




**CONDOMIX**

# TEMA 6 – GESTÃO DE CONDOMÍNIOS



Trabalho realizado no âmbito da cadeira de  
Algoritmos e Estrutura de Dados do 2ºano do 1º  
Semestre do MIEIC-FEUP

Realizado por:

Beatriz Mendes 201806551

Daniel Silva 201909935

Paulo Ribeiro 201806505



# Descrição do Problema e Solução



# PROBLEMA

## PARTE 1

Criar um programa capaz de gerir um condomínio com habitações de diferentes tipos, informação dos condóminos e ainda serviços disponíveis a cada um destes condóminos

## PARTE 2

Para além do problema da parte 1, é necessário guardar os condomínios administrados pela CondomiX numa árvore binária de pesquisa e guardar os antigos condóminos numa tabela de dispersão. Deve ser possível guardar a informação de uma rede de transportes públicos de um condomínio numa fila de prioridade.



## SOLUÇÃO

- Utilizar a aplicação desenvolvida na Parte 1 e utilizar estruturas de dados não lineares (BST's, filas de prioridade e tabelas de dispersão) de forma a ser possível a organização da informação pelos diferentes condomínios de maneira eficiente
- operações básicas CRUD (Create, Read, Update, Delete)
- listagens várias: totais ou parciais com critérios a definir pelo utilizador (não aplicável a filas de prioridade)





## ALGORITMOS RELEVANTES (Parte 2)

- Priority Queue (STL)
  - Algoritmos de Pesquisa em estrutura de dados não lineares (implementados por nós)
  - Algoritmos de Escrita e Leitura de Ficheiros
- Etc.

# Operadores Associados às Estruturas de dados não lineares

- Operator==(BST, HT)
- Operator< (BST, PQ)
- Operator= (HT)
- Operator() (PQ, HT)

(HT – Hash Table; BST – Binary Search Tree; PQ – priority queue)

```
bool Condominio::operator < (const Condominio &c1) const {  
    if (habitacoes.size() == c1.habitacoes.size())  
        return (numeroVivendas() < c1.numeroVivendas());  
    else  
        return (habitacoes.size() < c1.habitacoes.size());  
}
```

*“(...) sendo a ordenação efetuada por número de habitações e, em caso de empate, por número de vivendas.”*

```
bool AntigoCondomino::operator == (const  
AntigoCondomino &c1) const {  
    return nif == c1.nif;  
}
```

# Estrutura de Ficheiros (Ex: Habitacoes.txt)

```
Condominio do Sul
Lisboa, Portugal
Rua Apartamento de Teste
Teste
4999-000
456
Apartamento
t3
2
-----
Rua 25 de Abril
Lisboa
1000-004
20
Apartamento
T2
3
-----
Rua de Alvalade
Lisboa
1906-777
300
Apartamento
t5
4
-----
Condominio do Norte
Porto, Portugal
Rua de Agosto
Aveiro
3800-003
150
Vivenda
1000
1
-----
```

Habitacoes.txt

```
void CondomiX::Ler_Habitacoes(const string &ficheiro)
{
    ifstream ler;
    string designacao, localizacao, end ,cid ,cp,
    tipo_habitacao, separacao, temp;
    float area_habitacional;
    try{
        ler.open(ficheiro);
    }
    catch (Erro_Ler(&ficheiro)){
        cout << "Erro a Ler Habitacoes\n";
        exit(1);
    }
    condominios.makeEmpty();
    do{
        getline(ler, designacao);
        getline(ler, localizacao);
        Condominio cond(designacao, localizacao);
        while(end != "-----")
        {
            getline(ler, temp);
            if (temp == "-----")
            {
                end = temp;
            }
            else
            {
                end = temp;
                getline(ler, cid);
                getline(ler, cp);
                Morada *morada = new Morada(end, cid, cp);
                ler >> area_habitacional;
                ler.ignore(1000, '\n');
                getline(ler, tipo_habitacao);
                if (tipo_habitacao == "Vivenda") {
                    (. . .)
                }
                else {
                    (. . .)
                }
            }
        }
        adicionaCondominio(cond);
        end = "";
    } while (!ler.eof());
    ler.close();
}
```

