PRÁTICA 5 – Trabalho 2 Exercício 5: Busca binária e ordenação

Nome: Davi Gabriel Domingues

Número USP: 15447497

O objetivo desta atividade é o de ajudar uma prefeitura de uma cidade hipotética a fazer uma pesquisa sobre os domicílios na área urbana. Em cada domicilio, são coletados dados sobre renda média mensal, número total de ocupantes, endereço e número de ocupantes em idade escolar.

Para tanto, foi – se solicitado um programa, em C++, que contivesse um menu que permitisse:

(a) O número de domicílios que serão cadastrados;

(b) A leitura dos dados do domicílio;

(c) A ordenação dos dados pelo método de inserção direta, de acordo com a ordem alfabética devida dos endereços;

(d) A listagem dos endereços no programa e a impressão das informações do respectivo domicílio, cujo endereço foi digitado pelo usuário. Para tal objetivo, foi – se requisitado a utilização da busca binária (usei a versão simplificada, não a rápida);

Dessa forma, o programa a seguir foi desenvolvido, a fim de ser tomado como resposta para a situação-problema proposta na Prática:

```cpp
#include <iostream>
#include <limits>
#include <string> //só basta colocar nome1 < nome2, que a comparação alfabética é automática
#include <cmath>

using namespace std;

struct Domicilio{
    float rendaMensal;
    int totalOcupantes;
    string endereco;
    int ocupantesIdadeEscolar;
};

bool preencherInformacaoDomicilio(Domicilio& domicilio) {
    cout << endl <<"Renda mensal do domicilio: ";
    cin >> domicilio.rendaMensal;
    if (domicilio.rendaMensal <= 0) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        cin.ignore(); // Ignora a linha
        return false;
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Total de ocupantes do domicilio: ";
    cin >> domicilio.totalOcupantes;
    if (domicilio.totalOcupantes <= 0) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return false;
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Endereco do domicilio: ";
    getline(cin, domicilio.endereco);
    if (domicilio.endereco.empty()) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        return false;
    }

    cout << "Ocupantes em idade escolar do domicilio: ";
    cin >> domicilio.ocupantesIdadeEscolar;
    if (domicilio.ocupantesIdadeEscolar < 0 || domicilio.ocupantesIdadeEscolar >= domicilio.totalOcupantes) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return false;
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    return true;
}

void ordenacaoInformacoesDomicilioDeclarados(Domicilio domicilio[], int tamanho){ //ordenação por inserção direta
    for (int i = 2; i <= tamanho; i++){
        Domicilio x = domicilio[i];
        int j = i - 1;

        for (int j = i - 1; j >= 0 && x.endereco < domicilio[j].endereco; j--)
            domicilio[j + 1] = domicilio[j];

        domicilio[j + 1] = x;
    }
}

void encontrarDadosDomicilio(Domicilio domicilio[], int i, string endereco_desejado){ // busca binaria, i = tamanho
    int a = 1, b = i - 1, m;
    bool encontrado = false;

    while ((a <= b) && (!encontrado)){
        m = floor((a + b)/2);
        if (domicilio[m].endereco == endereco_desejado){
            cout<<"Endereco localizado, dados encontrados!"<<endl;
```

```
            cout<<"Renda mensal: "<<domicilio[m].rendaMensal<<endl;
            cout<<"Total de ocupantes: "<<domicilio[m].totalOcupantes<<endl;
            cout<<"Ocupantes em idade escolar: "<<domicilio[m].ocupantesIdadeEscolar<<endl;
            encontrado = true;
        }

        else{
            if (domicilio[m].endereco > endereco_desejado)
                b = m - 1;
            else
                a = m + 1;
        }
    }

    if (!encontrado) cout<<"Endereco nao localizado, dados inexistentes!"<<endl;
}

int main(){
    int opcao = 0, i = 0, total_domicilios;
    string endereco_desejado;

    cout<<"Informe o total de domicilios: ";
    cin>>total_domicilios;
    cout<<endl;

    Domicilio* domicilio = new Domicilio[total_domicilios + 1]; //vetor que comporta o sentinela

    do{
        cout<<"Informe o que deseja fazer:"<<endl;
        cout<<"(1) Digitar os dados de um domicilio"<<endl;
        cout<<"(2) Buscar os dados de um domicilio, a partir de um endereco"<<endl;
        cout<<"(3) Sair"<<endl;
        cin>>opcao;

        switch(opcao){
            case 1:
                if (i < total_domicilios){
                    if (preencherInformacaoDomicilio(domicilio[i])) i++;
                }

                else cout<<"Numero maximo de domicilios alcancado, impossivel adicionar informacoes!"<<endl;
                break;

            case 2:
                cout<<"Informe o endereco desejado para saber os dados associados: ";
                getline(cin, endereco_desejado);

                ordenacaoInformacoesDomicilioDeclarados(domicilio, i);

                  encontrarDadosDomicilio(domicilio, i, endereco_desejado);
                  break;

              case 3:
                  cout<<endl<<"Saindo..."<<endl;
                  break;

              default:
                  cout<<"Opcao invalida, tente novamente..."<<endl;
          }

          cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpa o buffer antes de um novo loop
    } while(opcao != 3);

    delete[] domicilio;

return 0;
}
```

