Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Viana do Castelo

Algoritmos e Estruturas de Dados

Espaço Urbano de Edifícios

Engenharia Informática

Jackson Barreto Costa Júnior, 24031

Wallefer Pantoja Sousa, 24544

2019-2020

Doutor Salvador Lima

08 de janeiro de 2020

Sumário

1 - Enquadramento do Trabalho

> O *espaço urbano de edifícios* é o espaço ocupado por edifícios com finalidades diversas (habitação, comércio, serviços, etc.) que configuram as inúmeras cidades, vilas e aldeias portuguesas e noutros países.

Citação 1 - Enunciado do Trabalho Prático - Parte 1

Amparados pelo enunciado (Parte I), optou-se por contextualizar o presente trabalho em um cenário internacional, mas teve-se a atenção de se escolher uma conjuntura onde o mesmo fosse não apenas realista e significativo, mas também alcançasse o maior nível de impacto social. Embora a contextualização se passe em um enredo fictício amparamos todos os detalhes do desenvolvimento em fatos reais.

Para tal escolhemos a região conhecida como Liberland, oficialmente República Livre de Liberland (em inglês: Free Republic of Liberland). Liberland é uma micronação, localizada entre a Croácia e a Sérvia. Teve a sua proclamação em abril de 2015, ainda que sem reconhecimento da comunidade internacional, pelo seu então presidente libertário Vit Jedlicka.

O site oficial de Liberland afirma que a nação foi criada devido à atual disputa fronteiriça entre Croácia e Sérvia, em que algumas áreas ao leste do Danúbio são reivindicadas tanto pela Sérvia quanto pela Croácia, enquanto algumas áreas a oeste, incluindo a área de Liberland, são consideradas parte da Sérvia pela Croácia, mas a Sérvia não as reivindica.

Uma vez que a região é uma várzea na parte ocidental do rio Danúbio e não possui infraestrutura foi proposto pelo então presidente a criação de um sistema que disponibilizasse aos futuros cidadãos um mapa da, então, proposta cidade, com os imóveis disponíveis para venda e arrendamento, bem como os pontos de interesses, nomeadamente quiosques, monumentos, praças, espaços governamentais, terminais de multibanco, entre outros.

> Os espaços urbanos constituem os espaços morfologicamente retratados em plantas (desenhos em projeção horizontal) em que descrevem a diversidade de formas (casas, ruas, pontos de interesse, etc.) no interior de um aglomerado populacional urbano de uma aldeia, vila ou cidade

Citação 2 - Enunciado do Trabalho Prático - Parte 1

O Governo pretende implementar seu plano de zoneamento urbano, onde um determinado percentual das habitações será disponibilizado para locação pelo próprio Estado (servindo como fonte de receita para a nação), estas serão construídas por investidores que terão, em contrapartida, o direito a construir uma habitação permanente para uso próprio ou venda direta a outros empreiteiros ou funcionários públicos.

Será proposto, aos investidores, a opção de construção de espaços comunitários e de infraestruturas, auferindo, assim, o direito a construção de imóveis para venda ou imóveis comerciais para exploração per si, como supermercados, clínicas particulares, oficinas mecânicas e restaurantes.

Todos os imóveis para locação contam com vaga de garagem e mobília, conforme determina a legislação em vigor em Liberland.

O governo criou uma secretaria responsável pela gestão da cartografia, zoneamento urbano e da carteira de imóveis – SGCZC, e licenciou três imobiliárias com direitos a explorar a comercialização dos referidos bens. As agências imobiliárias têm acesso a um sistema institucional onde podem aceder informações em tempo real sobre a disponibilidade de um determinado imóvel e informações acerca das características do mesmo, contando, também, com acesso a relatórios gerenciais.

Ao mesmo tempo os futuros cidadãos podem consultar informações sobre os imóveis diretamente, bem como informações sobre o espaço envolvente ao seu local de interesse, como praias, comércio, espaços governamentais e, ainda, informações relevantes sobre o mesmo, como se determinada praia é propensa para miúdos ou tem águas mais revoltas, ou ainda, escolher morar ao lado de uma zona de cultivo de uvas ou de um hospital.

Mapa de contexto

2 - Arquitetura da Solução

Para alcançarmos uma melhor compreensão da situação em causa começamos por modelar o problema pela sua parte mais elementar.

> todos os edifícios e ruas usam um desenho próprio dentro do espaço urbano para possibilitar facilmente a identificação do seu tipo de edifício e ´de outros elementos urbanos (vulgarmente conhecido por legenda).
>
> O *espaço urbano de edifícios* é um espaço matricial 40 x 30 onde estão localizados os diversos edifícios desse espaço; por outro lado, o espaço também possui zonas fixas ocupadas por dois quiosques de venda de artigos (jornais, recordações, etc.), quatro monumentos e seis máquinas de multibanco.

Citação - 3 - Enunciado do Trabalho Prático - Parte 1

> Na sequência do enunciado do trabalho apresentado na primeira parte, o *espaço urbano de edifícios*, como qualquer espaço urbano, tem de possuir uma planta (desenho de projeção horizontal), como analogia esquemática aproximada (desenho aproximado) da figura ilustrada abaixo, com uma zona de aglomerado populacional urbano (aldeia, vila ou cidade), com casas de habitação (permanente, aluguer e venda), casas de prestação de serviços (clínicas, farmácias, comércio e advocacia), pontos de interesse (igrejas, monumentos e teatros), edifícios comunitários (câmaras municipais, juntas de freguesia e casas de povo), quiosques de venda de artigos, máquinas de multibanco e vias de comunicação de acesso público (ruas, avenidas e vielas) . A planta é um espaço urbanizado matricial de 40 x 30.

Citação - 4 - Enunciado do Trabalho Prático - Parte 2

Chegando, assim, a uma representação matricial que engloba os elementos mais comuns em um espaço urbano, inclusive elementos de topografia relevantes. Este mapa foi a base do desenvolvimento deste sistema.

Representação matricial

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	^	^	^	^	R	R	R	R	R	R	R	R	R	G	+	Ø	Ø	Ø	Ø	Ø	Ø	Ø	P	P	P	»	»	»	»	»
2	^	^	^	^	^	R	R	R	R	R	R	R	R	G	+	C	V	A	A	A	H	H	+	P	P	»	»	»	»	»
3	^	^	^	^	^	^	R	R	R	R	R	R	R	G	+	C	V	A	A	A	H	H	+	P	P	»	»	»	»	»
4	^	^	^	^	^	^	^	^	R	R	R	R	R	C	+	C	V	A	A	A	H	H	+	P	P	P	»	»	»	»
5	^	^	^	^	^	^	^	^	^	^	^	+	+	+	+	+	+	+	+	+	+	+	+	+	+	P	»	»	»	»
6	^	^	^	^	^	^	^	+	+	+	+	+	T	T	+	A	A	A	A	+	@	@	@	@	+	P	»	»	»	»
7	^	^	^	^	^	^	+	+	H	H	H	+	T	T	+	A	A	A	A	+	@	@	@	@	+	P	»	»	»	»
8	^	^	^	^	+	+	+	+	H	H	H	+	T	T	+	A	A	A	A	+	@	@	@	@	+	P	»	»	»	»
9	^	^	^	+	+	Ø	Ø	+	H	H	H	+	T	T	+	A	A	A	A	+	@	@	@	@	+	P	P	»	»	»
10	^	^	+	+	Ø	Ø	Ø	+	H	H	H	+	Ø	Q	+	+	+	+	+	+	+	+	+	+	+	+	P	»	»	»
11	^	+	+	Ø	Ø	Ø	Ø	+	H	H	H	+	A	A	+	H	V	A	$	+	Ø	Ø	Ø	+	A	+	P	»	»	»
12	^	+	Ø	Ø	Ø	Ø	Ø	+	H	H	H	+	A	A	+	H	V	A	C	+	Ø	M	Ø	+	A	+	P	»	»	»
13	^	+	Ø	Ø	Ø	Ø	Ø	+	H	H	H	+	@	@	+	H	V	A	C	+	Ø	Ø	Ø	+	A	+	P	»	»	»
14	^	+	Ø	Ø	Ø	Ø	Ø	+	H	H	H	+	@	@	+	H	V	A	C	+	+	+	+	+	A	+	P	»	»	»
15	^	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	A	A	A	A	A	+	P	P	»	»
16	^	R	A	+	C	C	C	C	+	A	A	C	+	C	C	G	C	C	+	A	A	A	H	H	H	+	P	P	»	»
17	R	R	A	+	C	C	C	C	+	A	A	C	+	G	G	G	G	C	+	A	A	A	H	H	H	+	P	P	»	»
18	R	R	A	+	C	C	C	C	+	A	A	C	+	G	G	G	G	C	+	A	A	A	H	H	H	+	P	P	»	»
19	R	R	A	+	C	C	C	C	+	A	A	C	+	C	C	G	C	C	+	A	A	A	C	C	C	+	P	P	»	»
20	R	R	A	+	A	A	A	C	+	A	A	C	+	+	+	+	+	+	+	+	+	+	+	+	+	+	@	@	@	»
21	R	R	H	+	Ø	Ø	Ø	C	+	+	+	+	+	+	V	V	A	A	+	C	C	C	A	A	C	+	P	P	»	»
22	R	R	H	+	Ø	M	Ø	C	+	@	M	Ø	@	+	V	V	A	A	+	C	C	C	A	A	C	+	P	P	»	»
23	R	R	H	+	Ø	Ø	Ø	C	+	@	@	@	@	+	V	V	A	A	+	@	@	@	A	A	C	+	P	P	P	»
24	R	R	H	+	H	H	H	C	+	@	@	@	@	+	H	H	H	H	+	@	@	@	A	A	C	+	P	P	P	»
25	R	R	V	+	H	H	H	C	+	@	@	@	@	+	H	H	H	H	+	@	@	@	A	A	C	+	P	P	P	»
26	R	R	V	+	H	H	H	C	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	$	P	P	»
27	R	R	V	+	H	H	H	C	+	$	V	A	A	A	A	A	C	C	+	T	T	T	T	T	T	+	Q	P	P	»
28	R	R	V	+	H	H	H	C	+	V	V	A	A	A	A	A	Ø	Ø	+	T	T	T	T	T	T	+	@	P	P	P
29	R	R	V	+	A	A	A	C	+	H	H	A	A	A	A	A	M	Ø	+	T	T	T	T	T	T		^	P	P	P
30	R	R	V	+	A	A	A	C	+	H	H	C	C	C	C	C	Ø	Ø	+	T	T	T	T	T	T	^	^	^	P	P
31	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	^	^	^	^	^	P
32	+	V	V	V	V	V	V	V	+	A	A	A	A	A	@	@	R	R	R	R	R	R	R	R	^	^	^	^	^	^
33	+	V	V	V	V	V	V	V	+	A	A	A	A	A	@	@	R	R	R	R	R	R	R	R	^	^	^	^	^	^
34	+	V	V	V	V	V	V	V	+	A	C	C	C	C	C	C	R	R	R	R	R	^	^	^	^	^	^	^	^	^
35	+	+	+	+	+	+	+	+	+	C	C	C	C	C	C	C	R	R	R	R	R	G	G	^	^	^	^	^	^	^
36	+	C	A	V	H	H	A	V	+	+	+	+	+	+	+	+	+	+	+	+	+	G	G	G	^	^	^	^	^	^
37	+	C	A	V	H	H	A	V	+	C	A	A	V	V	A	A	C	C	+	C	C	G	G	G	G	^	^	^	^	^
38	+	+	+	+	+	+	+	+	+	G	A	A	V	V	A	H	H	H	+	H	H	G	G	G	G	G	^	^	^	^
39	R	R	R	R	R	R	R	R	+	G	A	A	V	V	A	H	+	+	+	+	H	G	G	G	G	G	G	^	^	^
40	R	R	R	R	R	R	R	R	+	G	^	^	^	^	^	H	H	H	H	H	G	G	G	G	G	G	G	G	^	^

Legenda:

+	Rua
V	Vivenda
A	Apartamento
C	Comércio

H	Habitação Permanente
Ø	Parque e Praça
^	Montanha
P	Praia

T	Terreno
R	Rural
»	Rio
G	Governo

$	Multibanco
M	Monumento
Q	Quiosque
@	Espaço Público

Tendo posto nosso ponto de partida, precisávamos definir quem seria o nosso público alvo, ou seja, quem de fato seriam os nossos utilizadores. E ainda quais as os problemas que deveríamos solucionar para este usuário.

> Com a planta matricial do *espaço urbano de edifícios* torna-se possível aos cidadãos (utilizadores) conhecer quais os edifícios desse espaço que estão disponíveis para o arrendamento, bem como para a compra de habitações. Deverá ser possível aos cidadãos (utilizadores) a consulta detalhada de informações sobre os edifícios disponíveis para o arrendamento e para a compra, bem como elementos urbanos associados a ponto de interesse.

Citação - 5 - Enunciado do Trabalho Prático - Parte 1

> interessadas em comprar casas ou procurar casas para arrendar. O agente imobiliário deve disponibilizar informações detalhadas às pessoas interessadas na compra de casa (nome do proprietário, morada, localidade, tipologia da habitação, registo predial, área da habitação, classe do certificado energético e preço) ou no aluguer de casa (nome do proprietário, morada, localidade, número de identificação fiscal, tipologia da habitação, certificado de habitabilidade e renda mensal). Todos os edifícios da planta devem estar devidamente identificados na sua funcionalidade.
>
> O agente imobiliário deve ter o conhecimento da situação das casas vendidas e alugadas (em termos de pessoas interessadas, de áreas de habitação, de valores e de sustentabilidade energética) por mês para efeitos de gestão.

Citação - 6 - Enunciado do Trabalho Prático - Parte 2

Constatamos assim que, a princípio, teríamos dois tipos de utilizadores, os cidadãos e as imobiliárias, onde os problemas solucionados para ambos seriam, em um primeiro momento, iguais, divergindo apenas para as imobiliárias, no que toca a parte de relatórios gerenciais.

Entretanto, sob o nosso contexto inicialmente proposto, identificamos um terceiro utilizador, que seria o Estado de Liberland, mais especificamente a SGCZC, e portanto teríamos que ter um utilizador com poderes para incluir os elementos urbanos, ou seja, construir o mundo, bem como realizar as operações fundamentais em uma base de dados (CRUD) e ainda realizar testes no sistema que garantam a certeza quanto ao estado do sistema.

Neste ponto já tínhamos por completo o levantamento de requisitos e iniciamos então a modelagem de nossas estruturas de dados. Neste ponto definimos também para onde penderia a balança de nosso *trade-off* entre processamento e memória.

Uma vez que pelas dimensões do mundo (40x30) vamos lidar com um número máximo de até 1200 elementos, que este valor pode aumentar de forma exponencial em caso de expansão do mundo, optamos por salvaguardar a memória.

Tendo isto em conta tivemos uma destacável atenção para a escolha dos tipos, optamos por usar todo o range de opções de qualificação de uma variável atentando para o rigor máximo na sua utilização, presando a semântica, manutenibilidade, segurança e boas práticas.

Dentre estas escolhas podemos destacar algumas, sendo:

- Uso de constantes

  Se uma "variável" não vai variar ela é uma constante. Com o propósito de usar de forma correta as propriedades da linguagem e visando a segurança e informação clara para quem futuramente possa utilizar alguma das funções ou procedimentos aqui implementados, optamos por declarar como constante todas as "variáveis" que assim se enquadravam, em especial nos parâmetros. Garantindo, assim, por exemplo, ao utilizador de algum destes módulos que um argumento enviado por referência terá a sua integridade garantida após a manipulação pela função.
- Mensuração do uso da variável

  Avaliamos a real finalidade e utilização de cada atributo que queríamos representar através de variáveis, para que assim pudéssemos prever com acuidade qual seria o limite necessário. Desta forma fizemos uso significativo de *unsigned* e *short*, sobretudo em variáveis de controle, como sinalizadores de características – a venda ou a arrendar, status de erros, etc – ou, por exemplo, nas variáveis responsáveis por representar a tipologia do imóvel, o enquadramento da legenda ou sua posição na matriz. São características que não admitem valores negativos e que não vão ultrapassar, por exemplo, 62k que é o limite aproximado deste tipo. Com isso conseguimos uma economia de cerca de 50% por variável, o que resulta em significativa redução no tamanho de um elemento urbano, que se mostra extremamente relevante em um processo escalável. Uma vez que um *int* ocupa 4 bytes e um *short int* ocupa 2 bytes.
- Utilização de *char* e *short int*

  Em alguns casos onde seria normal utilizar inteira implementamos o caractere enquanto onde seria normal um caractere implementamos inteiro e utilizamos funções auxiliares para manipular a formatação destas informações de maneira

adequada na tela. Podemos citar a título de exemplo o NIF que é naturalmente números, mas que optamos por representá-lo em forma de *string* para garantir a semântica da informação, uma vez que pode ser possível que identificadores fiscais comessem com zero o tratamento como inteiro não garante a presença de zeros a esquerda. Entretanto não incluímos nesta *string* os separadores ponto (economia de 2bytes), utilizando uma função para impressão destes pontos no intervalo de 3 dígitos. Já no horário, operamos com ele como número, com uma variável para os minutos e outra para as horas. O que consome 4 bytes, contra no mínimo 5 bytes, ou 6 bytes caso se optasse por armazenar o separador ':' na *string*. Fizemos uso então de uma função para colocar os zeros a esquerda, garantindo o formato de dois dígitos.

- Tamanho das *Strings*

  Definimos os tamanhos das strings sempre através de uma constante simbólica acrescido de +1, para que caso se deseje alterar o tamanho de um determinado campo, isto seja feito de maneira rápida, em um só lugar e sem se preocupar com o caractere terminador.

