){

for (j=0;j<47;j++)

if ((posicao[0]==i) && (posicao[1]==j))

printf("@");

else if((posicao_atual_m1[0]==i) && (posicao_atual_m1[1]==j)){

printf("&");}

 else if((posicao_atual_m2[0]==i) && (posicao_atual_m2[1]==j)){

printf("B");}

else if((posicao_atual_m3[0]==i) && (posicao_atual_m3[1]==j)){

printf("$");}

else if((posicao_atual_m4[0]==i) && (posicao_atual_m4[1]==j)){

printf("?");}

else{

printf("%c",grelha[i][j]); }

if (j==47) printf("\n");}

printf("Posicao do Jogador: (%d,%d)\n", posicao[0], posicao[1]);

printf("\n Instrucoes do jogo: \n W: Movimento para cima\n A: Movimento para a esquerda\n S:

Movimento para baixo\n D: Movimento para a direita \n G: Gravar o jogo \n C: Carregar o jogo\n");}
```

4.4.Mudança de estado/movimento do jogador

A função **controla_personagem** que controla a nossa personagem, caso a tecla premida seja a,s,d e w.

A função testa verifica se tem obstáculo ou se tem pontos à frente e caso seja obstáculo, nada acontece, caso seja ponto, soma um ao contador.

A posição inicial do labirinto é 1,1 e ao andar para baixo a personagem vai andar mais uma casa

Ao andar para a esquerda a personagem vai andar uma para casa para a esquerda daí ser -1

Ao andar para a direita a personagem vai andar uma casa para a direita, subindo na coordenada y daí ser +1

4.5.Leitura e Gravação de Jogo

Na função **gravar jogo**, partimos da situação atual do jogo e criamos um novo ficheiro que guarda a nossa posição, assim como as posições dos 4 monstros por nós criados. Usamos o sprintf para ler o inteiro numa string, ou seja, neste caso, para converter um inteiro numa string e no fim utilizamos o strcat para unir as variáveis inteiras com a string. A função **load** serve para carregar os dados anteriormente guardados de modo a continuar o jogo na posição que ficou registada. Isto implica guardar as posições da nossa personagem e dos monstros, assim como os pontos que estavam já alcançados e as posições que iam ficar no ficheiro de jogo guardado, testamos para a posição 17 a 26.

4.6.Final da Execução

O jogo tem duas possibilidades: caso os inimigos alcancem o jogador, este perde o jogo; ou caso o jogador atinja o número máximo de pontos, ganha o jogo. Se a posição dos monstros for igual a do jogador aparece um printf a dizer game over e damos break ao ciclo, se apanharmos 483 pontos aparece um printf a dizer que completamos o jogo e damos um break ao ciclo.

5.Conclusões e Perspetivas de Desenvolvimento

Procedendo à análise do trabalho desenvolvido concluímos que quando existe motivação na realização do trabalho, este leva-nos a assumir maior responsabilidade e torna-se mais simples a sua conclusão.

Apesar de trabalhoso foi um projeto que gostámos de realizar, permitindo-nos aprender bastante mais acerca da linguagem C e de que modo a podemos utilizar.

Para além disto, o trabalho levou-nos a adquirir competências na gestão de tempo e organização de trabalhos, tendo sido estas as nossas maiores dificuldades durante este processo.

Contudo, fazemos um balanço positivo do resultado final, cumprindo o objetivo principal, apresentando um jogo realista e que obedeça a todas as suas regras, utilizando todos os conhecimentos que temos.

Bibliografia

- DAMAS, Luís, *Linguagem C*, Lisboa: FCA – Editora de Informática, 1999;
- Site stack overflow;