Nome: **Estevão Luiz Duarte da Costa** Matrícula: **CJ3002853**

PROGRAMA AGENDA

ORGANIZAÇÃO HIERÁRQUICA DO PROGRAMA

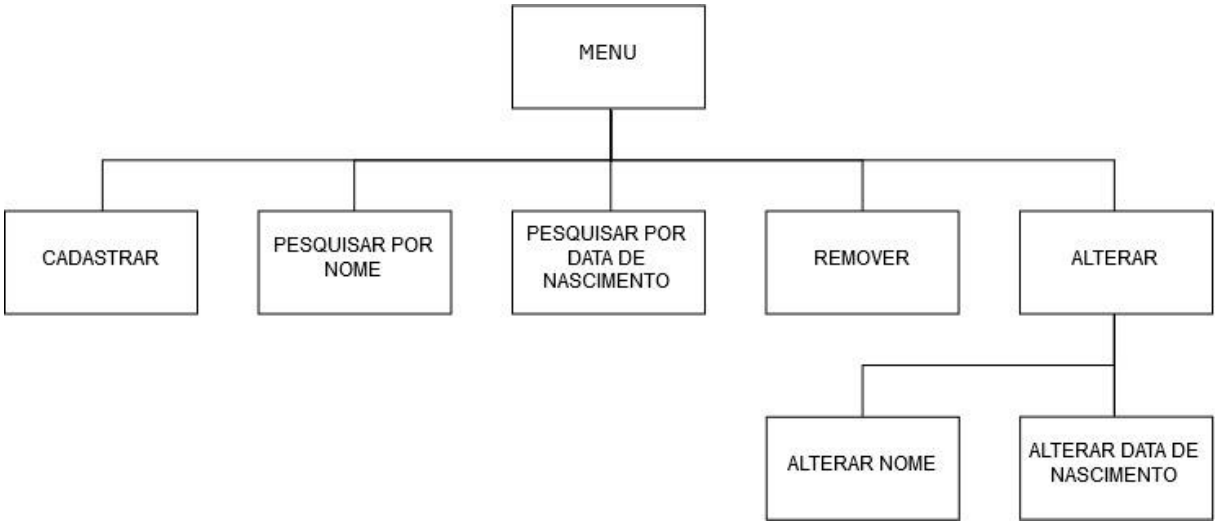
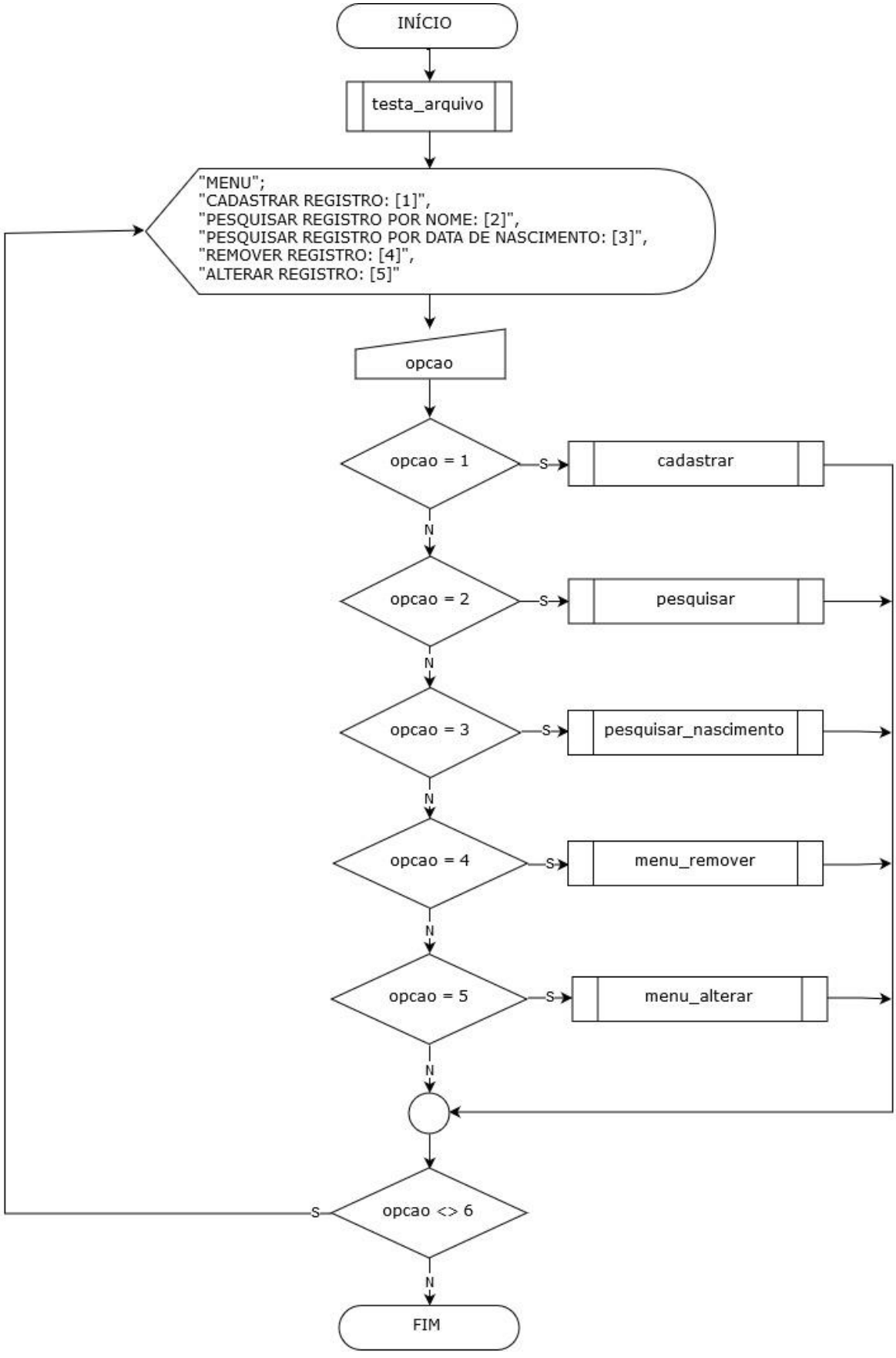
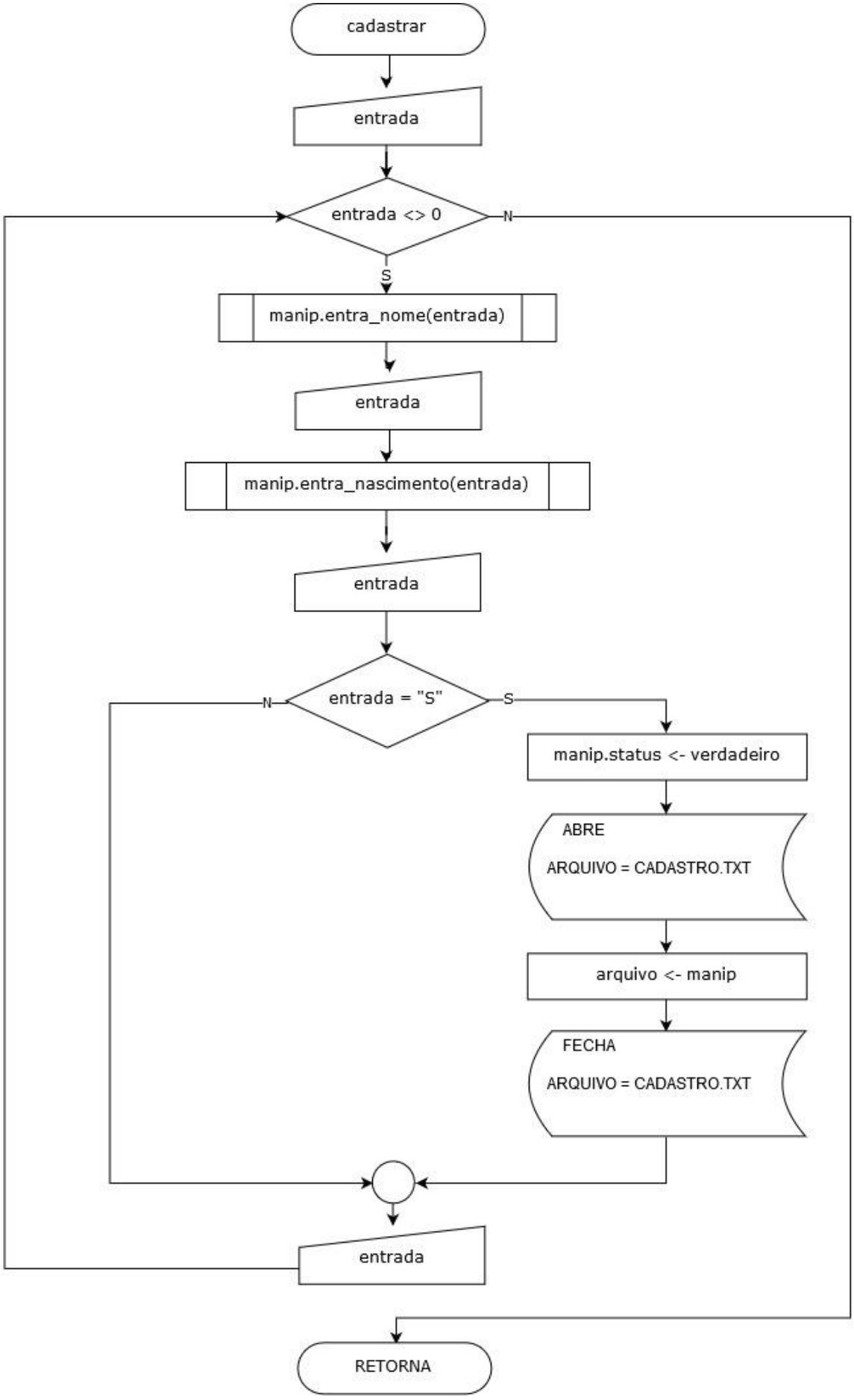


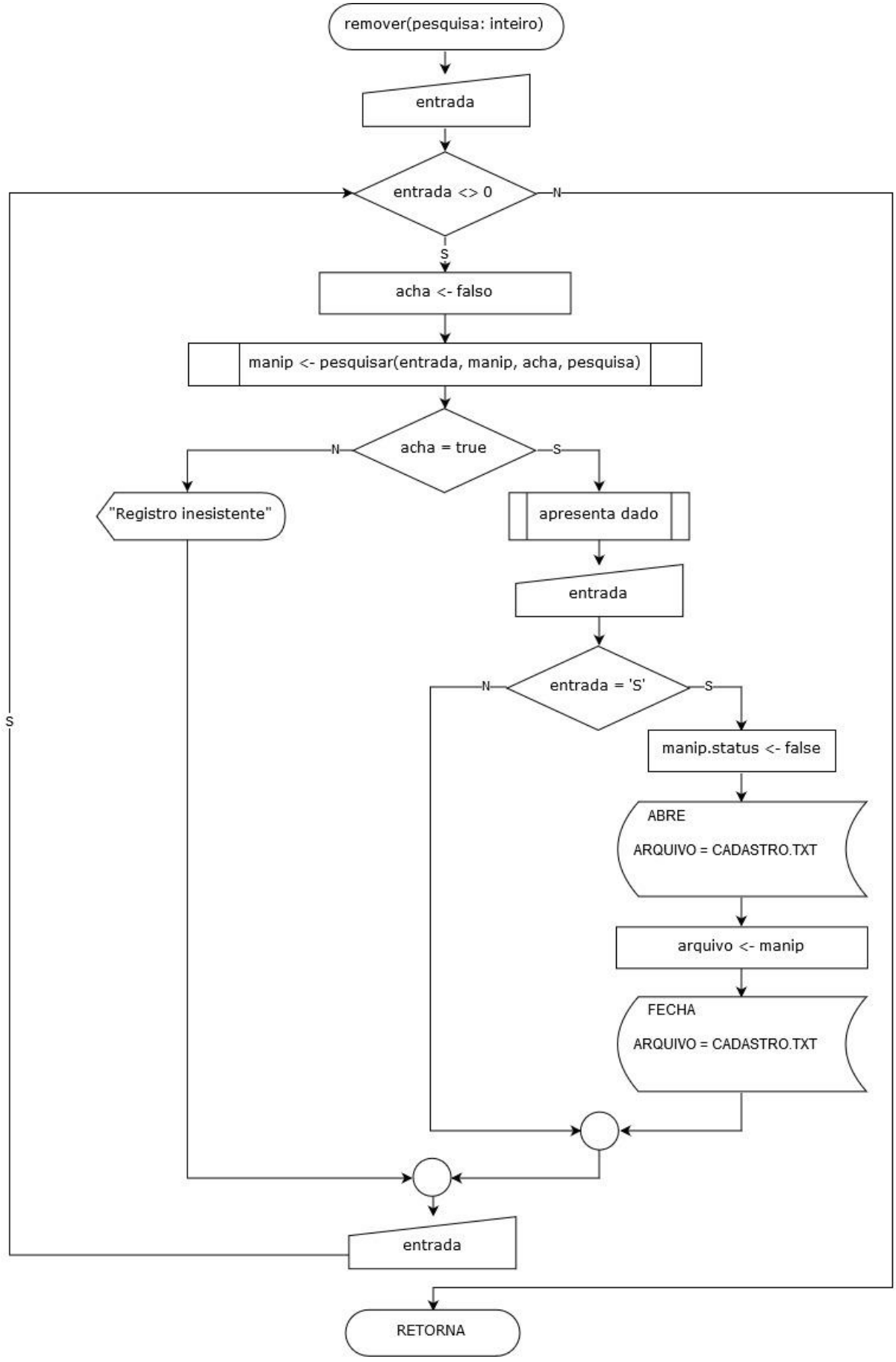