Desta forma, conforme a modelagem do problema, concluímos que todo o sistema iria girar em torno de uma estrutura central, que era o elemento urbano em si, implementamos, então, uma estrutura principal chamada <elemento>, que representará todas as possibilidades de um ponto no mapa, está terá os seguintes atributos:

Elemento
unsigned short int id: <contém um número de referência único> unsigned short Int categoria: <Enquadramento da Legenda>; char obs[200+1]: <observações pertinentes ao elemento urbano>; estrutura posyx inicio: <o ponto matricial onde o elemento inicia-se>; estrutura posyx tamanho: <a quantidade de linhas e colunas que o elemento ocupa>; estrutura imóvel: <reúne características próprias dos imóveis> estrutura pInteresse: <reúne características próprias dos elementos não disponíveis para venda ou arrendamento> estrutura endereco: <armazena o endereço do elemento>

Implementaremos ainda as seguintes estruturas complementares:

visita
unsigned short int id: <contém a identificação do elemento urbano – chave estrangeira> estrutura data: <contém a data em que a visita foi realizada>

pInteresse
Char descricao[100+1]: <síntese descritiva do elemento> Estrutura horario: <informa o horário de funcionamento do elemento, caso aplique-se>

imovel
unsigned short Int classe: <1 caso seja venda ou 0 caso seja arrendamento> unsigned short int tipologia: <total de quartos do imóvel> unsigned short int área: <dimensão do imóvel em m2> unsigned short int suítes: <total de suítes que o imóvel possuí> unsigned short int certEnerg: <classificação do certificado energético, de A+, b- até F> char regPredial[11]: <número de identificação do imóvel> char certHabit[11]: <certificado de habitabilidade> unsigned long int valor: <preço do imóvel> Estrutura proprietário Estrutura negocio

Negocio
unsidned short int status: <1 caso esteja disponível ou 0 caso esteja vendida ou arrendada> estrutura data: <data da negociação – venda ou arrendamento>

Proprietário
Char nome[100+1] Char nif[9]

Endereço
Char rua[100+1] Char localidade[60+1]

Data
Unsigned short int dia: <dia da data> Unsigned short int mes: <mês da data> Unsigned short int ano: <ano da data>

Horário
unsigned short int aplica: <1 caso aplica-se o horário ou 0 caso não se aplique horário ao elemento> Estrutura hora abertura: <horário de abertura> Estrutura hora encerramento: <horário de encerramento>

hora
unsigned short hora: <inteiro entre 0 e 24> unsigned short minuto: <inteiro entre 0 e 60>

posyx
unsigned short int linha: <inteiro entre 0 e o máximo de linha> unsigned short int coluna: <inteiro entre 0 e o máximo de colunas>

Em termos de armazenamento dos dados, optou-se por manter a base de dados em ficheiros binários – que possuem acesso mais rápido que os ficheiros de texto por não necessitarem de conversão – sendo separados em dois ficheiros distintos, um que diz respeito aos elementos urbanos e outro as visitas – interessados – realizadas.

2.1 – Organização do Projeto

Dada a dimensão do sistema e com fim de facilitar a manutenibilidade optamos por dividir o sistema em arquivos separados conforme sua finalidade. Separando os da seguinte forma:

- Crud

  O módulo CRUD para as funções e procedimentos que realizam as manipulações fundamentas na base de dados.
- Auxiliares

  Módulo responsável por funções que são utilitárias, ou seja, que servem de meio para as funções principais realizarem suas tarefas.
- Design

  Módulo responsável por desenhar as telas de entrada e saída de dados.
- Tester

  Módulo responsável por implementar as rotinas de teste.
- Structs

  Módulo responsável por definir as estruturas de dados do projeto.
- Config

  Módulo responsável pela definição das configurações do sistema.
- Main

  Módulo principal, responsável pela orquestração do sistema.

Para cada módulo utilizou-se o correspondente *header file* para a definição da interface do módulo, bem como se trabalhou com macros condicionais para evitar a reinclusão de códigos.

Cumpre o destaque para o módulo config, que contém as definições de tamanho, de controle e o controle de dependência através de macros condicionais e a visibilidade das variáveis e constantes globais (*extern*).

Foi levado em conta, também, na questão de organização do projeto a sua portabilidade, dando se preferência por desenvolver funções proprietárias para soluções em que as funções nativas apresentam notada incompatibilidade com certos sistemas operativos, como por exemplo, com a limpeza de buffer do teclado. Em outros casos se utilizou meios que garantissem a portabilidade sem a necessidade de alterar código, procedendo-se apenas a recopilação, como foi o caso da função limpar tela, que faz uso da comunicação direta com o shell através da função system, fizemos o uso, então, do operador lógico ‘ou’ para garantir a compatibilidade nos sistemas baseados no GNU e no Windows, com a instrução “clr || clear”.

Outro item de destaque na organização do projeto foi quanto a documentação do projeto. Inspirados pela metodologia ágil demos ênfase na documentação dinâmica e funcional, implementada através de comentários pontuais no código e forte uso da semântica com funções e procedimentos com nomes bem expressivos, a abstenção do uso de “números mágicos” com a substituição por constantes que expressam o real significado daquele número.

Os comentários foram sempre realizados através do formato /* */ para que em caso de ser apagado algum terminador fique imediatamente evidente a falha.

2.2 – Usabilidade e experiência do usuário

Pensamos o sistema para que estive sempre disponível e que conseguisse se recuperar automaticamente das falhas, prevenindo-as sempre que possível.

Além disto a navegação pelo sistema devia ser intuitiva, simples e confortável. Para isso implementamos uma navegação estrutural, *breadcrumbs*, onde o usuário sempre sabe onde se encontra, de onde veio e para onde ainda pode ir. Como no exemplo abaixo:

Figura 1 - breadcrumbs

Cumpre destacar que para implementar esta funcionalidade, de forma automática e a fim de garantir legibilidade ao código, se fez uso da função “cabeçalho” que é uma função com argumentos variáveis, que foi implementado através da livraria stdarg.h. Podendo assim receber quantas *strings* forem necessárias para representar cada nível.

Ainda neste sentido, garantimos sempre ao usuário a opção de desistir da atua escolha e regressar ao estado anterior ou diretamente ao início. Como com a opção “voltar” ou “descartar” as alterações que estão sendo realizadas em um elemento em edição.

Para promover a sensação de confiança e segurança no uso utilizamos comandos sempre bem expressivos, com exemplo do formato ou resposta esperados. E fomos além implementando

uma política de tratamento de erros, onde informamos através de box temáticos – em vermelho – a instrução de como deve ser a resposta correta para cada caso, ou seja, trabalhamos com mensagens de erros específicas para cada caso e não com mensagens genéricas que não traduzem a realidade imediata do problema do usuário.

Apenas em um caso ocorre o encerramento do programa, o que consideramos uma situação para erro fatal, que foi quando não é possível aceder a base de dados, neste caso o sistema é encerrado através do comando *exit* que retorna o código 1, mas antes retorna uma mensagem no terminal que informa o erro, qual ficheiro que se tentou aceder e em qual módulo do CRUD aconteceu a falha.

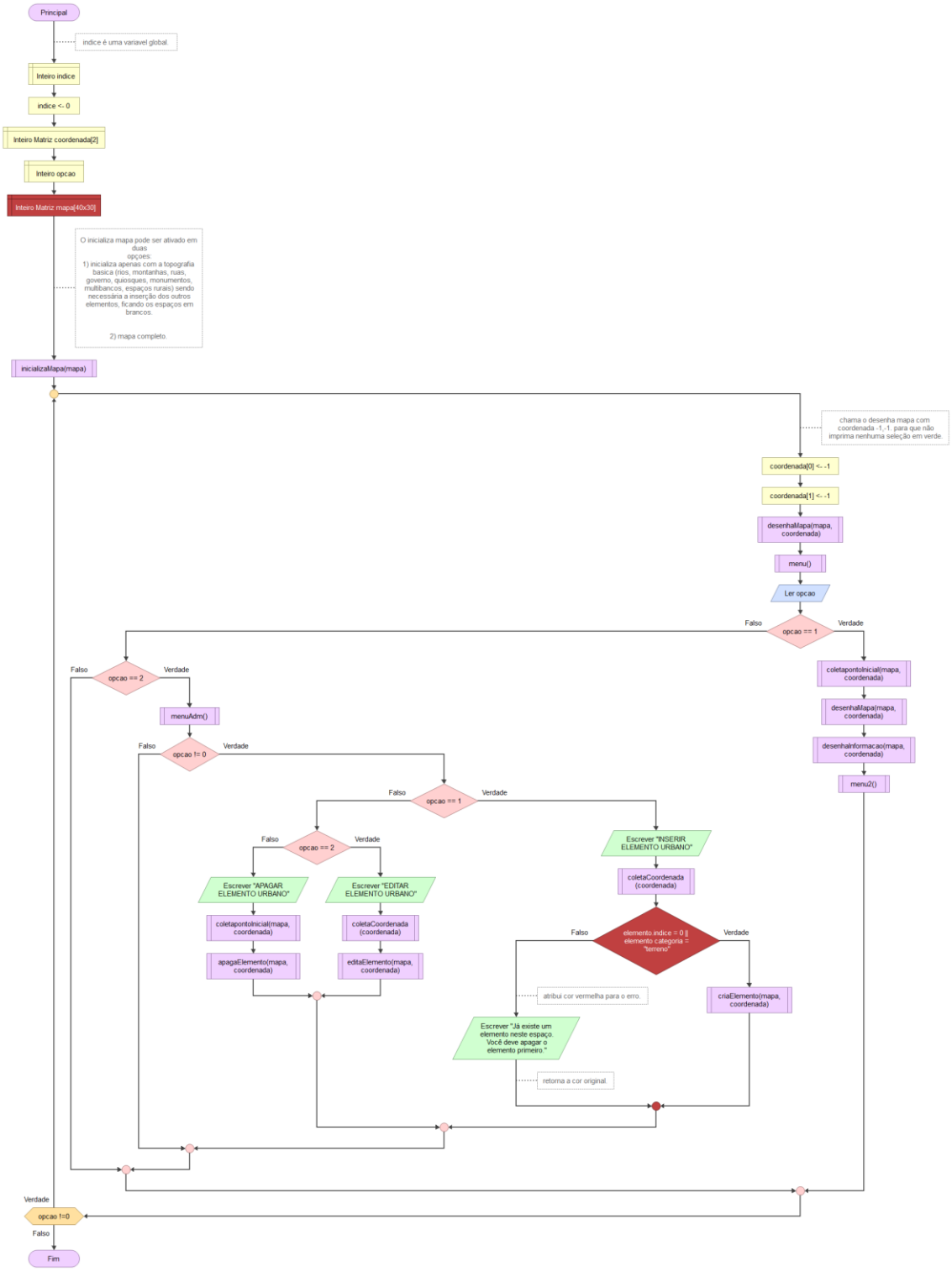
Para se garantir menus sem poluição visual e com opções claras e objetivas, e ainda, garantir a hierarquia de acesso, ou seja, acesso apenas a quem é devidamente autorizado por cada setor, seccionamos o menu conforme o usuário, mas tudo através da mesma interface e de forma transparente.

Para isto garantimos no acesso inicial ao sistema o usuário de mais baixa permissão, o cidadão. Já a imobiliária faz uso do acesso inicial e para as funções exclusivas acessa seu menu simplesmente digitando sua senha de acesso no menu principal, da mesma forma que a SGCZC digita sua senha – neste caso representada pelos números mecanográficos dos discentes realizadores do projeto – e tem acesso ao seu menu exclusivo.

E por fim, para garantir um visual agradável aos olhos, realizamos a aplicação de cores para traduzir significado – como o vermelho no erro e o verde no sucesso – e o design da tela com alinhamentos de blocos e texto, bem como a utilização de padrões – cabeçalho primário, secundário, box, etc.