Código utilizado:

```
#include <iostream>
#include <limits>
#include <string> //só basta colocar nome1 < nome2, que a comparação alfabética é automática
#include <cmath>

using namespace std;

struct Domicilio{
  float rendaMensal;
  int totalOcupantes;
  string endereco;
  int ocupantesIdadeEscolar;
};

bool preencherInformacaoDomicilio(Domicilio& domicilio) {
  cout << endl <<"Renda mensal do domicilio: ";
  cin >> domicilio.rendaMensal;
  if (domicilio.rendaMensal <= 0) {
    cout << "Dado invalido! Insercao interrompida..." << endl;
    cin.ignore(); // Ignora a linha
    return false;
  }
  cin.ignore(numeric_limits<streamsize>::max(), '\n');

  cout << "Total de ocupantes do domicilio: ";
  cin >> domicilio.totalOcupantes;
  if (domicilio.totalOcupantes <= 0) {
    cout << "Dado invalido! Insercao interrompida..." << endl;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

```
        return false;
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Endereco do domicilio: ";
    getline(cin, domicilio.endereco);
    if (domicilio.endereco.empty()) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        return false;
    }

    cout << "Ocupantes em idade escolar do domicilio: ";
    cin >> domicilio.ocupantesIdadeEscolar;
    if (domicilio.ocupantesIdadeEscolar < 0 || domicilio.ocupantesIdadeEscolar >=
domicilio.totalOcupantes) {
        cout << "Dado invalido! Insercao interrompida..." << endl;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return false;
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    return true;
}

void ordenacaoInformacoesDomicilioDeclarados(Domicilio domicilio[], int tamanho){
//ordenação por inserção direta
    for (int i = 2; i <= tamanho; i++){
        Domicilio x = domicilio[i];
        domicilio[0] = x;
        int j = i;

        while (j >= 0 && x.endereco < domicilio[j - 1].endereco){
```

```
        domicilio[j] = domicilio[j - 1];
        j--;
      }
      domicilio[j] = x;
    }
}

void encontrarDadosDomicilio(Domicilio domicilio[], int i, string endereco_desejado){ // busca binaria, i = tamanho
    int a = 1, b = i - 1, m;
    bool encontrado = false;

    while ((a <= b) && (!encontrado)){
      m = floor((a + b)/2);
      if (domicilio[m].endereco == endereco_desejado){
        cout<<"Endereco localizado, dados encontrados!"<<endl;
        cout<<"Renda mensal: "<<domicilio[m].rendaMensal<<endl;
        cout<<"Total de ocupantes: "<<domicilio[m].totalOcupantes<<endl;
        cout<<"Ocupantes em idade escolar: "<<domicilio[m].ocupantesIdadeEscolar<<endl;
        encontrado = true;
      }

      else{
        if (domicilio[m].endereco > endereco_desejado)
          b = m - 1;
        else
          a = m + 1;
      }
    }

    if (!encontrado) cout<<"Endereco nao localizado, dados inexistentes!"<<endl;
```

```
}

int main(){
   int opcao = 0, i = 0, total_domicilios;
   string endereco_desejado;

   cout<<"Informe o total de domicilios: ";
   cin>>total_domicilios;
   cout<<endl;

   Domicilio* domicilio = new Domicilio[total_domicilios + 1]; //vetor que comporta o
sentinela

   do{
      cout<<"Informe o que deseja fazer:"<<endl;
      cout<<"(1) Digitar os dados de um domicilio"<<endl;
      cout<<"(2) Buscar os dados de um domicilio, a partir de um endereco"<<endl;
      cout<<"(3) Sair"<<endl;
      cin>>opcao;

      switch(opcao){
         case 1:
            if (i < total_domicilios){
               if (preencherInformacaoDomicilio(domicilio[i])) i++;
            }

            else cout<<"Numero maximo de domicilios alcancado, impossivel adicionar
informacoes!"<<endl;
            break;

         case 2:
            cout<<"Informe o endereco desejado para saber os dados associados: ";
```

```
            getline(cin, endereco_desejado);

            ordenacaoInformacoesDomicilioDeclarados(domicilio, i);
            encontrarDadosDomicilio(domicilio, i, endereco_desejado);
            break;

          case 3:
            cout<<endl<<"Saindo..."<<endl;
            break;

          default:
            cout<<"Opcao invalida, tente novamente..."<<endl;
        }

        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpa o buffer antes de um
novo loop
    } while(opcao != 3);

    delete[] domicilio;

return 0;
}
```