DIAGRAMA DE CLASSE

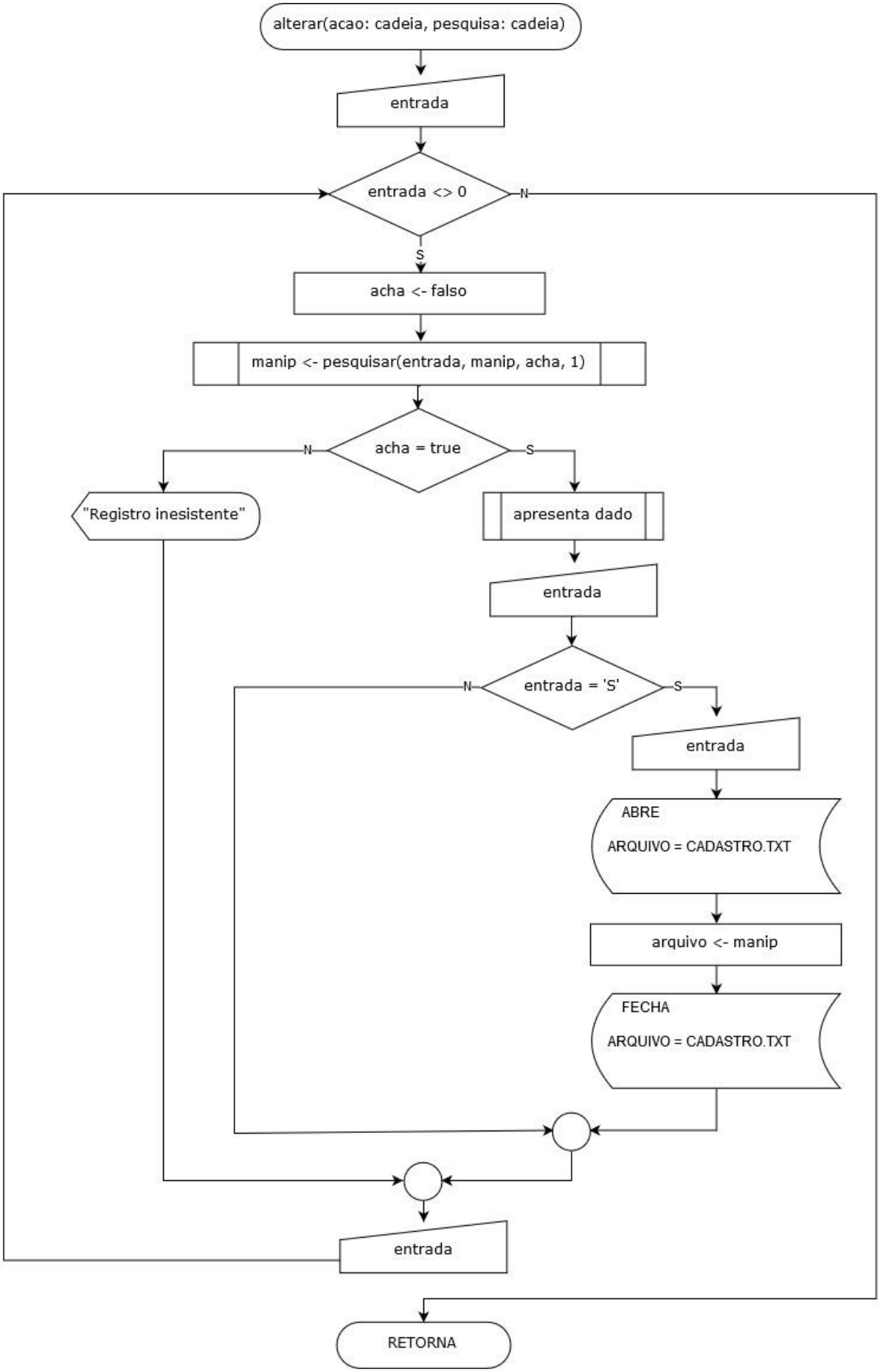
Classe: agenda
- nome: cadeia - nascimento: cadeia