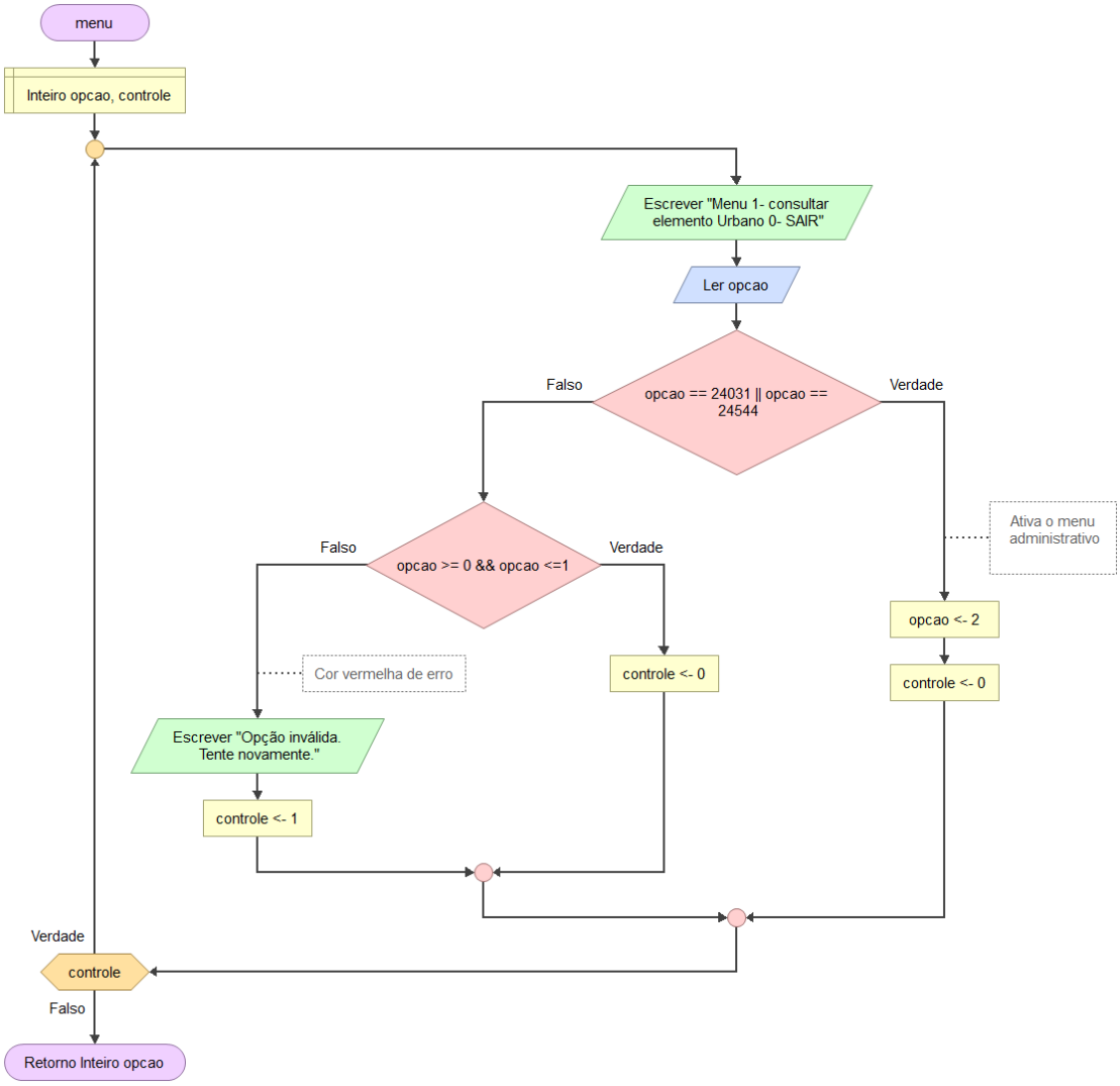
3 – Codificação

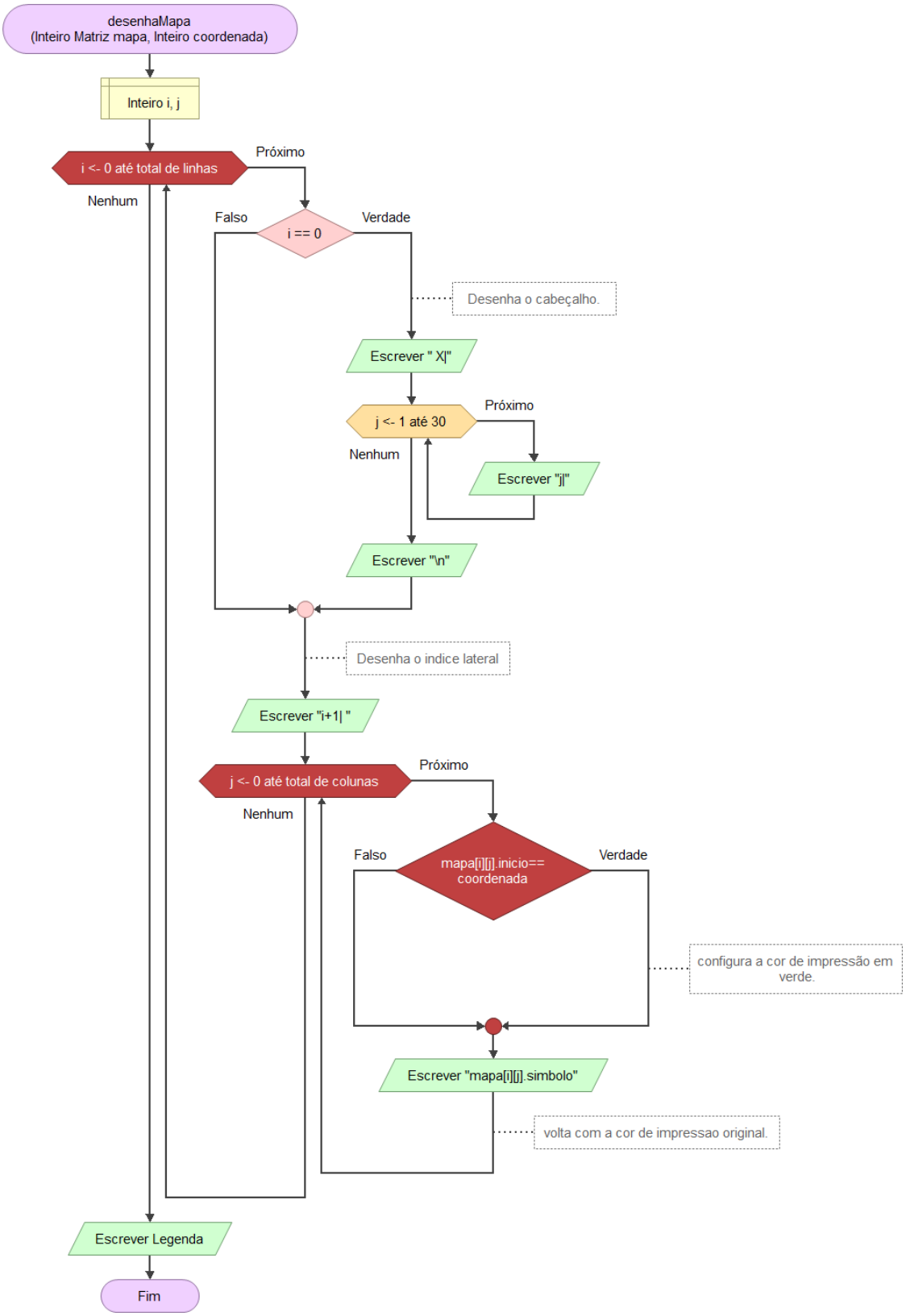
3.1 – Estrutura de Dados

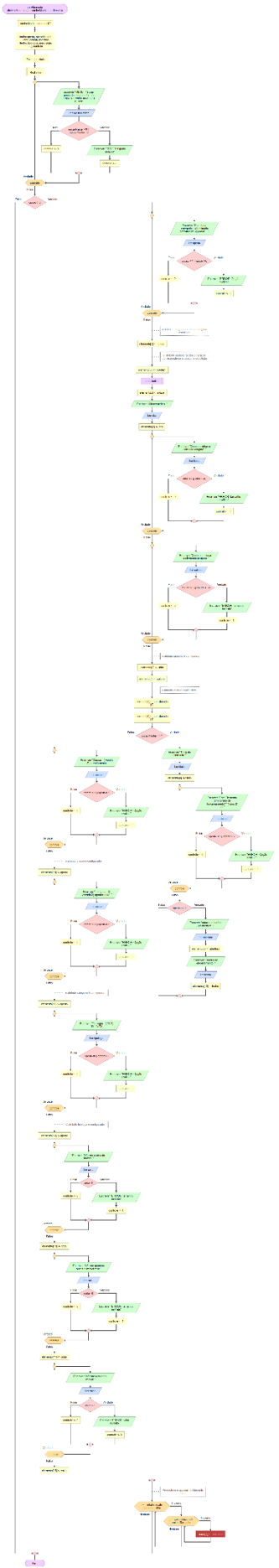


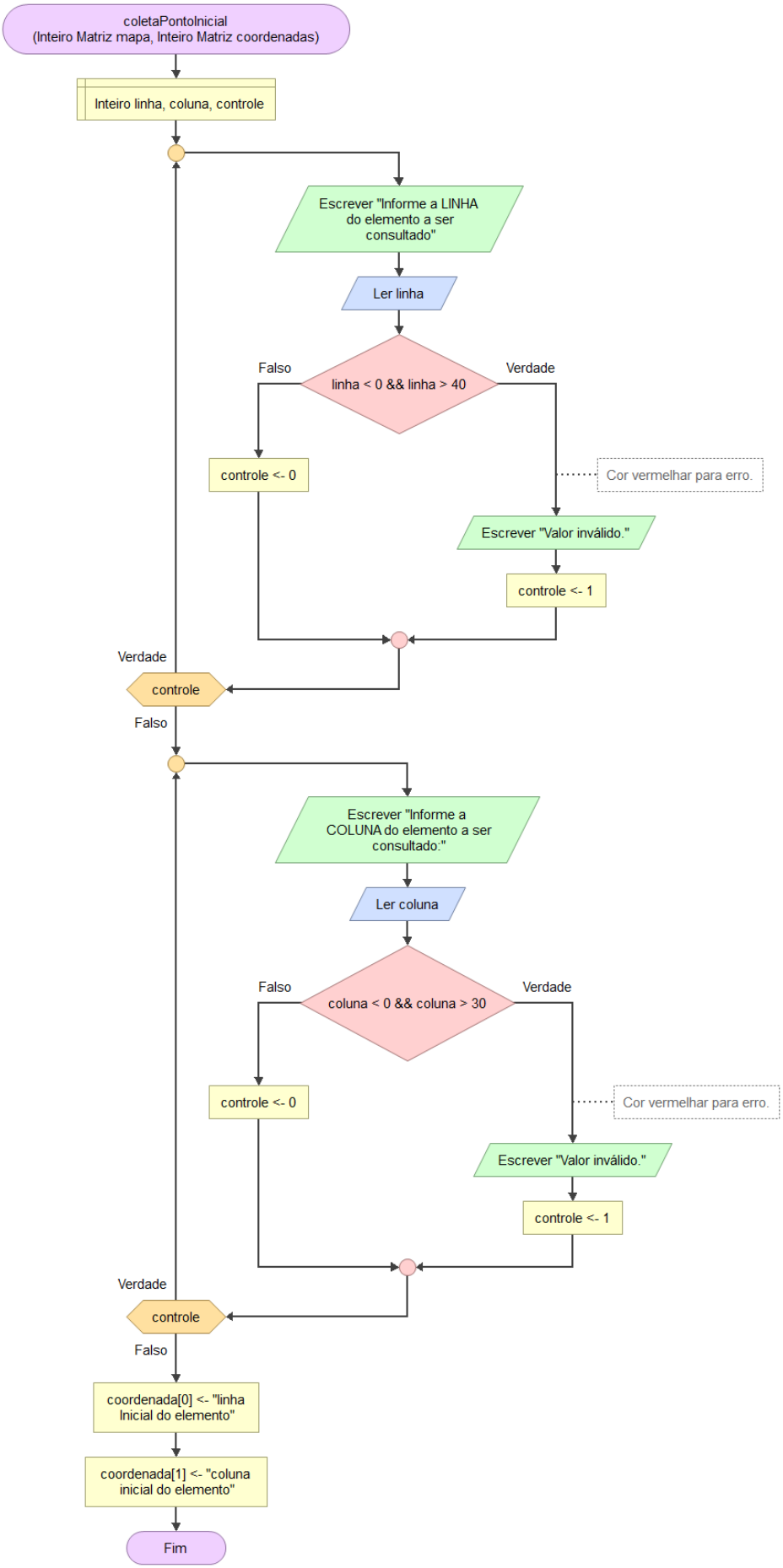
Programa Principal

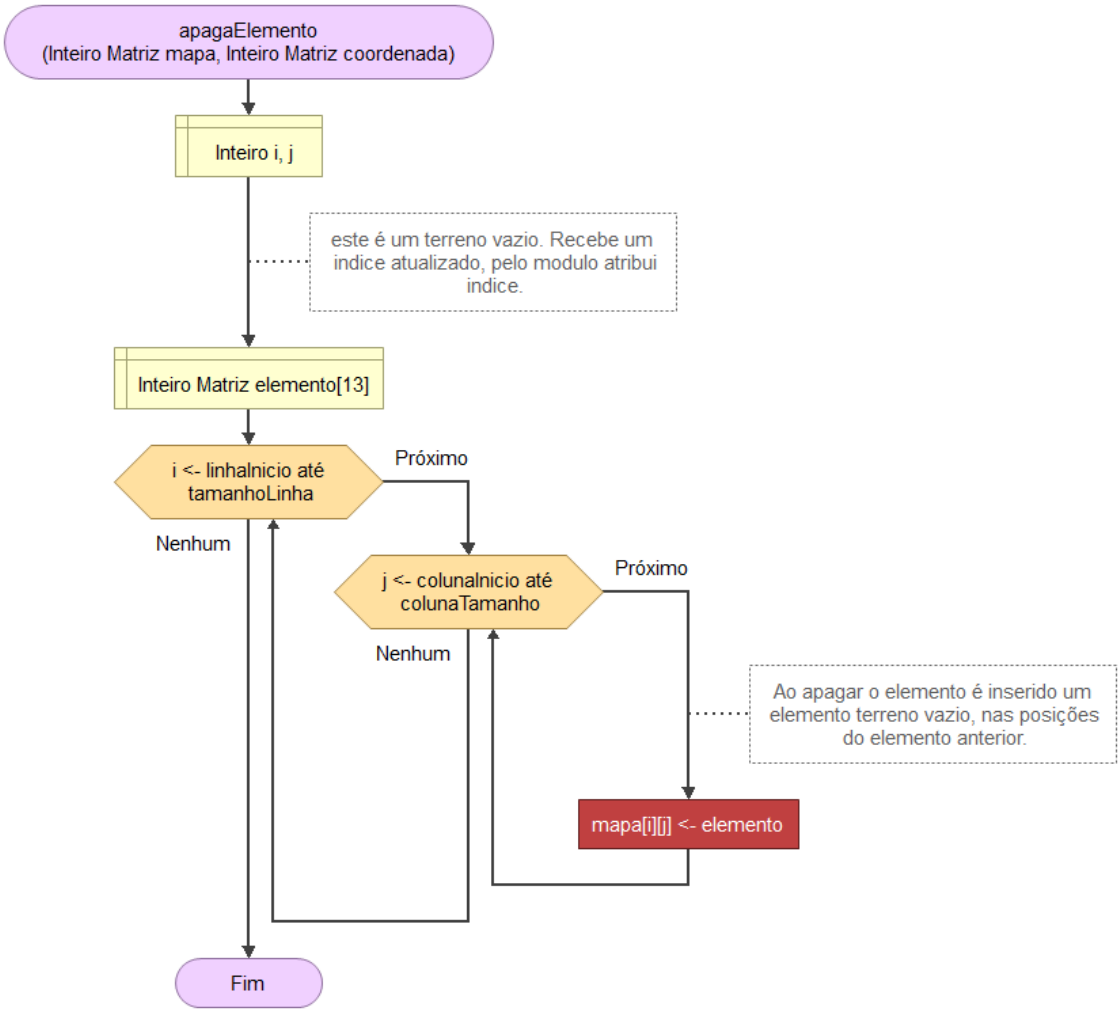
3.2 – Módulos


menu
Inteiro opcao, controle
Escrever "Menu 1- consultar elemento Urbano 0- SAIR"
Ler opcao
opcao == 24031 || opcao == 24544
Falso
Verdade
Ativa o menu administrativo
opcao <- 2
controle <- 0
opcao >= 0 && opcao <=1
Falso
Verdade
Cor vermelha de erro
controle <- 0
Escrever "Opção inválida. Tente novamente."
controle <- 1
Verdade
controle
Falso
Retorno Inteiro opcao


desenhaMapa
(Inteiro Matriz mapa, Inteiro coordenada)
Inteiro i, j
i <- 0 até total de linhas
Próximo
Nenhum
Falso
i == 0
Verdade
Desenha o cabeçalho.
Escrever " X|"
j <- 1 até 30
Próximo
Nenhum
Escrever "j|"
Escrever "\n"
Desenha o indice lateral
Escrever "i+1| "
j <- 0 até total de colunas
Próximo
Nenhum
Falso
mapa[i][j].inicio==
coordenada
Verdade
configura a cor de impressão em
verde.
Escrever "mapa[i][j].simbolo"
volta com a cor de impressao original.
Escrever Legenda
Fim




coletaPontoInicial
(Inteiro Matriz mapa, Inteiro Matriz coordenadas)
Inteiro linha, coluna, controle
Escrever "Informe a LINHA do elemento a ser consultado"
Ler linha
Falso
linha < 0 && linha > 40
Verdade
controle <- 0
Cor vermelhar para erro.
Escrever "Valor inválido."
controle <- 1
Verdade
controle
Falso
Escrever "Informe a COLUNA do elemento a ser consultado:"
Ler coluna
Falso
coluna < 0 && coluna > 30
Verdade
controle <- 0
Cor vermelhar para erro.
Escrever "Valor inválido."
controle <- 1
Verdade
controle
Falso
coordenada[0] <- "linha Inicial do elemento"
coordenada[1] <- "coluna inicial do elemento"
Fim


apagaElemento
(Inteiro Matriz mapa, Inteiro Matriz coordenada)
Inteiro i, j
este é um terreno vazio. Recebe um indice atualizado, pelo modulo atribui indice.
Inteiro Matriz elemento[13]
i <- linhalnicio até tamanhoLinha
Próximo
Nenhum
j <- colunalnicio até colunaTamanho
Próximo
Nenhum
Ao apagar o elemento é inserido um elemento terreno vazio, nas posições do elemento anterior.
mapa[i][j] <- elemento
Fim

3.3 – Código-Fonte Completo

3.3.1 - Auxiliares

Auxiliares.c

```
#include "auxiliares.h"


unsigned short int anoAtual(void){
	unsigned short int anoAtual;
	struct tm *tempoAtual;

	time_t segundos;
	time(&segundos);
	tempoAtual=localtime(&segundos);
	anoAtual=tempoAtual->tm_year+1900;

	return anoAtual;
}
void limpaTela(void){
	system("clear || cls");
}
void zeraVetor(unsigned short int *vetor, const unsigned short int tamanho){
	unsigned short int i;
	for(i=0;i<tamanho;i++)
		vetor[i]=0;
}

unsigned short int noIntervalo(const short int escolha, const short int limiteA, const short int limiteB, const unsigned short int tipoIntervalo, const unsigned short int localnoIntervalo){
	/*Verifica se a escolha encontra-se na faixa de intervalo ou fora dela*/
	switch(tipoIntervalo){
		case INTERVALOFECHADO:
			if(localnoIntervalo == DENTRODOINTERVALO){
				return (escolha >= limiteA && escolha <= limiteB);
			}else if(localnoIntervalo == FORADOINTERVALO){
				return (!(escolha >= limiteA && escolha <= limiteB));
			}else
				return FALHA;
			break;
		case INTERVALOABERTO:
			if(localnoIntervalo == DENTRODOINTERVALO){
				return (escolha > limiteA && escolha < limiteB);
			}else if(localnoIntervalo == FORADOINTERVALO){
				return (!(escolha > limiteA && escolha < limiteB));
			}else
				return FALHA;
			break;
		default:
			return FALHA;
```

```
		}
		return OK;
	}

	unsigned short int validaFormatoData(const char *dataTexto){
		unsigned short int i, dia, mes, ano;

		if(strlen(dataTexto) == 10){ /*String data tem 10 caracteres no formato: dd/mm/aaaa */
			for(i=0;i<10;i++){
				if(i==2 || i==5){/*Posição das barras '/' da data*/
					if(dataTexto[i]!='/')
						return FALHA;
				}else
					if(!isdigit(dataTexto[i]))
						return FALHA;
			}
		}else
			return FALHA;
		/*Neste ponto esta garantido que a data encontra-se no formato dd/mm/aaaa*/
		dia = (((int) dataTexto[0] - 48) * 10) + ((int) dataTexto[1] - 48);
		mes = (((int) dataTexto[3] - 48) * 10) + ((int) dataTexto[4] - 48);
		ano = (((int) dataTexto[6] - 48) * 1000) + ( ((int) dataTexto[7] - 48) * 100) + (((int) dataTexto[8] - 48) * 10) +  ((int) dataTexto[9] - 48);

		if(ano != anoAtual()) /*Nesta rotina vamos operar apenas com datas inclusas no ano vigente.*/
			return FALHA;
		if(!noIntervalo(dia,1,31,INTERVALOFECHADO,DENTRODOINTERVALO))
			return FALHA;
		if(!noIntervalo(mes,1,12,INTERVALOFECHADO,DENTRODOINTERVALO))
			return FALHA;
		if(mes == 2 && dia>29)
			return FALHA;

		return OK;
	}
	unsigned short int validaFormatoHorario(const char *horarioTexto){
		unsigned short int i, hora, minuto;

		if(strlen(horarioTexto) == 5){ /*String data tem 05 caracteres no formato: hh:mm */
			for(i=0;i<5;i++){
				if(i==2){/*Posição do ':' do horário*/
					if(horarioTexto[i]!=':')
						return FALHA;
				}else
					if(!isdigit(horarioTexto[i]))
						return FALHA;
			}
		}else
```

```
                return FALHA;
        /*Neste ponto esta garantido que o horário encontra-se no formato hh:mm*/

        hora = (((int) horarioTexto[0] - 48) * 10) + ((int) horarioTexto[1] - 48);
        minuto = (((int) horarioTexto[3] - 48) * 10) + ((int) horarioTexto[4] - 48);

        if(!noIntervalo(hora,0,23,INTERVALOFECHADO,DENTRODOINTERVALO))
                return FALHA;
        if(!noIntervalo(minuto,0,59,INTERVALOFECHADO,DENTRODOINTERVALO))
                return FALHA;

        return OK;
}
void horarioTexthorarioNum(const char *horarioTexto, unsigned short int
*hora,unsigned short int *minuto){
                *hora =(((int) horarioTexto[0] - 48) * 10) + ((int) horarioTexto[1] - 48);
                *minuto = (((int) horarioTexto[3] - 48) * 10) + ((int) horarioTexto[4] -
48);
}

void dataTextDataNum(const char *dataTexto, unsigned short int *dia,unsigned short
int *mes,unsigned short int *ano){
                *dia = (((int) dataTexto[0] - 48) * 10) + ((int) dataTexto[1] - 48);
                *mes = (((int) dataTexto[3] - 48) * 10) + ((int) dataTexto[4] - 48);
                *ano = (((int) dataTexto[6] - 48) * 1000) + ( ((int) dataTexto[7] - 48) *
100) + (((int) dataTexto[8] - 48) * 10) +  ((int) dataTexto[9] - 48);
}


unsigned short int validaNif(const char *nif){
        unsigned short int i,j;
        char temp[TMAXNIF+1];

        if(strlen(nif)==TMAXNIF){
                for(i=0;i<TMAXNIF;i++){
                        if(!isdigit(nif[i]))
                                return FALHA;
                }
                /*Testa se é um nif com todos os números iguais, ex: 999999999 */
                for(i=0;i<=9;i++){
                        for(j=0;j<TMAXNIF;j++)
                                temp[j]= (char) i+48;
                        temp[j]='\0';
                        if(strcmp(temp,nif)==0)
                                return FALHA;
                }
        }else
                return FALHA;
        return OK;
}

unsigned short int lerInteiro(const char *pergunta){
```

```
		unsigned short int inteiro;
		printf("%s",pergunta);
		scanf("%hu",&inteiro);
		limpaBuf();
		return inteiro;
	}

	void lerTexto(const char *pergunta, char *resposta, const unsigned short int tamanho){
		char formato[12]="%",temp[12];

		sprintf(temp,"%hu[^\n]s",tamanho);
		strcat(formato,temp);
		printf("%s",pergunta);
		scanf(formato,resposta);
		limpaBuf();
	}
	void validaSalvaInteiro(unsigned short int *resposta,const unsigned short int decremento, const char *pergunta, const char *tituloErro, const char *msgErro,const short int limiteA, const short int limiteB, const unsigned short int tipoIntervalo,const unsigned short int localnoIntervalo){
		unsigned short int respostaStatus=OK, respostaTemp;

		do{
			if(respostaStatus==FALHA)
				tratarErro(tituloErro,msgErro);

			respostaTemp=lerInteiro(pergunta);
			if(decremento==DECREMENTO)
				respostaTemp--;
			respostaStatus = noIntervalo(respostaTemp,limiteA,limiteB,tipoIntervalo,localnoIntervalo);

		}while(respostaStatus==FALHA);

		*resposta=respostaTemp;
	}
	void validaSalvaHorario(const char *pergunta, unsigned short int *hora, unsigned short int *minuto){
		unsigned short int leituraStatus=OK;
		char horario[6];

		do{
			//TRATAMENTO DE ERRO AQUI
			if(leituraStatus==FALHA)
				tratarErro("HORÁRIO INVÁLIDO!","Formato: HH:MM com H: 0 a 23 e M: 0 a 59.");

			lerTexto(pergunta,horario,5);
			leituraStatus = validaFormatoHorario(horario);
		}while(leituraStatus==FALHA);
```

```
		horarioTexthorarioNum(horario,hora,minuto);
	}
	void zeraMapa(unsigned short int mapa[TMAXLINHA][TMAXCOLUNA]){
		unsigned short int i, j;

		for(i=0;i<TMAXLINHA;i++)
			for(j=0;j<TMAXCOLUNA;j++)
				mapa[i][j]=NULO;
	}

	unsigned short int criaId(void){
		static unsigned short int inicializado=NAO;
		unsigned short int i, id, inicio = 1, fim = USHRT_MAX;
		ELEMENTO elemento;

	  if(inicializado==NAO){
	    srand((unsigned int) time(NULL));
	    inicializado = SIM;
	  }
	  fim = (fim-inicio)+1;
		do{
			id = inicio + rand() % fim;
			elemento=consultaElemento(id);
		}while(elemento.id != NULO);

		return id;
	}

	ELEMENTO elementoNulo(void){
		ELEMENTO elemento;

		demolidor(&elemento);
		elemento.id=NULO;
		strcpy(elemento.pInteresse.descricao,"ELEMENTO NULO");
		return elemento;
	}

	unsigned short int validaTamanho(const unsigned short int coordenada, const unsigned short int tamanho, const unsigned short int procedimento){
		unsigned short int tamanhoTotal = (coordenada + tamanho)-1;

			if(procedimento == VLINHA)
				return (tamanhoTotal <= TMAXLINHA);
			else
				return (tamanhoTotal <= TMAXCOLUNA);
	}

	unsigned short int validaConstrucao(unsigned short int mapa[TMAXLINHA][TMAXCOLUNA],const unsigned short int *coordenada, const unsigned short int *tamanho){
		/*Verifica se a tentativa de construção é valida, sendo permitido a construção apenas em elemento nulou ou terreno de tamnho 1x1*/
```

```
		ELEMENTO elemento;
		unsigned short int a,b, i,j;

		for(i=0, a=coordenada[0]; i < tamanho[0] ; a++, i++){
			for(j=0, b=coordenada[1]; j < tamanho[1]; b++, j++){

					elemento=consultaElemento(mapa[a][b]);
					if(! (elemento.id== NULO ||
(elemento.categoria==TERRENO && (elemento.tamanho.linha==1 &&
elemento.tamanho.coluna==1))) )
						return FALHA;
			}
		}
		return OK;
	}

	void demolidor(ELEMENTO *elemento){
		/*Info do Elemento*/
		elemento->categoria=TERRENO;
		strcpy(elemento->obs,"");
		elemento->tamanho.linha=1;
		elemento->tamanho.coluna=1;
		strcpy(elemento->endereco.rua,"");
		strcpy(elemento->endereco.localidade,"");
		/*Info do Imóvel*/
		elemento->imovel.classe= VENDA;
		elemento->imovel.tipologia=0;
		elemento->imovel.area = 0;
		elemento->imovel.suites= 0;
		elemento->imovel.certEnerg = 0;
		strcpy(elemento->imovel.regPredial,"");
		strcpy(elemento->imovel.certHabit,"");
		elemento->imovel.valor= 0;
		/*Info do Proprietario*/
		strcpy(elemento->imovel.proprietario.nome,"República Livre de Liberland");
		strcpy(elemento->imovel.proprietario.nif,"000000000");
		/*Info do Negócio*/
		elemento->imovel.negocio.status=DISPONIVEL;
		elemento->imovel.negocio.data.dia=0;
		elemento->imovel.negocio.data.mes=0;
		elemento->imovel.negocio.data.ano=0;
		/*Info do Ponto de Interesse*/
		strcpy(elemento->pInteresse.descricao,"TERRENO PARA CONSTRUÇÃO");
		elemento->pInteresse.horario.aplica=NAOAPLICAVEL;
		elemento->pInteresse.horario.abertura.hora=0;
		elemento->pInteresse.horario.abertura.minutos=0;
		elemento->pInteresse.horario.encerramento.hora=0;
		elemento->pInteresse.horario.encerramento.minutos=0;
	}
	void limpaBuf(void){
		/*Esvazia o buffer do teclado consumindo todos os caracteres que lá estão*/
		char c;
```

```
		while((c=getchar()) != '\n' && c != EOF);
	}
	void duploZero(const unsigned short int numero){
		/*Faz a impresão de números menores que 10 no formato de dois digitos.*/
		if(numero>=0 && numero<10)
			printf("0%hu",numero);
		else
			printf("%hu",numero);
	}
	void imprimeNif(const char *nif){
		/*Faz a impressão formatada do nif, no formato xxx.xxx.xxx*/
		unsigned short int i;

		for(i=0;nif[i]!='\0';i++){
			if(i!=0 && (i%3 ==0))
				putchar('.');
			printf("%c",nif[i]);
		}
	}
```

