INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

**INSTITUTO POLITÉCNICO DE COIMBRA**

**Licenciatura em Engenharia Informática 2º Ano – 2º Semestre 2021/2022**

**Pipe Dream Game**

**Rafael Couto** **Nº 2019142454**

**COIMBRA**

20 de junho de 2022

Índice

[Introdução 3](#_Toc103509664)

[Implementação 4](#_Toc103509665)

[Estruturas de Dados 4](#_Toc103509666)

[Servidor 4](#_Toc103509667)

[Monitor 5](#_Toc103509668)

[Cliente 5](#_Toc103509669)

[Utils.c 5](#_Toc103509670)

[Utils.h 5](#_Toc103509671)

[Funcionalidades Implementadas 6](#_Toc103509672)

[Anexos 7](#_Toc103509673)

# Introdução

A elaboração deste trabalho prático consiste na implementação do jogo “*Pipe Dream*” em versão multiutilizador. Todos os intervenientes serão processos a correr e os programas existentes terão recurso a interfaces gráfica ou consola.

O trabalho será desenvolvido em linguagem C para Windows API (Win32) que visa o desenvolvimento de aplicações 32-bit.

O jogo “*Pipe Dream*” baseia-se em conduzir um caudal de água de um ponto de origem para um ponto de destino numa determinada área que será o denominado “*board*” ou tabuleiro neste trabalho prático.

Será seccionado em três projetos essenciais sendo eles, servidor, responsável pelo controlo de jogo, monitor, responsável pela monitorização do jogo, e cliente, responsável pela jogabilidade.

Deste modo, pretende-se desenvolver conhecimentos em relação a *named pipes* e *threads*, dando continuação a matérias anteriormente lecionadas na unidade curricular de Sistemas Operativos I. São também implementados métodos com recorrência a memória partilhada e ao *Windows registry*, que se trata de uma base de dados que gere recursos e guarda configurações para as aplicações do sistema operativo Windows.

Relativamente ao *registry*, este é usado no trabalho para guardar valores de variáveis de jogo de modo que seja possível reutilizar valores de argumentos sem necessidade de estar recorrentemente a introduzi-los cada vez que é iniciado um jogo.

Text

Description automatically generated

# Implementação

## Estruturas de Dados

De modo que existisse bom funcionamento do sistema, foram criadas três estruturas de dados referentes à comunicação, ao jogo e ao controlo de dados.

Para que fosse possível comunicar entre o monitor e o servidor, recorreu-se a uma estrutura para o “buffer circular” responsável apenas pela passagem do comando de um lado a outro. Esta estrutura está incluída dentro da estrutura de jogo que carrega todas as informações necessárias para a inicialização do tabuleiro e carrega os valores dos argumentos necessários ao início do jogo.

Num cenário de encapsulamento foi então criado uma última estrutura que controlo de dados que engloba a estrutura de dados de jogo e “*handles*” relativos a ficheiros de leitura, escrita, “mutexes” e semáforos, responsáveis pela transferência de informação.

## Servidor

No ficheiro *servidor.c* foi implementado os semáforos relativos ao controlo de instâncias dos programas servidor, monitor e cliente. São verificados os valores presentes para os argumentos de início de jogo e a leitura e ou registo, ou não, do *registry*. Também é no *servidor.c* que são inicializados o tabuleiro e a função responsável pela gestão de comandos. É também inicializada a *thread* responsável pela leitura dos comandos vindos do cliente com recurso a *named pipes*.

## Monitor

No ficheiro *monitor.c* inicializamos os semáforos necessários à comunicação com o servidor e é mapeado o bloco de memória para o espaço de endereçamento do processo criado. Posto isto, lançamos a *thread* responsável pela gestão dos comandos e aguardamos em ciclo infinito até que haja ordem de saída. Está em escuta permanente.

## Cliente

No ficheiro *cliente.c* foi implementado um teste de interface gráfica como apresentado em aula, porém sem qualquer tipo de funcionalidade associada.

## Utils.c

No ficheiro *utils.c* estão presentes todas as funções necessárias ao bom funcionamento do sistema tais como o inicializador do tabuleiro, função de comandos, função de impressão de tabuleiro, função que envia os comandos para o servidor e função responsável pela leitura de valores do *registry*.

## Utils.h

No ficheiro *utils.h* está apenas declarado as estruturas do sistema, definidas as macros do sistema e respetivos *includes*.

# Funcionalidades Implementadas

Relativamente às funcionalidades do sistema atualmente implementadas, estão presente a possibilidade de comunicação entre o programa monitor e o programa servidor, comandos básicos, com funções básicas de um programa, como, comando de saída, impressão de tabuleiro, limpeza de ecrã e comando de ajuda.

São chamados através da introdução dos comandos, “*print*”, “*pause*”, “resume”, “wall”, “*clear*” e “*help*”, “*exit*”, respetivamente.

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidades** | **Implementada** | **Parcialmente** | **Não**  **Implementada** |
| Mecanismo(s) de comunicação e sincronização com o programa monitor. | X |  |  |
| Utilizar a informação que se encontra definida no Registry. | X |  |  |
| Mapa com uma solução para o problema (ligação entre o ponto de origem e destino) | X |  |  |
| Implementação do movimento da água nos tubos. |  | X |  |
| Visualizar o(s) mapa(s) de jogo em tempo real no monitor |  |  | X  (access violation reading) |
| Interage com o utilizador e permite a receção e envio de informação de/para o servidor. | X |  |  |

# Anexos

* servidor.c
* monitor.c
* cliente.c
* utils.c
* utils.h
* SO2 – 2022 – TP - Relatório.pdf

# Conclusão

A título de conclusão, a realização deste trabalho permitiu aprofundar conhecimentos sobre sistemas operativos em geral, em particular Windows. Conhecimentos estes focados em aplicações Win32 em C

Tal como qualquer trabalho de programação permite através da consistência de trabalho evoluir nos mais diversos aspetos, especialmente no que toca a organização e estruturação de código como também no planeamento de soluções em ajuda à resolução de problemas apresentados.

Foi possível aplicar os conhecimentos obtidos em aula, tais como, aplicações de recurso a Unicode, utilidades de *Registry*, *Multithreading*, mecanismos de sincronização e mecanismos de comunicação *Named Pipes*.