+ entra _nome(e_nome : cadeia) + entra_nascimento(e_nascimento : cadeia) + mostra_nome: cadeia + mostra_nascimento:cadeia + posicao: inteiro + status: lógico

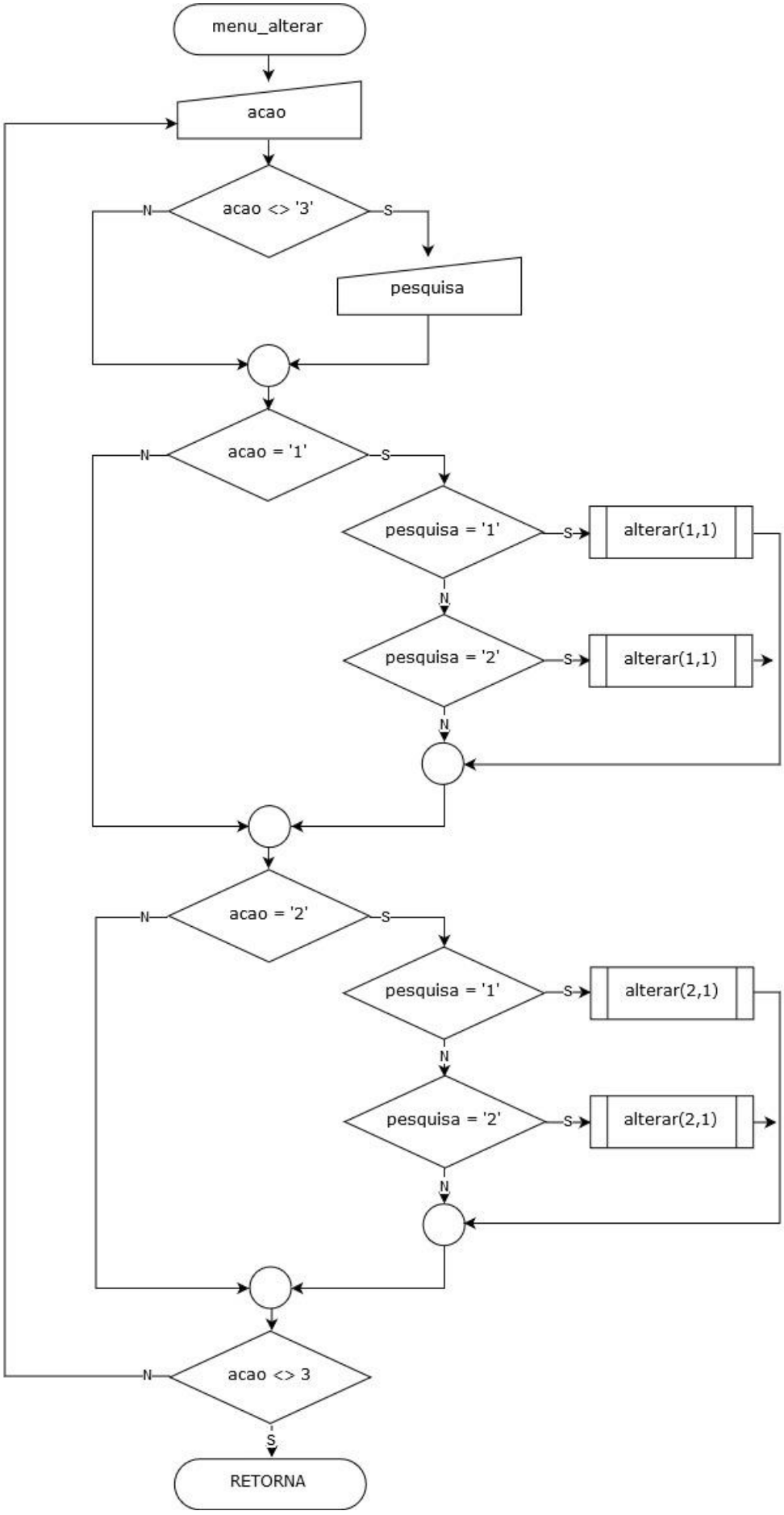
DIAGRAMA DE BLOCOS

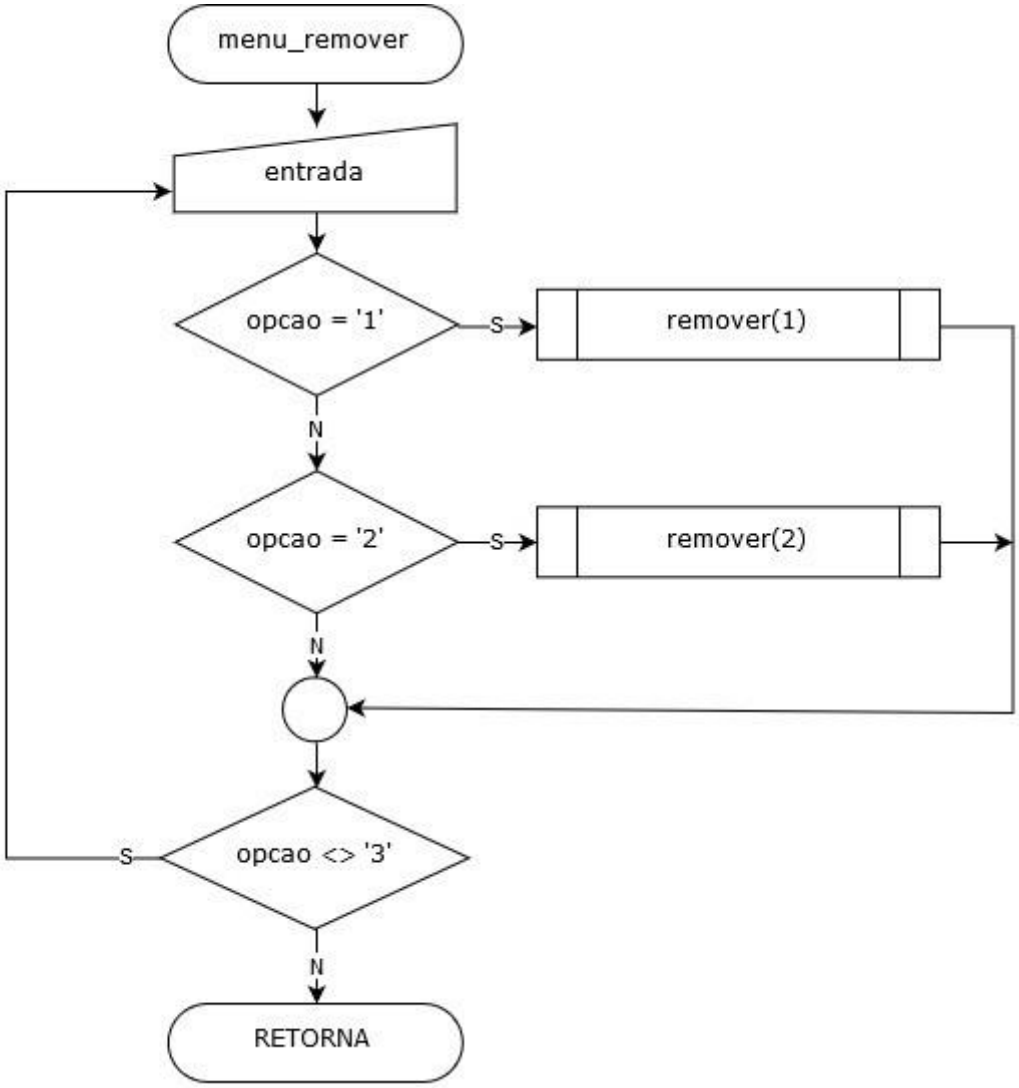