3.3.2 - Crud

Crud.c

```
#include "crud.h"

char consultaSimbolo(const unsigned short int id){
	ELEMENTO elemento;
	/*Se for um elemento nulo retorna um terreno, para inicializaçao do mundo*/
	if(id == 0)
		return 'T';

	elemento=consultaElemento(id);

	return SIMBOLO[elemento.categoria];
}

ELEMENTO consultaElemento(const unsigned short int id){
	FILE *BD;
	ELEMENTO elemento;

	BD=fopen(ELEMENTOS_BD,"rb");

	if(BD != NULL){
		do{
			fread(&elemento,sizeof(ELEMENTO),1,BD);
			if(elemento.id == id){
				fclose(BD);
				return elemento;
			}
		}while(!feof(BD));

		fclose(BD);
		return elementoNulo();
	}else{
		/*Redireciona a impressão para o buffer de erro*/
		fprintf(stderr,RED"\n\tErro em 'ConsultaElemento'. Falha ao abrir o arquivo: %s ."RESET,ELEMENTOS_BD);
		exit(1);
	}
}

unsigned short int atualizaElemento(const ELEMENTO *elementoAtualizado){
	FILE *BD;
	ELEMENTO temp;

	if(elementoAtualizado->id == 0)
		return FALHA;
	BD=fopen(ELEMENTOS_BD,"r+b");

	do{
```

```
                        fread(&temp,sizeof(ELEMENTO),1,BD);
                        if(temp.id == elementoAtualizado->id){

                                fseek(BD,- (long) sizeof(ELEMENTO), SEEK_CUR);
                                if(fwrite(elementoAtualizado,sizeof(ELEMENTO),1,BD) == 1){

                                        fclose(BD);
                                        return OK;
                                }else{
                                        fclose(BD);
                                        return FALHA;
                                }
                        }
                }while(!feof(BD));
                fclose(BD);
                return FALHA;
        }
        unsigned short int apagaElemento(const unsigned short int id){
                FILE *BD;
                ELEMENTO elemento;
                unsigned short int i,j,a,b, tamanho[2], posInicial[2];
                if(id == 0)
                        return FALHA;

                BD=fopen(ELEMENTOS_BD,"r+b");

                do{
                        fread(&elemento,sizeof(ELEMENTO),1,BD);
                        if(feof(BD))
                                break;
                        if(elemento.id == id){

                                tamanho[0]=elemento.tamanho.linha;
                                tamanho[1]=elemento.tamanho.coluna;
                                posInicial[0]=elemento.inicio.linha;
                                posInicial[1]=elemento.inicio.coluna;

                                demolidor(&elemento);
                                fseek(BD,- (long) sizeof(ELEMENTO), SEEK_CUR);

                                if(fwrite(&elemento,sizeof(ELEMENTO),1,BD) == 1){
                                        fseek(BD,0, SEEK_END);
                                        /*cria um terreno vazio em cada posição ocupada
pelo elemento apagado. */
                                        for(i=0, a=posInicial[0]; i < tamanho[0] ; a++, i++){
                                                for(j=0, b=posInicial[1]; j < tamanho[1]; b++,
j++){
                                                        if(i==0 && j==0)
                                                                b++;
                                                        elemento.id=criaId();
                                                        elemento.inicio.linha=a;
```

```
                                                elemento.inicio.coluna=b;


        fwrite(&elemento,sizeof(ELEMENTO),1,BD);

                                                        }
                                                }
                                                fclose(BD);
                                                return OK;
                                        }else{
                                                fclose(BD);
                                                return FALHA;
                                        }
                                }
                        }while(!feof(BD));
                        fclose(BD);
                        return FALHA;
        }

        unsigned short int insereElemento(const ELEMENTO *elemento){
                FILE *BD;

                BD=fopen(ELEMENTOS_BD,"ab");

                if(fwrite(elemento,sizeof(ELEMENTO),1,BD) == 1){
                        fclose(BD);
                        return OK;
                }else{
                        fclose(BD);
                        return FALHA;
                }
        }

        unsigned short int inicializaMapa(unsigned short int
mapa[TMAXLINHA][TMAXCOLUNA]){
                FILE *BD;
                ELEMENTO elemento;
                unsigned short int i, j, a, b;
                /*Zerando o mapa para usar o 0 como zona null, imprimindo um marcador
especial, para fase de construcao do mundo*/
                zeraMapa(mapa);

                BD=fopen(ELEMENTOS_BD,"rb");

                if(BD != NULL){
                        do{
                                fread(&elemento,sizeof(ELEMENTO),1,BD);
                                if(feof(BD))
                                        break;
                                for(i=0, a=elemento.inicio.linha; i < elemento.tamanho.linha
; a++, i++){
```

```
                                        for(j=0, b=elemento.inicio.coluna; j < elemento.tamanho.coluna; b++, j++){
                                                mapa[a][b]=elemento.id;
                                        }
                                }
                        }while(!feof(BD));

                        fclose(BD);
                        return FALHA;
                }else{
                        /*ERRO FATAL*/
                        fprintf(stderr,RED"\n\tErro em 'inicializaMapa'. Falha ao abrir o arquivo: %s ."RESET,ELEMENTOS_BD);
                        exit(1);
                }
        }

        void negociosMes(unsigned short int vivendasArrendadas[12],unsigned short int vivendasVendidas[12],unsigned short int apartamentosArrendados[12],unsigned short int apartamentosVendidos[12]){
                ELEMENTO elemento;
                FILE *BD;

                zeraVetor(vivendasArrendadas,12);
                zeraVetor(vivendasVendidas,12);
                zeraVetor(apartamentosVendidos,12);
                zeraVetor(apartamentosArrendados,12);

                BD=fopen(ELEMENTOS_BD,"rb");

                if(BD != NULL){
                        do{
                                fread(&elemento,sizeof(ELEMENTO),1,BD);
                                if(feof(BD))
                                        break;
                                        if((elemento.categoria == VIVENDA || elemento.categoria == APARTAMENTO) && elemento.imovel.negocio.status == INDISPONIVEL){
                                                if(elemento.imovel.classe==VENDA){
                                                        if(elemento.categoria == VIVENDA)

        vivendasVendidas[elemento.imovel.negocio.data.mes-1]++;
                                                        else

        apartamentosVendidos[elemento.imovel.negocio.data.mes-1]++;
                                                }else{
                                                        if(elemento.categoria == VIVENDA)

        vivendasArrendadas[elemento.imovel.negocio.data.mes-1]++;
                                                        else

        apartamentosArrendados[elemento.imovel.negocio.data.mes-1]++;
```

```
                                        }

                                    }
                        }while(!feof(BD));
                        fclose(BD);
                }else{
                        /*ERRO FATAL*/
                        fprintf(stderr,RED"\n\tErro em 'negociosMes'. Falha ao abrir o
arquivo: %s ."RESET,ELEMENTOS_BD);
                        exit(1);
                }
        }

        void distribuicaoTipologia(unsigned short int tipologia[6]) {
                ELEMENTO elemento;
                FILE *BD;
                zeraVetor(tipologia,6);

                BD=fopen(ELEMENTOS_BD,"rb");

                if(BD != NULL){
                        do{
                                fread(&elemento,sizeof(ELEMENTO),1,BD);
                                if(feof(BD))
                                        break;
                                if(elemento.categoria == VIVENDA || elemento.categoria ==
APARTAMENTO){
                                        if(elemento.imovel.tipologia>=5)
                                                tipologia[5]++;
                                        else
                                                tipologia[elemento.imovel.tipologia]++;
                                }
                        }while(!feof(BD));
                        fclose(BD);
                }else{
                        fprintf(stderr,RED"\n\tErro em 'distribuicaoTipologia'. Falha ao abrir
o arquivo: %s ."RESET,ELEMENTOS_BD);
                        exit(1);
                }
        }
        void distribuicaoCertEnerg(unsigned short int certEnergetico[8]) {
                ELEMENTO elemento;
                FILE *BD;
                zeraVetor(certEnergetico,8);
                BD=fopen(ELEMENTOS_BD,"rb");

                if(BD != NULL){
                        while(fread(&elemento,sizeof(ELEMENTO),1,BD)){
                                if(elemento.categoria == VIVENDA || elemento.categoria ==
APARTAMENTO){
                                        certEnergetico[elemento.imovel.certEnerg]++;
                                }
```

```
                }
                fclose(BD);
        }else{
                fprintf(stderr,RED"\n\tErro em 'distribuicaoCertEnerg'. Falha ao abrir
o arquivo: %s ."RESET,ELEMENTOS_BD);
                exit(1);
        }
}
void distribuicaoArea(unsigned short int areas[4]) {
        ELEMENTO elemento;
        FILE *BD;
        zeraVetor(areas,4);
        BD=fopen(ELEMENTOS_BD,"rb");

        if(BD != NULL){
                do{
                        fread(&elemento,sizeof(ELEMENTO),1,BD);
                        if(feof(BD))
                                break;
                        if(elemento.categoria == VIVENDA || elemento.categoria ==
APARTAMENTO){
                                if(elemento.imovel.area < 60){
                                        areas[0]++;
                                }else if(elemento.imovel.area < 100){
                                        areas[1]++;
                                }else if(elemento.imovel.area < 150){
                                        areas[2]++;
                                }else{
                                        areas[3]++;
                                }
                        }
                }while(!feof(BD));
                fclose(BD);
        }else{
                fprintf(stderr,RED"\n\tErro em 'distribuicaoArea'. Falha ao abrir o
arquivo: %s ."RESET,ELEMENTOS_BD);
                exit(1);
        }
}
void distribuicaoPreco(unsigned short int precosVenda[4],unsigned short int
precosArrenda[4]) {
        ELEMENTO elemento;
        FILE *BD;
        zeraVetor(precosVenda,4);
        zeraVetor(precosArrenda,4);
        BD=fopen(ELEMENTOS_BD,"rb");

        if(BD != NULL){
                do{
                        fread(&elemento,sizeof(ELEMENTO),1,BD);
                        if(feof(BD))
                                break;
```

```
                        if(elemento.categoria == VIVENDA || elemento.categoria ==
APARTAMENTO){
                                if(elemento.imovel.classe==VENDA){

                                        if(elemento.imovel.valor <= 80000){
                                                precosVenda[0]++;
                                        }else if(elemento.imovel.valor <= 150000){
                                                precosVenda[1]++;
                                        }else if(elemento.imovel.valor <= 200000){
                                                precosVenda[2]++;
                                        }else{
                                                precosVenda[3]++;
                                        }
                                }else{
                                        if(elemento.imovel.valor <= 350){
                                                precosArrenda[0]++;
                                        }else if(elemento.imovel.valor <= 700){
                                                precosArrenda[1]++;
                                        }else if(elemento.imovel.valor <= 1200){
                                                precosArrenda[2]++;
                                        }else{
                                                precosArrenda[3]++;
                                        }
                                }
                        }
                }while(!feof(BD));
                fclose(BD);
        }else{
                /*ERRO FATAL*/
                fprintf(stderr,RED"\n\tErro em 'distribuicaoPreco'. Falha ao abrir o
arquivo: %s ."RESET,ELEMENTOS_BD);
                exit(1);
        }


}
short int visitas(const unsigned short int id, const unsigned short int mes) {
        VISITA visita;
        int contador=0;
        FILE *BD;

        if(noIntervalo(mes,0,12,INTERVALOFECHADO,FORADOINTERVALO))
                return -1;

        BD=fopen(VISITAS_BD,"rb");

        if(BD != NULL){
                while(fread(&visita,sizeof(VISITA),1,BD)){
                        if(visita.id==id)
                                switch(mes){
                                        case VISITASANO:
                                                if(visita.data.ano==anoAtual())
```

```
                                                        contador++;
                                                break;
                                        default:
                                                /*Retorna as visitas de um mes, no ano atual.*/
                                                if(visita.data.mes==mes && visita.data.ano==anoAtual())
                                                        contador++;
                                        }
                        }
                        fclose(BD);
                        return contador;
                }else{          /*ERRO FATAL*/
                        fprintf(stderr,RED"\n\tErro em 'visitas'. Falha ao abrir o arquivo: %s ."RESET,ELEMENTOS_BD);
                        exit(1);
                }
        }
        unsigned short int distribuicaoInteresse(const unsigned short int mes, const unsigned short int classe){
                ELEMENTO elemento;
                int contador=0;
                FILE *BD;
                BD=fopen(ELEMENTOS_BD,"rb");

                if(BD != NULL){
                        while(fread(&elemento,sizeof(ELEMENTO),1,BD)){

                                if((elemento.categoria == VIVENDA || elemento.categoria == APARTAMENTO) && elemento.imovel.classe==classe)
                                        contador+=visitas(elemento.id,mes);

                        }
                        fclose(BD);
                        return contador;
                }else{
                        /*ERRO FATAL*/
                        fprintf(stderr,RED"\n\tErro em 'distribuicaoInteresse'. Falha ao abrir o arquivo: %s ."RESET,ELEMENTOS_BD);
                        exit(1);
                }
        }
        unsigned short int insereVisita(const VISITA *visita) {
                FILE *BD;
                BD=fopen(VISITAS_BD,"ab");

                if(fwrite(visita,sizeof(VISITA),1,BD) == 1){
                        fclose(BD);
                        return OK;
                }else{
                        fclose(BD);
                        return FALHA;
```

```
        }
    }
```