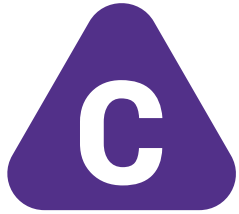
Leitura de Habitacoes.txt

```
void CondomiX::Escrever_Habitacoes(const string &ficheiro)
{
    ofstream escrever;
    bool inicio = true;
    BSTIterIn<Condominio> ita(condominios);
    vector<Habitacao *>::const_iterator it;
    try{
        escrever.open(ficheiro);
    }
    catch (Erro_Escrita(&ficheiro)){
        cout << "Erro a Escrever Habitacoes\n";
        exit(1);
    }
    while (!ita.isAtEnd()) {
        vector<Habitacao *> habitacoes =
        ita.retrieve().getHabitacoes();
        if (inicio) {
            escrever << ita.retrieve().getDesignacao() << endl
            << ita.retrieve().getLocalizacao() << endl;
            inicio = false;
        }
        else
            escrever << endl << ita.retrieve().getDesignacao()
            << endl << ita.retrieve().getLocalizacao() << endl;
        for (it = habitacoes.begin(); it != habitacoes.end();
        it++) {
            if (it == habitacoes.begin())
                (*it)->escrever_habitacao(escrever);
            else {
                escrever << endl;
                (*it)->escrever_habitacao(escrever);
            }
        }
        if(habitacoes.size() != 0)
            escrever << endl << "-----";
        else
            escrever << "-----";
        ita.advance();
    }
    escrever.close();
}
```

Escrita em Habitacoes.txt



# Funcionalidades Implementadas



Create



Read



Update



Delete

```
void Ler_Servicos(const string &ficheiro);
void Ler_Prestadores(const string &ficheiro);
void Ler_Servicos_Efetutados(const string &ficheiro);
void Ler_Habitacoes(const string &ficheiro);
void Ler_Condominos(const string &ficheiro);
void Ler_Antigos_Condominos(const string &ficheiro);
void Ler_Transportes(const string &ficheiro);
void Escrever_Servicos(const string &ficheiro);
void Escrever_Prestadores(const string &ficheiro);
void Escrever_Servicos_Efetutados(const string &ficheiro);
void Escrever_Habitacoes(const string &ficheiro);
void Escrever_Condominos(const string &ficheiro);
void Escrever_Antigos_Condominos(const string &ficheiro);
void Escrever_Transportes(const string &ficheiro);
```

```
void Adicionar_Habitacao(CondomiX &gestao);
void Editar_Habitacao(CondomiX &gestao);
void Eliminar_Habitacao(CondomiX &gestao);
void Adicionar_Condominio(CondomiX &gestao);
void Editar_Condominio(CondomiX &gestao);
void Eliminar_Condominio(CondomiX &gestao);
void Adicionar_Condomino(CondomiX &gestao);
void Editar_Condomino(CondomiX &gestao);
void Eliminar_Condomino(CondomiX &gestao);
void Adquirir_Habitacao(CondomiX &gestao);
void Remover_Habitacao(CondomiX &gestao);
void Adicionar_Servico(Condominio &condominio, CondomiX &gestao);
void Editar_Servico(Condominio &condominio, CondomiX &gestao);
void Eliminar_Servico(Condominio &condominio, CondomiX &gestao);
void Requisitar_Servico(CondomiX &gestao);
void Terminar_Servico(CondomiX &gestao);
void Adicionar_Transporte(CondomiX &gestao);
void Editar_Transporte(CondomiX &gestao);
void Eliminar_Transporte(CondomiX &gestao);
```