3.3.3 - Tester

Tester.c

```
    #include "tester.h"
    void tester(void){
        cabecalho(BORDALARG,'*',RECUOSIMPLES, "TESTE DO SISTEMA",3,"Início","Menu Administrativo","Teste do Sistema");


        start_tester("validaFormatoData",tester_validaFormatoData());
      start_tester("noIntervalo", tester_noIntervalo());
      start_tester("validaNif",tester_validaNif());
      start_tester("criaId",tester_criaId());
        start_tester("validaFormatoHorario",tester_validaFormatoHorario());
        start_tester("horarioTexthorarioNum",tester_horarioTexthorarioNum());
        start_tester("dataTextDataNum",tester_dataTextDataNum());
    }

    unsigned short int tester_validaFormatoData(void){

        //Casos validos - Atenção a Exclamação ( ! )
        if(! validaFormatoData("01/01/2020"))
            return FALHA;
        if(! validaFormatoData("31/01/2020"))
            return FALHA;
        if(! validaFormatoData("23/12/2020"))
            return FALHA;
        if(! validaFormatoData("01/01/2020"))
            return FALHA;
        if(! validaFormatoData("29/02/2020"))
            return FALHA;
        //Casos Invalidos - Atenção para NÃO ter Exclamação ( ! )
        if( validaFormatoData("30/02/2020"))
            return FALHA;
        if( validaFormatoData("30-11/2020"))
            return FALHA;
        if( validaFormatoData("30/09-2020"))
            return FALHA;
        if( validaFormatoData("27/04/2019"))
            return FALHA;
        if( validaFormatoData("030/02/2020"))
            return FALHA;
        if( validaFormatoData("10/02/202a"))
            return FALHA;
        if( validaFormatoData("30-02-2020"))
            return FALHA;
```

```
		if( validaFormatoData("17/13/2020"))
			return FALHA;
		if( validaFormatoData("07/7/2020"))
			return FALHA;
		if( validaFormatoData("28/03/2021"))
			return FALHA;

		return OK;
	}

	unsigned short int tester_noIntervalo(void){

		//Casos validos - Atenção a Exclamação ( ! )  - Atenção a Exclamação ( ! )
		if(! noIntervalo(4,1,15,INTERVALOFECHADO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(1,1,19,INTERVALOFECHADO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(22,1,22,INTERVALOFECHADO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(-18,-18,22,INTERVALOFECHADO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(-6,-7,6,INTERVALOFECHADO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(2,1,32,INTERVALOABERTO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(14,1,15,INTERVALOABERTO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(6,1,15,INTERVALOABERTO,DENTRODOINTERVALO) )
			return FALHA;
		if(! noIntervalo(59,0,58,INTERVALOFECHADO,FORADOINTERVALO) )
			return FALHA;
		if(! noIntervalo(74,5,74,INTERVALOABERTO,FORADOINTERVALO) )
			return FALHA;
		if(! noIntervalo(43,43,233,INTERVALOABERTO,FORADOINTERVALO) )
			return FALHA;
		if(!
noIntervalo(1500,MINOPCAOPADRAO,MAXAREA,INTERVALOFECHADO,DENTRODOINTERVA
LO) )
			return FALHA;

		//Casos Invalidos - Atenção para NÃO ter Exclamação ( ! )
		if( noIntervalo(0,0,15,INTERVALOABERTO,DENTRODOINTERVALO)  )
			return FALHA;
		if( noIntervalo(25,1,25,INTERVALOABERTO,DENTRODOINTERVALO)  )
			return FALHA;
		if( noIntervalo(2,3,55,INTERVALOFECHADO,DENTRODOINTERVALO)  )
			return FALHA;
		if( noIntervalo(-24,-23,89,INTERVALOFECHADO,DENTRODOINTERVALO)  )
			return FALHA;
		if( noIntervalo(26,8,25,INTERVALOFECHADO,DENTRODOINTERVALO)  )
			return FALHA;
		if( noIntervalo(73,5,74,INTERVALOABERTO,FORADOINTERVALO) )
```

```
                return FALHA;
        if( noIntervalo(43,43,233,INTERVALOFECHADO,FORADOINTERVALO) )
                return FALHA;
        if( noIntervalo(-5,-5,12,INTERVALOFECHADO,FORADOINTERVALO) )
                return FALHA;

        return OK;
}

unsigned short int tester_validaNif(void){
        unsigned short int i,j;
        char temp[TMAXNIF+1];
        //Casos validos - Atenção a Exclamação ( ! )
        if (! validaNif("289120950") )
                return FALHA;
        if (! validaNif("007123954") )
                return FALHA;
        if (! validaNif("000000001") )
                return 1;//FALHA;
        //Casos Invalidos - Atenção para NÃO ter Exclamação ( ! )
        if(validaNif("2891299509233"))
                return FALHA;
        if(validaNif("dO2934865"))
                return FALHA;
        if(validaNif("983 458 984"))
                return FALHA;
        if(validaNif("983.458.984"))
                return FALHA;
        for(i=0;i<=9;i++)       {
                for(j=0;j<TMAXNIF;j++)
                        temp[j]=(char)i+48;
                temp[j]='\0';
                if(validaNif(temp))
                        return FALHA;
        }

        return OK;
}

unsigned short int tester_criaId(void){
        unsigned short int i,id;

                for(i=0;i<100;i++){
                        id=criaId();
                        if(!(id>0 && id<USHRT_MAX))
                                return FALHA;
                }
                return OK;
}
unsigned short int tester_validaFormatoHorario(void){

        //Casos validos - Atenção a Exclamação ( ! )
```

```
		if(! validaFormatoHorario("23:30"))
			return FALHA;
		if(! validaFormatoHorario("00:30"))
			return FALHA;
		//Casos Invalidos - Atenção para NÃO ter Exclamação ( ! )
		if( validaFormatoHorario("25:30") )
			return FALHA;
		if( validaFormatoHorario("12:60") )
			return FALHA;
		if( validaFormatoHorario("2:40") )
			return FALHA;
		if( validaFormatoHorario("09-45") )
			return FALHA;
		return OK;
	}
	unsigned short int tester_horarioTexthorarioNum(void){
		unsigned short int hora,minuto;

		//Casos validos - Atenção a Exclamação ( ! )
		horarioTexthorarioNum("23:30",&hora,&minuto) ;
		if(! (hora==23 && minuto==30))
			return FALHA;

		horarioTexthorarioNum("00:30",&hora,&minuto) ;
		if(! (hora==0 && minuto==30))
			return FALHA;

		horarioTexthorarioNum("25:45",&hora,&minuto) ;
		if(! (hora==25 && minuto==45))
			return FALHA;


		return OK;
	}

	unsigned short int tester_dataTextDataNum(void){
		unsigned short int dia, mes, ano;


		//Casos validos - Atenção a Exclamação ( ! )
		dataTextDataNum("20/10/1985",&dia,&mes,&ano);
		if(! (dia==20 && mes==10 && ano==1985) )
			return FALHA;
		dataTextDataNum("02/01/2020",&dia,&mes,&ano);
		if(! (dia==2 && mes==1 && ano==2020) )
			return FALHA;
		dataTextDataNum("31/15/9999",&dia,&mes,&ano);
		if(! (dia==31 && mes==15 && ano==9999) )
			return FALHA;

		return OK;
	}
```

```
/* MODELO DE FUNÇAO DE TESTE
-------------------------------------------

unsigned short int tester_ nome(void){

        //Casos validos - Atenção a Exclamação ( ! )
        if(! )
                return FALHA;
        //Casos Invalidos - Atenção para NÃO ter Exclamação ( ! )
        if( )
                return FALHA;

        return OK;
}

-------------------------------------------
*/

/* GERENCIAMENTO DE TESTE */

void start_tester(char *name, unsigned short int result){

        /*#### CABEÇALHO ####*/
        printf("\t\t########################");
   printf("  Teste da função ''%s''  ", name);
   printf("########################");
   printf("\n\n");


        /*#### RESULTADO ####*/

        printf("\t\t- %s foi ", name);
        if(result)
                printf(GREEN " APROVADO " RESET);
        else
                printf(RED " REPROVADO " RESET);

        printf("no teste.");

        /*#### RODAPÉ ####*/
        printf("\n\t\t-----------------------------------------------\n\n");
}
```