Completas

## Listagem Total

```
void Visualizar_Todas_Habitacoes(CondomiX &gestao);
void Visualizar_Todos_Condominos (CondomiX &gestao);
void Visualizar_Todos_Antigos_Condominos (CondomiX &gestao);
void Visualizar_Servicos(Condominio &condominio, CondomiX &gestao);
void Visualizar_Prestadores(Condominio &condominio, CondomiX &gestao);
void Visualizar_Transportes(CondomiX &gestao);
void Visualizar_Todos_Condominios(CondomiX &gestao);
```

[illegible]

```

string CondominiX::escreveTotalAntigosCondominios() const {
    stringstream oss;
    BSTIterIn<Condominio> ita(condominios);
    while (!ita.isAtEnd()) {
        HashTabAntigoClienteRecord ant_cond =
        ita.retrieve().getAntigosCondominios();
        HashTabAntigoClienteRecord::const_iterator it =
        ant_cond.begin();
        oss << "Condominio: " <<
        ita.retrieve().getDesignacao() << " - " (...);
        while (it != ant_cond.end()) {
            oss << (*it).getAntigoCondominoPointer()-
            >info() << endl;
            it++;
        }
        ita.advance();
    }
    return oss.str();
}
}

```

# Funcionalidade Destaque

*“Na informação relativa a um antigo condómino, deve constar o período em que este esteve no condomínio.”*

```
Condomino::Condomino(const string &nm, const string &ni) {  
    nome = nm;  
    nif = ni;  
    time_t now = time(NULL);  
    dataEntrada = *localtime(&now);  
}
```

Exemplo em “Remover\_Habitacoes”

(...)

```
cout << (*itc)->getNome() << " sem Habitacoes. Novo estatuto:  
Antigo Condomino\n\n";  
time_t now = time(NULL);  
struct tm dataEntrada = (*itc)->getDataEntrada();  
double seconds = difftime(now, mktime(&dataEntrada));  
int periodo = (int) (seconds/30000);  
AntigoCondomino * antigoCondomino = new AntigoCondomino((*itc)-  
>getNome(), (*itc)->getNif(), periodo);
```

Regista-se a data de entrada de um condómino num condomínio. Quando esse condómino não for proprietário de nenhuma habitação, compara-se a data de entrada com a data atual (do Sistema Operativo) e calculamos o período em que esteve no condomínio



# Principais Dificuldades

Utilização das Filas de Prioridade e dos Vetores  
para eliminar um elemento

```
void Eliminar_Transporte(CondomiX &gestao)
{
    (...)
    cout << "\n ID do condominio onde se vai eliminar um transporte: ";
    do{
        cin >> id;
        cin.ignore(1000, '\n');
    }while (id < 1 || id > gestao.numeroCondominios());
    BSTItrIn<Condominio> ita(condominios);
    while (!ita.isAtEnd()) {
        if(contador == id)
        {
            Condominio cond = ita.retrieve();
            priority_queue<Transporte *, vector<Transporte *>, TransportesCmp>
transportes = cond.getTransportes();
            vector<Transporte *> aux;

            while (!transportes.empty()) {
                aux.push_back(transportes.top());
                transportes.pop();
            }
            vector<Transporte *>::const_reverse_iterator it;
            vector<Transporte *>::const_iterator itt;
            for (it = aux.rbegin(); it != aux.rend(); it++) {
                transportes.push(*it);
            }
        }
    }
}
```

```
    cout << endl;
    cond.Mostrar_Transportes_Edicao();
    cout << "\n Transporte a Eliminar: ";
    do{
        cin.clear();
        cin >> id_t;
        cin.ignore(1000, '\n');
    }while (id_t < 1 || id_t > cond.numeroTransportes());
    for (itt = aux.begin(); itt != aux.end(); itt++) {
        if (contador_t == aux.size() - id_t + 1) {
            aux.erase(itt);
            break;
        }
        contador_t++;
    }
    vector<Transporte *> novo_aux;
    for (it = aux.rbegin(); it != aux.rend(); it++) {
        novo_aux.push_back(*it);
    }
    cond.setTransportes(novo_aux);
    (...)
    cout << "\n\nTransporte Eliminado!\n\n";
    break;
}
contador++;
ita.advance();
}
```