3.3.4 - Design

Design.c

```
#include "design.h"



unsigned short int menuPrincipal(void){
	unsigned short int escolha, statusEscolha=OK;


	/*############### CABEÇALHO ###############*/
	cabecalho(BORDALARG,'*',RECUOSIMPLES, "MENU PRINCIPAL",1,"Início");

	/*############### OPÇÕES ###############*/
	bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"");
	bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"     1 - Consultar Elemento Urbano");
	bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"     2 - Registrar Negócio");
	bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"     3 - Gerar Relatório");
	bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"    0 - SAIR");
	borda(BORDALARG,'*',RECUOSIMPLES);

	/*############### LEITURA DA ESCOLHA ###############*/
	do{
		/*TRATAMENTO DE ERRO*/
		if(statusEscolha==FALHA)
			tratarErro("OPÇÃO INVÁLIDA!","Escolha uma opção entre 0 e "SMAXOPCAOPRINCIAL);

		escolha=lerInteiro("\n\t\tEscolha --> ");

		if(escolha == MATRICULA1 || escolha == MATRICULA2 || noIntervalo(escolha,MINOPCAOPADRAO,MAXOPCAOPRINCIAL,INTERVALOFECHADO,DENTRODOINTERVALO) ){
			statusEscolha=OK;
		}else{
			statusEscolha=FALHA;
		}
	}while(statusEscolha==FALHA);

	return escolha;
}

unsigned short int menuAdm(void){
	unsigned short int escolha, statusEscolha=OK;
	/*############### CABEÇALHO ###############*/
	cabecalho(BORDALARG,'*',RECUOSIMPLES, "MENU ADMINISTRATIVO",2,"Início","Menu Administrativo");
```

```
		/*############### OPÇÕES ###############*/
		bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"        1 - Criar
Elemento Urbano");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"       2 - Apagar
Elemento Urbano");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      3 - Atualizar
Elemento Urbano");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"       4 - Testar
Sistema");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"    0 - VOLTAR");
		borda(BORDALARG,'*',RECUOSIMPLES);

		/*############### LEITURA DA ESCOLHA ###############*/
		do{
			/*TRATAMENTO DE ERRO*/
			if(statusEscolha==FALHA)
				tratarErro("OPÇÃO INVÁLIDA!","Escolha uma opção entre 0 e
"SMAXOPCAOADM);

			escolha=lerInteiro("\n\t\tEscolha --> ");

	statusEscolha=noIntervalo(escolha,MINOPCAOPADRAO,MAXOPCAOADM,INTERVAL
OFECHADO,DENTRODOINTERVALO);
		}while(statusEscolha==FALHA);

		return escolha;
	}
	unsigned short int menuNegocio(void){
		unsigned short int escolha, statusEscolha=OK;

		/*############### CABEÇALHO ###############*/
		cabecalho(BORDALARG,'*',RECUOSIMPLES,                                "MENU
NEGOCIAL",2,"Início","Menu de Negociações");

		/*############### OPÇÕES ###############*/
		bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      1 - Registrar
uma Venda/Arrendamento");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      2 - Cancelar
uma Venda/Arrendamento");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      3 - Registrar
uma visita(Interessado)");
		bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"    0 - VOLTAR");
		borda(BORDALARG,'*',RECUOSIMPLES);

		/*############### LEITURA DA ESCOLHA ###############*/
		do{
			/*TRATAMENTO DE ERRO*/
			if(statusEscolha==FALHA)
```

```
                                tratarErro("OPÇÃO INVÁLIDA!","Escolha uma opção entre 0 e
"SMAXOPCAONEGOCIO);

                                escolha=lerInteiro("\n\t\tEscolha --> ");

        statusEscolha=noIntervalo(escolha,MINOPCAOPADRAO,MAXOPCAONEGOCIO,INTER
VALOFECHADO,DENTRODOINTERVALO);

                }while(statusEscolha==FALHA);

                return escolha;
        }

        unsigned short int menuRelatorio(void){
                unsigned short int escolha, statusEscolha=OK;

                /*############### CABEÇALHO ###############*/
                cabecalho(BORDALARG,'*',RECUOSIMPLES,   "RELATÓRIOS",2,"Início","Menu
de Relatórios Gerenciais");

                /*############### OPÇÕES ###############*/
                bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      1 - Imóveis
VENDIDOS por mês");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"      2 - Imóveis
ARRENDADOS por mês");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,  ESQUERDA,"     DISTRIUIÇÃO
DOS IMÓVEIS");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"    3 - Por FAIXA
DE PREÇO");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,   ESQUERDA,"        4 - Por
TIPOLOGIA");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"    5 - Por ÁREA
(m²)");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,   ESQUERDA,"        6 - Por
CERTIFICADO ENERGÉTICO");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,   ESQUERDA,"        7 - Por
INTERESSE - EM IMÓVEIS A VENDA");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES,   ESQUERDA,"        8 - Por
INTERESSE - EM IMÓVEIS À ARRENDAR");
                bordaVertical(BORDALARG,'*',RECUOSIMPLES, ESQUERDA,"   0 - VOLTAR");
                borda(BORDALARG,'*',RECUOSIMPLES);

                /*############### LEITURA DA ESCOLHA ###############*/
                do{
                        /*TRATAMENTO DE ERRO*/
                        if(statusEscolha==FALHA)
                                tratarErro("OPÇÃO INVÁLIDA!","Escolha uma opção entre 0 e
"SMAXOPCAORL);

                        escolha=lerInteiro("\n\t\tEscolha --> ");
```

```
	statusEscolha=noIntervalo(escolha,MINOPCAOPADRAO,MAXOPCAORL,INTERVALOFECHADO,DENTRODOINTERVALO);

		}while(statusEscolha==FALHA);

		return escolha;
	}

	void lerPosix(unsigned short int *coordenadas, unsigned short int *tamanho, const unsigned short int procedimento){
		unsigned short int i, totalLeitura=2, statusEscolha, limiteSuperior;

		if(procedimento == TAMANHO){
			borda(BORDALARG,'-',RECUOSIMPLES);

	bordaVertical(BORDALARG,'|',RECUOSIMPLES,ESQUERDA,"TAMANHO DO ELEMENTO URBANO");
			borda(BORDALARG,'-',RECUOSIMPLES);
		} else{
			borda(BORDALARG,'-',RECUOSIMPLES);

	bordaVertical(BORDALARG,'|',RECUOSIMPLES,ESQUERDA,"COOREDENADAS DO ELEMENTO URBANO");
			borda(BORDALARG,'-',RECUOSIMPLES);
		}


		for(i=0;i<totalLeitura;i++){
			statusEscolha=OK;
			limiteSuperior= (i==0)? TMAXLINHA-1 : TMAXCOLUNA-1;

			do{
				/*TRATAMENTO DE ERRO*/
				if(statusEscolha==FALHA)
					if(procedimento == TAMANHO){
						tratarErro("TAMANHO INVÁLIDO!","O tamanho ultrapassa o espaço do mundo.");
					}else{
						if(i==0)
							tratarErro("LINHA INVÁLIDA!","Informe uma LINHA entre 1 e "STMAXLINHA);
						else
							tratarErro("COLUNA INVÁLIDA!","Informe uma COLUNA entre 1 e "STMAXCOLUNA);
					}


				if(i==0)
					if(procedimento == TAMANHO)
						tamanho[i]=lerInteiro("\n\t\tTamanho em LINHAS: ");
```

```
                                        else
                                                coordenadas[i]=lerInteiro("\n\t\tInforme a
LINHA: ");
                                else
                                        if(procedimento == TAMANHO)
                                                tamanho[i]=lerInteiro("\n\t\tTamanho em
COLUNAS: ");
                                        else
                                                coordenadas[i]=lerInteiro("\n\t\tInforme a
COLUNA: ");

                                if(procedimento == COORDENADA)
                                        coordenadas[i]--; /*Decrementa a coordenada
informada, para o usuario utilizar de 1 ao max e nao de zero ao max*/
                                if(procedimento == TAMANHO)
                                        if(i==0)

        statusEscolha=validaTamanho(coordenadas[i],tamanho[i],VLINHA);
                                        else

        statusEscolha=validaTamanho(coordenadas[i],tamanho[i],VCOLUNA);
                                if(procedimento == COORDENADA)
                                        else

        statusEscolha=noIntervalo(coordenadas[i],MINOPCAOPADRAO,limiteSuperior,INTER
VALOFECHADO,DENTRODOINTERVALO);
                        }while(statusEscolha==FALHA);
                }
        }

        void imprimeElemento(const ELEMENTO *elemento){
                char titulo[20];
                putchar('\n');
                /*Define o título da Ficha*/
                if(elemento->categoria == VIVENDA || elemento->categoria ==
APARTAMENTO)
                        strcpy(titulo,"IMÓVEL");
                else
                        strcpy(titulo,"PONTO DE INTERESSE");
                /*Imprime o cabeçalho da ficha*/
                borda(BORDAFICHA,'-',RECUODUPLO);
                bordaVertical(BORDAFICHA,'|',RECUODUPLO,CENTRALIZADO,titulo);
                borda(BORDAFICHA,'-',RECUODUPLO);
                printf("\t\tReferência: %hu",elemento->id);
                printf("\t| Categoria: %s",CATEGORIA[elemento->categoria]);

                if(elemento->categoria == VIVENDA || elemento->categoria ==
APARTAMENTO){
                        /*Imprime as características do imóvel*/
                        printf("\n\t\tClasse: ");
                        (elemento->imovel.classe == VENDA) ? printf("VENDA") :
printf("ARRENDAMENTO");
```

```
                        printf("\t| Status: ");
                        if(elemento->imovel.negocio.status == DISPONIVEL){
                                printf(GREEN "DISPONÍVEL" RESET);

                        }else{
                                if(elemento->imovel.classe == VENDA)
                                        printf(RED "VENDIDA EM ");
                                else
                                        printf(RED "ARRENDADA EM ");
                                printf("%hu/%hu/%hu" RESET,elemento->imovel.negocio.data.dia, elemento->imovel.negocio.data.mes,elemento->imovel.negocio.data.ano);
                        }

                        printf("\n\t\tTipologia: T%hu", elemento->imovel.tipologia);
                        printf("\t| Área útil: %hum²", elemento->imovel.area);
                        printf("\t| Suítes: %hu",elemento->imovel.suites);
                        printf("\t| Cert. Energético: %s",CERTENERG[elemento->imovel.certEnerg]);

                        printf("\n\t\tRegistro Predial: %s",elemento->imovel.regPredial);
                        printf("\t\t| Cert. de Habitação: %s",elemento->imovel.certHabit);

                        printf("\n\t\tValor: EUR %lu,00",elemento->imovel.valor);
                        (elemento->imovel.classe == ARRENDA) ? printf("/mês") : putchar(' ');

                        printf("\t| Interessados: %hu\n",visitas(elemento->id,0));

                        borda(BORDAFICHA,'-',RECUODUPLO);

                bordaVertical(BORDAFICHA,'|',RECUODUPLO,CENTRALIZADO,"Proprietário");
                        borda(BORDAFICHA,'-',RECUODUPLO);

                        printf("\t\tNome: %s",elemento->imovel.proprietario.nome);
                        printf("\t|  NIF: ");
                        imprimeNif(elemento->imovel.proprietario.nif);
                        putchar('\n');

                }else{
                        /*Imprime as características do Ponto de Interesse*/

                        printf("\n\t\tDescrição: %s",elemento->pInteresse.descricao);

                        if(elemento->pInteresse.horario.aplica == APLICAVEL){
                                printf("\n\t\tHorário de Funcionamento: ");
                                duploZero(elemento->pInteresse.horario.abertura.hora);
                                printf("h");
                                duploZero(elemento->pInteresse.horario.encerramento.minutos);
                                printf(" às ");
                                duploZero(elemento->pInteresse.horario.encerramento.hora);
```

```
                                printf("h");
                                duploZero(elemento->pInteresse.horario.abertura.minutos);
                                printf(".");
                        }
                        putchar('\n');
                }
                borda(BORDAFICHA,'-',RECUODUPLO);
                bordaVertical(BORDAFICHA,'|',RECUODUPLO,CENTRALIZADO,"Endereço");
                borda(BORDAFICHA,'-',RECUODUPLO);

                printf("\t\tEndereço: %s - %s\n",elemento->endereco.rua, elemento->endereco.localidade);

                borda(BORDAFICHA,'-',RECUODUPLO);
                bordaVertical(BORDAFICHA,'|',RECUODUPLO,CENTRALIZADO,"Observações");
                borda(BORDAFICHA,'-',RECUODUPLO);

                printf("\t\t%s\n", elemento->obs);
                borda(BORDAFICHA,'-',RECUODUPLO);
        }

        void continuar(const char *mensagem){
                printf("%s",mensagem);
                setbuf(stdin,NULL);
                getchar();
        }

        void recuo(const unsigned short int recuo){
                unsigned short int i;
                for(i=0;i<recuo;i++)
                        putchar('\t');
        }

        void borda(const unsigned short int tamanho, const char simbolo, const unsigned short int margem){
                unsigned short int i;
                recuo(margem);
                for(i=0;i<tamanho;i++)
                        printf("%c",simbolo);
                putchar('\n');
        }

        void bordaVertical(const short int tamanho, const char simbolo, const unsigned short int margem,const unsigned short int alinhamento, const char *titulo){
                unsigned short int i, espacoLivre, espacoEsquerdo, espacoDireita;
                espacoLivre = tamanho - (strlen(titulo)+2);

                recuo(margem);
                putchar(simbolo);
                switch(alinhamento){
                        case 2: /*A ESQUERDA*/
```

```
                                putchar(' ');
                                printf("%s",titulo);
                                for(i=0;i<espacoLivre-1;i++)
                                        putchar(' ');
                                break;

                        case 3: /*A DIREITA*/
                                for(i=0;i<espacoLivre-1;i++)
                                        putchar(' ');
                                printf("%s",titulo);
                                putchar(' ');
                                break;

                        case 1:/*CENTRALIZADO*/
                        default:
                                espacoDireita = espacoLivre/2;
                                espacoEsquerdo = espacoLivre/2;
                                if(espacoLivre % 2 == 0)
                                        for(i=0;i<espacoEsquerdo;i++)
                                                putchar(' ');
                                else
                                        for(i=0;i<espacoEsquerdo+1;i++)
                                                putchar(' ');

                                printf("%s",titulo);
                                for(i=0;i<espacoDireita;i++)
                                        putchar(' ');
                }
                putchar(simbolo);
                putchar('\n');
        }

        void cabecalho(const unsigned short int tamanho, const char simbolo, const unsigned
short int margem, const char *titulo, const unsigned short int totsubtitulos, ...){
                unsigned short int i;
                char localizacao[150], *temp;
                localizacao[0]='\0';
                va_list subtitulos;
                printf("\n\n\n");
                borda(tamanho, simbolo, margem);
                bordaVertical(tamanho, simbolo, margem,CENTRALIZADO, "");
                bordaVertical(tamanho, simbolo, margem, CENTRALIZADO,titulo);
                bordaVertical(tamanho, simbolo, margem,CENTRALIZADO, "");

                if(totsubtitulos > 0){
                        va_start(subtitulos, totsubtitulos);
                        temp= va_arg(subtitulos,char *);

                        strcat(localizacao,temp);

                        for(i=1;i<totsubtitulos;i++){
                                strcat(localizacao," > ");
```

```
                                temp= va_arg(subtitulos,char *);
                                strcat(localizacao,temp);
                        }
                        va_end(subtitulos);
                        borda(tamanho, '-', margem);
                        bordaVertical(tamanho, simbolo, margem, ESQUERDA, localizacao);
                }
                borda(tamanho, simbolo, margem);
                putchar('\n');
        }

        int criaElemento(const unsigned short int id, const unsigned short int coordenadas[2],
const unsigned short int tamanho[2]){
                unsigned short int statusEscolha=OK, escolha, i,j;
                ELEMENTO elemento;

                /*############### CABEÇALHO ###############*/
                cabecalho(BORDALARG,'*',RECUOSIMPLES, "CRIANDO NOVO ELEMENTO
URBANO",3,"Início","Menu Administrativo","Criar Elemento Urbano");

                borda(BORDALARG,'-',RECUOSIMPLES);
                bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"ESCOLHA DA
CATEGORIA");
                borda(BORDALARG,'-',RECUOSIMPLES);

                desenhaLegenda(NLEGENDA);
                validaSalvaInteiro(&elemento.categoria,DECREMENTO,"\n\tCategoria:
","CATEGORIA INVÁLIDA!","Escolha uma CATEGORIA entre 1 e
16",MINOPCAOPADRAO,15,INTERVALOFECHADO,DENTRODOINTERVALO);

                limpaTela();
                if(elemento.categoria== VIVENDA || elemento.categoria==APARTAMENTO){
                        /*é um imóvel*/
                        /*############### CABEÇALHO ###############*/
                        cabecalho(BORDALARG,'*',RECUOSIMPLES, "CRIANDO UM
IMÓVEL",4,"Início","Menu Administrativo","Criar Elemento Urbano", "Criar Imóvel");


        validaSalvaInteiro(&elemento.imovel.classe,DECREMENTO,"\n\tInforme a CLASSE do
imóvel [1 - ARRENDAMENTO  | 2 - VENDA]: ","CLASSE INVÁLIDA!","Escolha uma CLASSE entre
1 e 2.",MINOPCAOPADRAO,1,INTERVALOFECHADO,DENTRODOINTERVALO);


        validaSalvaInteiro(&elemento.imovel.tipologia,SEMDECREMENTO,"\n\tQuartos:
","QUANTIDADE DE QUARTOS INVÁLIDA!","Escolha uma QUANTIDADE entre 0 e
"SMAXQUARTOS,MINOPCAOPADRAO,MAXQUARTOS,INTERVALOFECHADO,DENTRODOINTE
RVALO);


        validaSalvaInteiro(&elemento.imovel.suites,SEMDECREMENTO,"\n\tTotal de suítes:
","QUANTIDADE DE SUÍTES INVÁLIDA!","Nº de suítes deve ser inferior a tipologia do
```

```
imóvel.",MINOPCAOPADRAO,elemento.imovel.tipologia,INTERVALOFECHADO,DENTRODOINTERVALO);


        validaSalvaInteiro(&elemento.imovel.area,SEMDECREMENTO,"\n\tÁrea do imóvel: ","ÁREA INVÁLIDA!","Escolha uma ÁREA entre 0 e "SMAXAREA,MINOPCAOPADRAO,MAXAREA,INTERVALOFECHADO,DENTRODOINTERVALO);

                        do{
                                /*TRATAMENTO DE ERRO*/
                                if(statusEscolha==FALHA)
                                        tratarErro("VALOR INVÁLIDO!","O valor deve ser maior que ZERO.");

                                printf("\n\tValor: ");
                                scanf("%lu",&elemento.imovel.valor);
                                if(elemento.imovel.valor>0)
                                        statusEscolha=OK;

                                else
                                        statusEscolha=FALHA;
                        }while(statusEscolha==FALHA);

                        printf("\n\tESCOLHA UM CERTIFICADO ENERGÉTICO\n\t");
                        for(i=0;i<8;i++){
                                printf("%hu - %s\t|",i+1,CERTENERG[i]);
                        }

        validaSalvaInteiro(&elemento.imovel.certEnerg,DECREMENTO,"\n\n\tCertificado Energético: ","CERTIFICADO ENERGÉTICO INVÁLIDO!","Escolha um certificado entre 1 e 8.",MINOPCAOPADRAO,7,INTERVALOFECHADO,DENTRODOINTERVALO);

                        lerTexto("\n\tRegistro Predial: ",elemento.imovel.regPredial,TMAXREGPREDIAL);
                        lerTexto("\n\tCertificado de Habitabilidade: ",elemento.imovel.certHabit,TMAXCERTHAB);
                        lerTexto("\n\tNome do Proprietário: ",elemento.imovel.proprietario.nome,TMAXNOME);

                        do{
                                /*TRATAMENTO DE ERRO*/
                                if(statusEscolha==FALHA)
                                        tratarErro("NIF INVÁLIDO!","O NIF deve ter exclusivamente 9 digitos.");

                                lerTexto("\n\tNIF do Proprietário: ",elemento.imovel.proprietario.nif,TMAXNIF);

                                statusEscolha = validaNif(elemento.imovel.proprietario.nif);

                        }while(statusEscolha==FALHA);
```

```
                        elemento.imovel.negocio.status=DISPONIVEL;
                        elemento.imovel.negocio.data.dia=0;
                        elemento.imovel.negocio.data.mes=0;
                        elemento.imovel.negocio.data.ano=0;

                }else{
                        /*é um ponto de interesse*/
                        cabecalho(BORDALARG,'*',RECUOSIMPLES, "CRIANDO UM PONTO DE INTERESSE",4,"Início","Menu Administrativo","Criar Elemento Urbano", "Criar Ponto de Interesse");

                        lerTexto("\n\tDescrição: ",elemento.pInteresse.descricao,TMAXDESCRICAO);


        validaSalvaInteiro(&elemento.pInteresse.horario.aplica,DECREMENTO,"\n\tPossui HORÁRIO de funcionamento [1 - NÃO  | 2 - SIM]: ","OPÇÃO INVÁLIDA!","Escolha uma opção entre 1 ou 2.",MINOPCAOPADRAO,1,INTERVALOFECHADO,DENTRODOINTERVALO);

                        if(elemento.pInteresse.horario.aplica == APLICAVEL){
                                validaSalvaHorario("\n\tHORÁRIO de abertura [HH:MM]: ",&elemento.pInteresse.horario.abertura.hora,&elemento.pInteresse.horario.abertura.minutos);
                                validaSalvaHorario("\n\tHORÁRIO de encerramento [HH:MM]: ",&elemento.pInteresse.horario.encerramento.hora,&elemento.pInteresse.horario.encerramento.minutos);
                        }else{
                                elemento.pInteresse.horario.abertura.hora=0;
                                elemento.pInteresse.horario.abertura.minutos=0;
                                elemento.pInteresse.horario.encerramento.hora=0;
                                elemento.pInteresse.horario.encerramento.minutos=0;
                        }
                }

                lerTexto("\n\tRua c/ nº: ",elemento.endereco.rua,TMAXRUA);
                lerTexto("\n\tLocalidade: ",elemento.endereco.localidade,TMAXRUA);
                lerTexto("\n\tObservações: ",elemento.obs,TMAXOBS);

                elemento.id=criaId();
                elemento.inicio.linha=coordenadas[0];
                elemento.inicio.coluna=coordenadas[1];
                elemento.tamanho.linha=tamanho[0];
                elemento.tamanho.coluna=tamanho[1];

                insereElemento(&elemento);
                boxSucesso("SUCESSO","Elemento criado com Sucesso.");
        }

        void tratarErro(const char *titulo, const char *instrucao){
                char erroTitulo[MAXTITULOERRO]="ERRO: ";
                strcat(erroTitulo,titulo);
```

```
                printf("\n");
                printf(RED);
                borda(BORDAERROR,'-',RECUOERRO);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                bordaVertical(BORDAERROR,'|',RECUOERRO, CENTRALIZADO,erroTitulo);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                borda(BORDAERROR,'-',RECUOERRO);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,instrucao);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                borda(BORDAERROR,'-',RECUOERRO);
                printf(RESET);
                continuar("\t\t\t\tPressione qualquer tecla para tentar novamente. . . ");

                printf("\n");
        }
        void boxSucesso(const char *titulo, const char *instrucao){
                printf("\n");
                printf(GREEN);
                borda(BORDAERROR,'-',RECUOERRO);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                bordaVertical(BORDAERROR,'|',RECUOERRO, CENTRALIZADO,titulo);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                borda(BORDAERROR,'-',RECUOERRO);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,instrucao);
                bordaVertical(BORDAERROR,'|',RECUOERRO, ESQUERDA,"");
                borda(BORDAERROR,'-',RECUOERRO);
                printf(RESET);
                continuar("\t\t\t\tPressione qualquer tecla para continuar. . . ");

                printf("\n");
        }
        void desenhaLegenda(unsigned short int i){
                if(i==SLEGENDA){

                        putchar('\n');
                        printf("\n\t\t+ - RUA              H - HABITAÇÃO PERMANENTE      T -
TERRENO           $ - MULTIBANCO");
                        printf("\n\t\tV - VIVENDA       Ø - PARQUE OU PRAÇA         R - ÁREA
RURAL         M - MONUMENTO");
                        printf("\n\t\tA - APARTAMENTO    ^ - MONTANHA               » - RIO
Q - QUIOSQUE");
                        printf("\n\t\tC - COMÉRCIO        P - PRAIA                  G - ÁREA
GOVERNAMENTAL   @ - ESPAÇO PÚBLICO");
                        putchar('\n');
                }else{
                        putchar('\n');
                        printf("\n\t\t1 - RUA              5 - HABITAÇÃO PERMANENTE      9 -
TERRENO           13 - MULTIBANCO\n");
                        printf("\n\t\t2 - VIVENDA       6 - PARQUE OU PRAÇA        10 - ÁREA
RURAL         14 - MONUMENTO\n");
                        printf("\n\t\t3 - APARTAMENTO    7 - MONTANHA               11 - RIO
15 - QUIOSQUE\n");
```

```
                printf("\n\t\t4 - COMÉRCIO        8 - PRAIA                12 - ÁREA
GOVERNAMENTAL   16 - ESPAÇO PÚBLICO\n");
                putchar('\n');
        }
    }

    void desenhaMapa(unsigned short int mapa[TMAXLINHA][TMAXCOLUNA], const
unsigned short int id){
        int i, j;

        for(i=0;i<TMAXLINHA;i++){
            printf("\n\t\t");
            if(i==0){
                /*desenha o head*/
                printf(" X|");
                for(j=1;j<=TMAXCOLUNA;j++){
                    printf("%2i|",j);
                }
                printf("\n\t\t");
            }

            printf("%2i| ",i+1); /*desenha o indice lateral*/
            /*desenha os elementos do mapa*/
            for(j=0;j<TMAXCOLUNA;j++){

                switch(consultaSimbolo(mapa[i][j])){
                        case '»':
                            printf(BLUE);
                            break;
                        case '^':
                            printf(MARROM);
                            break;
                        case '+':
                            printf(CINZA);
                            break;
                        case 'Ø':
                            printf(VERDEPARQUE);
                            break;
                        case 'R':
                            printf(VERDERURAL);
                            break;
                        case 'P':
                            printf(AREIA);
                            break;
                        case '$':
                            printf(GREEN);
                            break;
                        case 'A':
                            printf(ROXOT);
                            break;
                        case 'V':
                            printf(LARANJAT);
```

```
                                        break;
                                }
                                if(id != 0){
                                        if(mapa[i][j] == id)
                                                printf(AMARELOT);

                                }
                                printf("%c  ", consultaSimbolo(mapa[i][j]));
                                printf(RESET);
                        }
                }
                if(id == MAPAGENERICO)
                        desenhaLegenda(SLEGENDA);
                printf("\n");
        }

        unsigned short int lerCoordenadaTamanho(unsigned short int mapa[TMAXLINHA][TMAXCOLUNA], unsigned short int *coordenadas, unsigned short int *tamanho){

                unsigned short int escolha,statusEscolha=OK,status=OK;

                do{
                        /*TRATAMENTO DE ERRO*/
                        if(status==FALHA){ /*Primeiro tente demolir o local desejado.*/
                                tratarErro("CONSTRUÇÃO NÃO AUTORIZADA!","Sua construção invade um ELEMENTO EXISTENTE.");
                                borda(BORDALARG,'-',RECUOSIMPLES);

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"O QUE DESEJA FAZER?");

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"");

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"0 - VOLTAR AO MENU PRINCIPAL  |  1 - INSERIR NOVAS COORDENADAS E TAMANHO");
                                borda(BORDALARG,'-',RECUOSIMPLES);
                                do{
                                        if(statusEscolha==FALHA)
                                                tratarErro("OPÇÃO INVÁLIDA!","Escolha uma opção entre 0 e 1.");

                                        escolha=lerInteiro("\n\tEscolha: ");

                                        statusEscolha = noIntervalo(escolha,0,1,INTERVALOFECHADO,DENTRODOINTERVALO);
                                }while(statusEscolha==FALHA);
                                if(escolha==0)
                                        return FALHA;
                        }
                        lerPosix(coordenadas,tamanho,COORDENADA);
                        lerPosix(coordenadas,tamanho,TAMANHO);
```

```
                status=validaConstrucao(mapa,coordenadas,tamanho);
        }while(status == FALHA);
        return OK;
    }

    void registraNegocio(const unsigned short int id, const unsigned short int operacao){
        ELEMENTO elemento=consultaElemento(id);
        unsigned short int status=OK;
        char data[TMAXDATA+1];

        limpaTela();
        /*############### CABEÇALHO ###############*/
        if(operacao == CANCELARNEGOCIO)
            cabecalho(BORDALARG,'*',RECUOSIMPLES, "CANCELAR
NEGÓCIO",2,"Início","Cancelar uma VENDA ou ARRENDAMENTO");
        else
            cabecalho(BORDALARG,'*',RECUOSIMPLES, "REGISTRAR
NEGÓCIO",2,"Início","Registrar uma VENDA ou ARRENDAMENTO");
        if(operacao != CANCELARNEGOCIO){
            do{
                /*TRATAMENTO DE ERRO*/
                if(status==FALHA)
                    tratarErro("DATA INVÁLIDA!","Formato válido:
dd/mm/aaaa. Com ano atual.");

                if(elemento.imovel.classe==VENDA)
                    lerTexto("\n\tData da venda [dd/mm/aaaa]:
",data,11);
                else
                    lerTexto("\n\tData do arrendamento [dd/mm/aaaa]:
",data,11);
                    status=validaFormatoData(data);

            }while(status==FALHA);

        dataTextDataNum(data,&elemento.imovel.negocio.data.dia,&elemento.imovel.neg
ocio.data.mes,&elemento.imovel.negocio.data.ano);
            elemento.imovel.negocio.status=INDISPONIVEL;
            atualizaElemento(&elemento);
            boxSucesso("SUCESSO","Negócio registrado com sucesso");
        }else{
            elemento.imovel.negocio.status=DISPONIVEL;
            if(atualizaElemento(&elemento))
                boxSucesso("SUCESSO","Cancelamento realizado com
sucesso");

        }
    }

    void imprimeRLvendidosMes(const unsigned short int classe){
```

```
		unsigned short int i, vivendasArrendadas[12]={0}, vivendasVendidas[12]={0},apartamentosArrendados[12]={0}, apartamentosVendidos[12]={0};
		/*############### CABEÇALHO ###############*/
		if(classe==VENDA)
			cabecalho(BORDALARG,'*',RECUOSIMPLES, "VENDA DE IMÓVEIS POR MESES",3,"Início","Relatório Gerencial","Imóveis Vendidos por Mês");
		else
			cabecalho(BORDALARG,'*',RECUOSIMPLES, "ARRENDAMENTO DE IMÓVEIS POR MESES",3,"Início","Relatório Gerencial","Imóveis Arrendados por Mês");

		negociosMes(vivendasArrendadas,vivendasVendidas,apartamentosArrendados,apartamentosVendidos);
		printf(BOLD"\n\tMÊS\t\t | Vivendas |\t\t| Apartamentos |\t\t| TOTAL |"RESET);

		for(i=0;i<12;i++){
			if(classe==VENDA)

	if(noIntervalo(i,2,7,INTERVALOFECHADO,DENTRODOINTERVALO))
					printf("\n\t%s:\t\t\t%hu\t\t\t%hu\t\t\t %hu",MESES[i],vivendasVendidas[i],apartamentosVendidos[i],vivendasVendidas[i]+apartamentosVendidos[i]);
				else
					printf("\n\t%s:\t\t%hu\t\t\t%hu\t\t\t %hu",MESES[i],vivendasVendidas[i],apartamentosVendidos[i],vivendasVendidas[i]+apartamentosVendidos[i]);
			else

	if(noIntervalo(i,2,7,INTERVALOFECHADO,DENTRODOINTERVALO))
					printf("\n\t%s:\t\t\t%hu\t\t\t%hu\t\t\t %hu",MESES[i],vivendasArrendadas[i],apartamentosArrendados[i],vivendasArrendadas[i]+apartamentosArrendados[i]);
				else
					printf("\n\t%s:\t\t%hu\t\t\t%hu\t\t\t %hu",MESES[i],vivendasArrendadas[i],apartamentosArrendados[i],vivendasArrendadas[i]+apartamentosArrendados[i]);
		}
		printf("\n\n");
	}
	void imprimeDistPreco(void){
		unsigned short int precosVenda[4], precosArrenda[4];
		distribuicaoPreco(precosVenda,precosArrenda);
		cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE IMÓVEIS - POR PREÇO",4,"Início","Relatório Gerencial","Distribuição de Imóveis","Por faixa de preço");

		borda(BORDALARG,'-',RECUOSIMPLES);
		bordaVertical(BORDALARG,'|',RECUOSIMPLES, ESQUERDA,"DISPONÍVEIS PARA ARRENDAMENTO");
		borda(BORDALARG,'-',RECUOSIMPLES);

		printf("\n\tAté 350,00: %hu",precosArrenda[0]);
```

```
                printf("\n\tEntre 351,00 e 700,00:  %hu",precosArrenda[1]);
                printf("\n\tEntre 701,00 e 1200,00:  %hu",precosArrenda[2]);
                printf("\n\tAcima de 1200,00:  %hu",precosArrenda[3]);
                printf("\n\n");
                borda(BORDALARG,'-',RECUOSIMPLES);
                bordaVertical(BORDALARG,'|',RECUOSIMPLES,       ESQUERDA,"DISPONÍVEIS
PARA VENDA");
                borda(BORDALARG,'-',RECUOSIMPLES);

                printf("\n\tAté 80.000,00:  %hu",precosVenda[0]);
                printf("\n\tEntre 80.001,00 e 150.000,00:   %hu",precosVenda[1]);
                printf("\n\tEntre 150.001,00 e 200.000,00:  %hu",precosVenda[2]);
                printf("\n\tAcima de 200.001,00:   %hu",precosVenda[3]);
                printf("\n\n");
        }
        void imprimeDistTipologia(void){
                unsigned short int i, tipologia[6];
                distribuicaoTipologia(tipologia);
                cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE IMÓVEIS -
POR TIPOLOGIA",4,"Início","Relatório Gerencial","Distribuição de Imóveis","Por Tipologia");

                for(i=0;i<6;i++){
                        if(i!=5)
                                printf("\n\tT%hu: %hu",i,tipologia[i]);
                        else
                                printf("\n\tT5 ou +: %hu",tipologia[i]);
                }
                printf("\n\n");
        }
        void imprimeDistArea(void){
                unsigned short int areas[4]={0};
                distribuicaoArea(areas);
                cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE IMÓVEIS -
POR ÁREA",4,"Início","Relatório Gerencial","Distribuição de Imóveis","Por Área");
                printf("\n\tAté 60m²:      %hu",areas[0]);
                printf("\n\tDe 60 a 100m²:  %hu",areas[1]);
                printf("\n\tDe 100 a 150m²: %hu",areas[2]);
                printf("\n\tAcima de 150m²: %hu",areas[3]);
                printf("\n\n");
        }
        void imprimeDistEnerg(void){
                unsigned short int certEnergetico[8];
                distribuicaoCertEnerg(certEnergetico);
                cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE IMÓVEIS -
POR CLASSIFICAÇÃO ENERGÉTICA",4,"Início","Relatório Gerencial","Distribuição de
Imóveis","Por Certificado Energético");

                printf("\n\tA+: %hu\t\tA:  %hu",certEnergetico[0],certEnergetico[1]);
                printf("\n\tB:  %hu\t\tB-: %hu",certEnergetico[2],certEnergetico[3]);
                printf("\n\tC:  %hu\t\tD:  %hu",certEnergetico[4],certEnergetico[5]);
                printf("\n\tE:  %hu\t\tF:  %hu",certEnergetico[6],certEnergetico[7]);
                printf("\n\n");
```

```
	}
	void imprimeRLinteresse(const unsigned short int classe){
		unsigned short int i;
		if(classe== VENDA)
			cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE INTERESSADOS - PARA IMÓVEIS A VENDA",4,"Início","Relatório Gerencial","Distribuição de Imóveis","Por Interessados");
		else
			cabecalho(BORDALARG,'*',RECUOSIMPLES, "DISTRIBUIÇÃO DE INTERESSADOS - PARA IMÓVEIS À ARRENDAR",4,"Início","Relatório Gerencial","Distribuição de Imóveis","Por Interessados");

		for(i=0;i<12;i++){
			if(noIntervalo(i,2,7,INTERVALOFECHADO,DENTRODOINTERVALO))

	printf("\n\t%s:\t\t%hu",MESES[i],distribuicaoInteresse(i+1,classe));
			else

	printf("\n\t%s:\t%hu",MESES[i],distribuicaoInteresse(i+1,classe));
		}
		printf("\n\n");
	}
	void cadastraVisita(unsigned short int id){
		VISITA visita;
		unsigned short int status=OK;
		char data[TMAXDATA+1];

		limpaTela();
		cabecalho(BORDALARG,'*',RECUOSIMPLES, "REGISTRO DE INTERESSADO",3,"Início","Registrar um Negócio","Registrar uma Visita");
		do{
			/*TRATAMENTO DE ERRO*/
			if(status==FALHA)
				tratarErro("DATA INVÁLIDA!","Formato válido: dd/mm/aaaa. Com ano atual.");

			lerTexto("\n\tData da Visita [dd/mm/aaaa]: ",data,11);
			status=validaFormatoData(data);
		}while(status==FALHA);
		dataTextDataNum(data,&visita.data.dia,&visita.data.mes,&visita.data.ano);
		visita.id=id;
		insereVisita(&visita);
		boxSucesso("SUCESSO","Visita registrada com Sucesso.");
		printf("\n\n");
	}
	void telaAtualizaElemento(const unsigned int id){
		ELEMENTO elemento=consultaElemento(id);
		unsigned short int i, statusEscolha, opcao, limiteB;

		do{
			limpaTela();
```

```
                        cabecalho(BORDALARG,'*',RECUOSIMPLES, "EDITAR ELEMENTO
URBANO",3,"Início","Menu administrativo","Edição de elemento urbano");
                        imprimeElemento(&elemento);
                        putchar('\n');

                        borda(BORDALARG,'-',RECUOSIMPLES);

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"ESCOLHA O ITEM
QUE DESEJA ALTERAR");
                        borda(BORDALARG,'-',RECUOSIMPLES);
                        if(elemento.categoria== VIVENDA ||
elemento.categoria==APARTAMENTO){

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO," 1 - Classe | 2 -
Tipologia | 3 - Área | 4 - Suites");

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO," 5 - Cert. Energ. | 6
- Reg. Predial | 7 - Cert. Habit.| 8 - Valor");

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"9 - Nome | 10 -
NIF | 11 - Endereço | 12 - Obs.");
                                borda(BORDALARG,'-',RECUOSIMPLES);
                                printf(BOLD);

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"13 - SALVAR | 0 -
DESCARTAR");
                                printf(RESET);
                                borda(BORDALARG,'-',RECUOSIMPLES);
                                limiteB=13;
                        }else{

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"1 - Descrição | 2 -
Horário | 3 - Endereço | 4 - Obs.");
                                borda(BORDALARG,'-',RECUOSIMPLES);
                                printf(BOLD);

        bordaVertical(BORDALARG,'|',RECUOSIMPLES,CENTRALIZADO,"5 - SALVAR | 0 -
DESCARTAR");
                                printf(RESET);
                                borda(BORDALARG,'-',RECUOSIMPLES);
                                limiteB=5;
                        }

                        validaSalvaInteiro(&opcao,SEMDECREMENTO,"\n\tEscolha -->
","OPÇÃO INVÁLIDA!","Escolha uma opção do
menu.",MINOPCAOPADRAO,limiteB,INTERVALOFECHADO,DENTRODOINTERVALO);
                        if(limiteB==5)
                                opcao+=14;

                        switch(opcao){
                                case 1:
```

```
validaSalvaInteiro(&elemento.imovel.classe,DECREMENTO,"\n\tInforme a CLASSE do imóvel [1 - ARRENDAMENTO  | 2 - VENDA]: ","CLASSE INVÁLIDA!","Escolha uma CLASSE entre 1 e 2.",MINOPCAOPADRAO,1,INTERVALOFECHADO,DENTRODOINTERVALO);
                                        break;
                                case 2:

validaSalvaInteiro(&elemento.imovel.tipologia,SEMDECREMENTO,"\n\tQuartos: ","QUANTIDADE DE QUARTOS INVÁLIDA!","Escolha uma QUANTIDADE entre 0 e "SMAXQUARTOS,MINOPCAOPADRAO,MAXQUARTOS,INTERVALOFECHADO,DENTRODOINTERVALO);
                                        break;
                                case 3:

validaSalvaInteiro(&elemento.imovel.area,SEMDECREMENTO,"\n\tÁrea do imóvel: ","ÁREA INVÁLIDA!","Escolha uma ÁREA entre 0 e "SMAXAREA,MINOPCAOPADRAO,MAXAREA,INTERVALOFECHADO,DENTRODOINTERVALO);
                                        break;
                                case 4:

validaSalvaInteiro(&elemento.imovel.suites,SEMDECREMENTO,"\n\tTotal de suítes: ","QUANTIDADE DE SUÍTES INVÁLIDA!","Nº de suítes deve ser inferior a tipologia do imóvel.",MINOPCAOPADRAO,elemento.imovel.tipologia,INTERVALOFECHADO,DENTRODOINTERVALO);
                                        break;
                                case 5:
                                        printf("\n\tESCOLHA UM CERTIFICADO ENERGÉTICO\n\t");
                                        for(i=0;i<8;i++)
                                                printf("%hu - %s\t|",i+1,CERTENERG[i]);


validaSalvaInteiro(&elemento.imovel.certEnerg,DECREMENTO,"\n\n\tCertificado Energético: ","CERTIFICADO ENERGÉTICO INVÁLIDO!","Escolha um certificado entre 1 e 8.",MINOPCAOPADRAO,7,INTERVALOFECHADO,DENTRODOINTERVALO);
                                        break;
                                case 6:
                                        lerTexto("\n\tRegistro Predial: ",elemento.imovel.regPredial,TMAXREGPREDIAL);
                                        break;
                                case 7:
                                        lerTexto("\n\tCertificado de Habitabilidade: ",elemento.imovel.certHabit,TMAXCERTHAB);
                                        break;
                                case 8:
                                        do{
                                                /*TRATAMENTO DE ERRO*/
                                                if(statusEscolha==FALHA)
                                                        tratarErro("VALOR INVÁLIDO!","O valor deve ser maior que ZERO.");

                                                printf("\n\tValor: ");
```

```
                                        scanf("%lu",&elemento.imovel.valor);
                                        if(elemento.imovel.valor>0)

                                                statusEscolha=OK;

                                        else
                                                statusEscolha=FALHA;
                                }while(statusEscolha==FALHA);
                                break;
                        case 9:
                                lerTexto("\n\tNome do Proprietário:
",elemento.imovel.proprietario.nome,TMAXNOME);
                                break;
                        case 10:
                                do{
                                        /*TRATAMENTO DE ERRO*/
                                        if(statusEscolha==FALHA)
                                                tratarErro("NIF INVÁLIDO!","O NIF
deve ter exclusivamente 9 digitos.");

                                        lerTexto("\n\tNIF do Proprietário:
",elemento.imovel.proprietario.nif,TMAXNIF);

                                        statusEscolha =
validaNif(elemento.imovel.proprietario.nif);


                                }while(statusEscolha==FALHA);
                                break;
                        case 17:
                        case 11:
                                lerTexto("\n\tRua c/ nº:
",elemento.endereco.rua,TMAXRUA);
                                lerTexto("\n\tLocalidade:
",elemento.endereco.localidade,TMAXRUA);
                                break;
                        case 18:
                        case 12:
                                lerTexto("\n\tObservações:
",elemento.obs,TMAXOBS);
                                break;
                        case 19:
                        case 13:
                                atualizaElemento(&elemento);
                                opcao=13;
                                break;
                        case 15:
                                lerTexto("\n\tDescrição:
",elemento.pInteresse.descricao,TMAXDESCRICAO);
                                break;
                        case 16:
```

```
        validaSalvaInteiro(&elemento.pInteresse.horario.aplica,DECREMENTO,"\n\tPossui
HORÁRIO de funcionamento [1 - NÃO  | 2 - SIM]: ","OPÇÃO INVÁLIDA!","Escolha uma opção
entre 1 ou 2.",MINOPCAOPADRAO,1,INTERVALOFECHADO,DENTRODOINTERVALO);

                                        if(elemento.pInteresse.horario.aplica == APLICAVEL){
                                                validaSalvaHorario("\n\tHORÁRIO de
abertura [HH:MM]:
",&elemento.pInteresse.horario.abertura.hora,&elemento.pInteresse.horario.abertura.min
utos);
                                                validaSalvaHorario("\n\tHORÁRIO de
encerramento [HH:MM]:
",&elemento.pInteresse.horario.encerramento.hora,&elemento.pInteresse.horario.encerra
mento.minutos);
                                        }else{

        elemento.pInteresse.horario.abertura.hora=0;

        elemento.pInteresse.horario.abertura.minutos=0;

        elemento.pInteresse.horario.encerramento.hora=0;

        elemento.pInteresse.horario.encerramento.minutos=0;
                                        }
                                        break;
                                case 14:
                                        opcao=0;
                        }
                }while(!(opcao == 0 || opcao == 13));

                if(opcao == 13)
                        boxSucesso("SUCESSO","Elemento alterado com Sucesso.");
        }
```

3.3.5 - structs

Structs

```
        #ifndef __STRUCTS_H__
        #define __STRUCTS_H__

        #include "config.h"

        typedef struct _hora{
                unsigned short int hora;
                unsigned short int minutos;
        }HORA;
        typedef struct _horario{
                unsigned short int aplica;
                HORA abertura;
```

```
        HORA encerramento;
}HORARIO;
typedef struct _posyx{
        unsigned short int linha;
        unsigned short int coluna;
}POSYX;
typedef struct _data{
        unsigned short int dia;
        unsigned short int mes;
        unsigned short int ano;
}DATA;
typedef struct _proprietario{
        char nome[TMAXNOME+1];
        char nif[TMAXNIF+1];
}PROPRIETARIO;
typedef struct _endereco{
        char rua[TMAXRUA+1];
        char localidade[TMAXLOCALIDADE+1];
}ENDERECO;
typedef struct _negocio{
        unsigned short int status;
        DATA data;
}NEGOCIO;
typedef struct _pinteresse{
        char descricao[TMAXDESCRICAO+1];
        HORARIO horario;
}PINTERESSE;
typedef struct _imovel{
        unsigned short int classe;
        unsigned short int tipologia;
        unsigned short int area;
        unsigned short int suites;
        unsigned short int certEnerg;
        char regPredial[TMAXREGPREDIAL+1];
        char certHabit[TMAXCERTHAB+1];
        unsigned long int valor;
        PROPRIETARIO proprietario;

        NEGOCIO negocio;
}IMOVEL;
typedef struct _elemento{
        unsigned short int id;
        unsigned short int categoria;
        char obs[TMAXOBS+1];
        POSYX inicio;
        POSYX tamanho;
        IMOVEL imovel;
        PINTERESSE pInteresse;
        ENDERECO endereco;
}ELEMENTO;
typedef struct _visita{
        unsigned short int id;
```

```
        DATA data;
}VISITA;
#endif
```