# Exemplos de Execução

## Exemplo1: Visualização de Condomínios

Visualizacao de Todos os Condominios

Condominio: Condominio Centro  
Localizacao: Coimbra  
Numero de Habitacoes: 2  
Numero de Vivendas: 1  
Numero de Apartamentos: 1  
Numero de Condominos: 1  
Numero de Antigos Condominos: 1  
Numero de Transportes: 0

-----

Condominio: Condominio Espanha  
Localizacao: Madrid  
Numero de Habitacoes: 2  
Numero de Vivendas: 2  
Numero de Apartamentos: 0  
Numero de Condominos: 2  
Numero de Antigos Condominos: 0  
Numero de Transportes: 0

-----

Condominio: Condominio do Sul  
Localizacao: Lisboa, Portugal  
Numero de Habitacoes: 3  
Numero de Vivendas: 0  
Numero de Apartamentos: 3  
Numero de Condominos: 1  
Numero de Antigos Condominos: 2  
Numero de Transportes: 2

-----

Condominio: Condominio do Norte  
Localizacao: Porto, Portugal  
Numero de Habitacoes: 5  
Numero de Vivendas: 4  
Numero de Apartamentos: 1  
Numero de Condominos: 4  
Numero de Antigos Condominos: 4  
Numero de Transportes: 3

-----

Sair do Programa? (S/N)

R:

# Exemplos de Execução

## Exemplo2: Visualizar os Antigos Condomínios

```
Visualizacao de Todos os Antigos Condomínios

Condomínio: Condomínio Centro - Coimbra (2 habitacoes, 1 condomínios)

Nome: apagar
NIF: 657847362
Período: 2 meses

Condomínio: Condomínio Espanha - Madrid (2 habitacoes, 2 condomínios)

Condomínio: Condomínio do Sul - Lisboa, Portugal (3 habitacoes, 1 condomínios)

Nome: Marco Horacio
NIF: 330033003
Período: 1946 meses

Nome: Raul Salgado
NIF: 101110111
Período: 24 meses

Condomínio: Condomínio do Norte - Porto, Portugal (5 habitacoes, 4 condomínios)

Nome: Condomino para Testar
NIF: 123654789
Período: 16 meses

Nome: Segundo Antigo Condomino
NIF: 757575757
Período: 2 meses

Nome: Primeiro Antigo Condomino
NIF: 191919191
Período: 12 meses

Nome: Fernando Madureira
NIF: 198319831
Período: 2 meses

Sair do Programa? (S/N)

R:
```



# Exemplos de Execução

## Exemplo2: Visualizar os Transportes

```
Visualizacao de Todos os Transportes

Condominio: Condominio Centro - Coimbra (2 habitacoes, 1 condominos)

Condominio: Condominio Espanha - Madrid (2 habitacoes, 2 condominos)

Condominio: Condominio do Sul - Lisboa, Portugal (3 habitacoes, 1 condominos)

Tipo: Comboio
Destino: Sao Joao
Paragem mais Proxima: Ermesinde
Distancia: 10

Tipo: Autocarro
Destino: Cais do Sodre
Paragem mais Proxima: Bairro Alto
Distancia: 100

Condominio: Condominio do Norte - Porto, Portugal (5 habitacoes, 4 condominos)

Tipo: Autocarro
Destino: Trindade
Paragem mais Proxima: Camara Municipal
Distancia: 20

Tipo: Comboio
Destino: Sao Joao
Paragem mais Proxima: Maia
Distancia: 40

Tipo: Metro
Destino: Povia de Varzim
Paragem mais Proxima: IPO
Distancia: 200

Sair do Programa? (S/N)

R:
```