3.3.6 – Config

Config

```
#ifndef __CONFIG_H__
#define __CONFIG_H__

/* BIBLIOTECAS PADRÃO */
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <ctype.h>
#include <string.h>


/* DEFINIÇÃO DE TAMANHO */
#define TMAXLINHA 40
#define TMAXCOLUNA 30
#define TMAXRUA 40
#define MAXCAMPOTEXTO 100
#define TMAXLOCALIDADE 20
#define TMAXDESCRICAO 25
#define TMAXDATA 10
#define TMAXNOME 40
#define TMAXOBS 100
#define TMAXNIF 9
#define TMAXREGPREDIAL 11
#define TMAXCERTHAB 11

#define STMAXLINHA "40."
#define STMAXCOLUNA "30."
#define MAXOPCAOPRINCIAL 3
#define MAXOPCAOADM 4
#define MAXOPCAORL 8
#define MAXOPCAONEGOCIO 3
#define MINOPCAOPADRAO 0
#define SMAXOPCAONEGOCIO "3."
#define SMAXOPCAORL "7."
#define SMAXOPCAOADM "3."
#define SMAXOPCAOPRINCIAL "3."
#define MAXQUARTOS 20
#define SMAXQUARTOS "20."
#define MAXAREA SHRT_MAX
#define SMAXAREA "32.767."
```

```
/* DEFINIÇÃO DE CORES */
        /*Apenas Texto*/
#define RED "\033[1;91m"
#define GREEN "\033[1;92m"
#define RESET "\033[0;0m"
#define BOLD "\x1B[1m"
#define AMARELOT "\x1B[38;2;255;255;0m"
#define ROXOT "\x1B[38;2;186;85;211m"
#define LARANJAT "\x1B[38;2;255;140;0m"
        /*Apenas Fundo*/
#define AMARELO "\x1B[48;2;255;255;0m"
#define BLUE "\x1B[48;2;0;0;205m"
#define MARROM "\x1B[48;2;139;69;19m"
#define CINZA "\x1B[48;2;128;128;128m"
#define AREIA "\x1B[48;2;255;228;181m"
#define VERDEPARQUE "\x1B[48;2;60;179;113m"
#define VERDERURAL "\x1B[48;2;0;128;0m"


/* DEFINIÇÃO DE TELAS */
#define BUFERTELA 120
#define BORDALARG 107
#define BORDAERROR 60
#define MAXTITULOERRO 55
#define BORDAFICHA 80
#define RECUOSIMPLES 1
#define RECUODUPLO 2
#define RECUOERRO 4
#define CENTRALIZADO 1
#define ESQUERDA 2
#define DIREITA 3

/* DEFINIÇÃO DAS CATEGORIAS DA LEGENDA */
#define RUA 0
#define VIVENDA 1
#define APARTAMENTO 2
#define COMERCIO 3
#define HABITACAO 4
#define PARQUE 5
#define MONTANHA 6
#define PRAIA 7
#define TERRENO 8
#define RURAL 9
#define RIO 10
#define GOVERNAMENTAL 11
#define MULTIBANCO 12
#define MONUMENTO 13
#define QUIOSQUE 14
#define PUBLICO 15
```

```
        /* MACROS DE CONTROLE */
        #define DISPONIVEL 1
        #define INDISPONIVEL 0
        #define VENDA 1
        #define ARRENDA 0
        #define APLICAVEL 1
        #define NAOAPLICAVEL 0
        #define OK 1
        #define FALHA 0
        #define SIM 1
        #define NAO 0
        #define DECREMENTO 1
        #define SEMDECREMENTO 0
        #define MATRICULA1 24031
        #define MATRICULA2 24544
        #define COORDENADA 1
        #define TAMANHO 0
        #define VLINHA 0
        #define VCOLUNA 1
        #define NULO 0
        #define MAPAGENERICO 0
        #define VISITASANO 0
        #define INTERVALOABERTO 0
        #define INTERVALOFECHADO 1
        #define DENTRODOINTERVALO 1
        #define FORADOINTERVALO 0
        #define CANCELARNEGOCIO 0
        #define REGISTRARNEGOCIO 1
        #define SLEGENDA 1
        #define NLEGENDA 0

        #endif

        /* ################## GERENCIAMENTO DE DEPENDENCIAS ################## */

        /* DEPENDÊNCIAS EM CRUD.H */
        #ifdef __CRUD_H__
                #include "auxiliares.h"
                #include <errno.h>
        #endif

        /* DEPENDÊNCIAS EM DESIGN.H */
        #ifdef __DESIGN_H__
                #include <stdarg.h>
                #include "auxiliares.h"
                #include "crud.h"
        #endif

        /* DEPENDÊNCIAS EM TESTER.H */
        #ifdef __TESTER_H__
```

```
        #include "auxiliares.h"
        #include "design.h"
#endif

/* DEPENDÊNCIAS EM AUXILIARES.H */
#ifdef __AUXILIARES_H__
        #include "crud.h"
        #include "design.h"
        #include <time.h>
        #include <limits.h>
#endif


/* DEPENDÊNCIAS EM MAIN */
#ifdef __PRINCIPAL__
        #include "auxiliares.h"
        #include "tester.h"
        #include "design.h"
#endif

/* COMPARTILHA STRUCT.H COM TODOS EXCETO STRUCT */
#ifndef __STRUCTS_H__
#include "structs.h"
#endif

/*VARIÁVEIS GLOBAIS*/
#ifndef __PRINCIPAL__
extern const  char CERTENERG[8][3];
extern const  char SIMBOLO[];
extern const char CATEGORIA[][21];
extern char ELEMENTOS_BD[20];
extern char MESES[12][10];
extern char VISITAS_BD[20];
#endif
```

3.3.7 - Main

Main

```
/*@@@@ DIRETIVAS DE INCLUSÃO                                    @@@@*/
#ifndef __PRINCIPAL__
#define __PRINCIPAL__
#endif
#include "config.h"

/*@@@@ VARIÁVEIS GLOBAIS                                         @@@@*/

const  char CERTENERG[8][3]={"A+","A","B","B-","C","D","E","F"};
```

```
	const  char SIMBOLO[]={'+','V','A','C','H','Ø','^','P','T','R','»','G','$','M','Q','@'};
	const char CATEGORIA[][21]={"RUA","VIVENDA","APARTAMENTO","COMÉRCIO","HABITAÇÃO PERMANENTE","PARQUE OU PRAÇA","MONTANHA","PRAIA","TERRENO","ÁREA RURAL","RIO","ÁREA GOVERNAMENTAL","MULTIBANCO","MONUMENTO","QUIOSQUE","ESPAÇO PÚBLICO"};
	char ELEMENTOS_BD[20]="elementos.dat";
	char VISITAS_BD[20]={"visitas.dat"};
	char MESES[12][10]={"JANEIRO","FEVEREIRO","MARÇO","ABRIL","MAIO","JUNHO","JULHO","AGOSTO","SETEMBRO","OUTUBRO","NOVEMBRO","DEZEMBRO"};

	int main(int argc, char *argv[]) {
		ELEMENTO elemento;
		unsigned short int mapa[TMAXLINHA][TMAXCOLUNA];
		unsigned short int coordenadas[2], tamanho[2];
		unsigned short int opcao, opcaoSecundaria;

		setlocale(LC_ALL, "");
		system("color");
		do{
			limpaTela();
			inicializaMapa(mapa);

			opcao=menuPrincipal();
			switch(opcao){
				case 1:
					limpaTela();
					desenhaMapa(mapa,MAPAGENERICO);

					lerPosix(coordenadas,tamanho,COORDENADA);

	elemento=consultaElemento(mapa[coordenadas[0]][coordenadas[1]]);
					limpaTela();
					cabecalho(BORDALARG,'*',RECUOSIMPLES, "CONSULTA ELEMENTO URBANO",2,"Início","Consulta elemento Urbano");

	desenhaMapa(mapa,mapa[coordenadas[0]][coordenadas[1]]);
					imprimeElemento(&elemento);
					continuar("\n\tPressione qualquer tecla para voltar ao MENU PRINCIPAL. . .");
					break;
				case 2:
					limpaTela();
					opcaoSecundaria=menuNegocio();
					switch(opcaoSecundaria){
						case 1: /*registra um negócio*/
							limpaTela();

	desenhaMapa(mapa,MAPAGENERICO);
```

```
lerPosix(coordenadas,tamanho,COORDENADA);


desenhaMapa(mapa,mapa[coordenadas[0]][coordenadas[1]]);


registraNegocio(mapa[coordenadas[0]][coordenadas[1]],REGISTRARNEGOCIO);

                                        break;
                                case 2: /*cancela um negócio*/
                                        limpaTela();

desenhaMapa(mapa,MAPAGENERICO);

lerPosix(coordenadas,tamanho,COORDENADA);


desenhaMapa(mapa,mapa[coordenadas[0]][coordenadas[1]]);

registraNegocio(mapa[coordenadas[0]][coordenadas[1]],CANCELARNEGOCIO);

                                        break;
                                case 3: /*registra uma visita*/
                                        limpaTela();

desenhaMapa(mapa,MAPAGENERICO);

lerPosix(coordenadas,tamanho,COORDENADA);


desenhaMapa(mapa,mapa[coordenadas[0]][coordenadas[1]]);

cadastraVisita(mapa[coordenadas[0]][coordenadas[1]]);

                                        break;
                        }
                        break;
                case 3:
                        limpaTela();
                        opcaoSecundaria=menuRelatorio();
                        switch(opcaoSecundaria){
                                case 1:
                                        limpaTela();
                                        imprimeRLvendidosMes(VENDA);

                                        break;
                                case 2:
                                        limpaTela();
                                        imprimeRLvendidosMes(ARRENDA);

                                        break;
```

```
                                        case 3:
                                                limpaTela();
                                                imprimeDistPreco();

                                                break;
                                        case 4:
                                                limpaTela();
                                                imprimeDistTipologia();

                                                break;
                                        case 5:
                                                limpaTela();
                                                imprimeDistArea();

                                                break;
                                        case 6:
                                                limpaTela();
                                                imprimeDistEnerg();

                                                break;
                                        case 7:
                                                limpaTela();
                                                imprimeRLinteresse(VENDA);

                                                break;
                                        case 8:
                                                limpaTela();
                                                imprimeRLinteresse(ARRENDA);

                                                break;

                                }
                                continuar("\n\tPressione qualquer tecla para voltar
ao MENU PRINCIPAL. . .");
                                break;

                        case 24031:
                        case 24544:
                                limpaTela();
                                opcaoSecundaria=menuAdm();
                                switch(opcaoSecundaria){
                                        case 1: /*Insere um novo elemento*/
                                                limpaTela();

        desenhaMapa(mapa,MAPAGENERICO);

        if(lerCoordenadaTamanho(mapa,coordenadas,tamanho)==FALHA)
                                                                break;
                                                limpaTela();

        criaElemento(mapa[coordenadas[0]][coordenadas[1]],coordenadas,tamanho);
```

```
                                                    break;
                                            case 2: /*Apaga um elemento*/
                                                    limpaTela();

        desenhaMapa(mapa,MAPAGENERICO);

        lerPosix(coordenadas,tamanho,COORDENADA);

        if(apagaElemento(mapa[coordenadas[0]][coordenadas[1]]))

        boxSucesso("SUCESSO","Elemento eliminado com Sucesso.");
                                                    else
                                                            tratarErro("EXCLUSÃO
FALHOU","Não foi possível eliminar o EL.");
                                                    break;
                                            case 3: /*Atualiza um elemento*/
                                                    limpaTela();

        desenhaMapa(mapa,MAPAGENERICO);

        lerPosix(coordenadas,tamanho,COORDENADA);

        telaAtualizaElemento(mapa[coordenadas[0]][coordenadas[1]]);

                                                    break;
                                            case 4:/*Testa o sistema*/
                                                    limpaTela();
                                                    tester();
                                                    continuar("\n\tPressione     qualquer
tecla para voltar ao MENU PRINCIPAL. . .");
                                                    break;
                                            }

                                            break;
                                    }

                }while(opcao!=0);

                return 0;
        }
```

4 – Testes

Desenvolvemos o nosso sistema com influência da metodologia orientada a testes, tentando, sempre que possível, trabalhar com alta coesão, baixo acoplamento e gerenciando a injeção de dependências, para que fosse possível implementar rotinas de testes no maior número de funções e procedimentos possíveis.

Para tal, não só implementamos um sistema de testes proprietário, como integramos este sistema de testes no sistema final, para garantir a estabilidade e a sensação de confiança uma vez que o sistema pode ser submetido a teste a qualquer momento sem que isto interfira ou interrompa o seu funcionamento.

Tal módulo garante ainda melhores condições para manutenção e modificação do sistema, uma vez que é possível verificar se o sistema continua estável de forma rápida e prática após uma edição.

Quanto a lógica do programa, a mesma foi demonstrada na apresentação do trabalho ao docente.

5 – Dificuldades

Uma das primeiras dificuldades que se mostrou presente foi a interpretação do trabalho. Perceber o problema e as suas conexões objetivas, ou seja, o que de fato deveríamos entregar foi deveras o primeiro impasse desta jornada.

Entretanto conseguimos superar esta agrura com uma modelagem do problema, que se deu através de digressões de abstrações, conjecturas e, em modo mais prático, com alguns protótipos de código para consubstanciar a ideia primordial.

Já na fase de arquitetar a solução nosso revés foi em efetivamente decidir entre operar com os elementos urbanos diretamente em uma matriz bidimensional ou lidar com eles em uma lista, com uma possível tabela hash, entretanto não visualizamos vantagens, em termos de otimização de tempo e processamento, que justificassem a inerente complexidade que seria agregada ao código. Concluímos por um meio termo, operar com eles em uma lista simples, implementada em arquivos binários.

Outro ponto, ainda na arquitetura da solução, que se tornou motivo de debate foi a existência ou não da possibilidade de executar-se operações administrativa, como criar, editar e deletar elementos urbanos. Por fim optamos por incluir estas opções, por considerarmos uma circunstância mais realista, entretanto optamos em manter este menu oculto, para causar uma experiência mais fluida e objetiva ao usuário. O menu administrativo acessa-se por uma senha inserida, pelo administrador, na leitura da opção do menu inicial.

6 – Sugestões Futuras

Implementar um módulo de configurações dinâmico, armazenado em arquivo binário, onde se pode alterar o arquivo que se irá usar para o mundo, ou seja, os elementos. Desta forma se poderia operar com um mapa a cada operação ou ainda a substituição de mapas em tempo de execução. Para facilitar esta implementação no futuro trabalhamos com o endereço do arquivo em uma variável global, de forma que a mesma possa ser manipulada de forma simples e por qualquer módulo, sem a necessidade e grandes alterações no código.

Fazer com que um imóvel vendido passe a habitação permanente, ocultando assim informações mais sensíveis, como o nome e NIF do proprietário.

Implementar uma gestão no arquivo de visitas para eliminar visitas de imóveis demolidos ou passados um determinado período.

Solucionar a questão da acentuação nos arquivos binários.

7 – Conclusões

O presente trabalho foi uma oportunidade salutar de consolidar os ensinamentos percebidos em sala de aula. Foi significativa a oportunidade e o desafio de implementar a base de dados em ficheiros, contar com a possibilidade de nos testarmos em uma situação extrema de entrega de trabalho final com uma tecnologia que não faz parte do conteúdo foi estimulantemente desafiador e empolgante ao ponto de nos fazer perceber que era possível e implementarmos técnicas que não faziam parte do currículo deste semestre como funções com argumentos variáveis, macros condicionais, divisão do projeto em múltiplos arquivos, entre outros.

Podemos dizer que aprendemos muito mais do que uma linguagem de programação, o C no nosso caso. As aulas e o trabalho nos permitiram compreender a programação além do código e do algoritmo, e sim sob a perspectiva da engenharia. Lidamos pela primeira vez com o desafio assustador de administrar um projeto, lidar com as “partes”, compreender o que são requisitos e como levantá-los.

Até o erro nos fez aprender, como lidar com o erro e a gestão da correção. Aprendemos a arriscar e mais, a gerir o risco.

Não podemos concluir este trabalho sem deixar o nosso mais sincero e profundo agradecimento ao nosso tutor, o dr. Salvador Lima, que nos ensinou que ser um programador é muito mais do que escrever código, ser um engenheiro de software é ser alguém que resolve problemas que outros não consegue, é ser alguém que ajuda pessoas, é ser alguém que transforma o mundo a sua volta e que para ser este alguém é preciso mais do que conhecer uma linguagem, é preciso mudar a forma de pensar.

8 – Bibliografia

Pressman, R., & Maxim, B. (2019). *Engenharia de Software - Uma Abordagem Profissional* (8th ed.). Porto Alegre: MC Graw Hill.

Guerreiro, S. (2015). *Introdução a Engenharia de Software* (1st ed.). Lisboa: FCA.

Damas, L. (2015). *Linguagem C* (24th ed.). Lisboa: FCA.

Adrego da Rocha, A. (2014). *Estrutura de Dados e Algoritmos em C* (3rd ed.). Lisboa: FCA.

Liberland e-government tools. (2019). Retrieved 14 November 2019, from http://e-liberland.org/

Free Republic of Liberland. (2019). Retrieved 14 November 2019, from https://liberland.org/