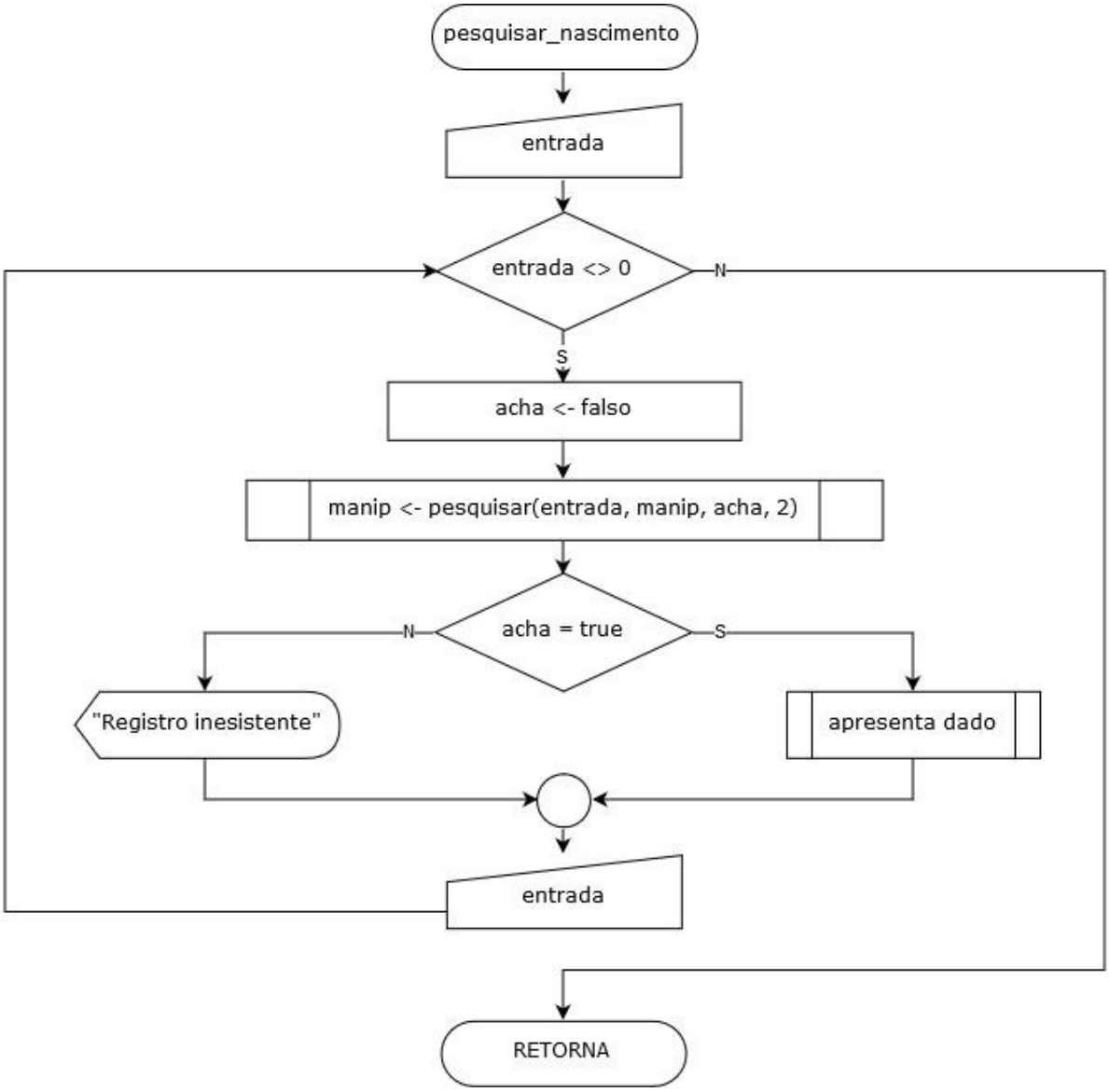

cadastrar
entrada
entrada <> 0
N
S
manip.entra_nome(entrada)
entrada
manip.entra_nascimento(entrada)
entrada
entrada = "S"
N
S
manip.status <- verdadeiro
ABRE
ARQUIVO = CADASTRO.TXT
arquivo <- manip
FECHA
ARQUIVO = CADASTRO.TXT
entrada
RETORNA

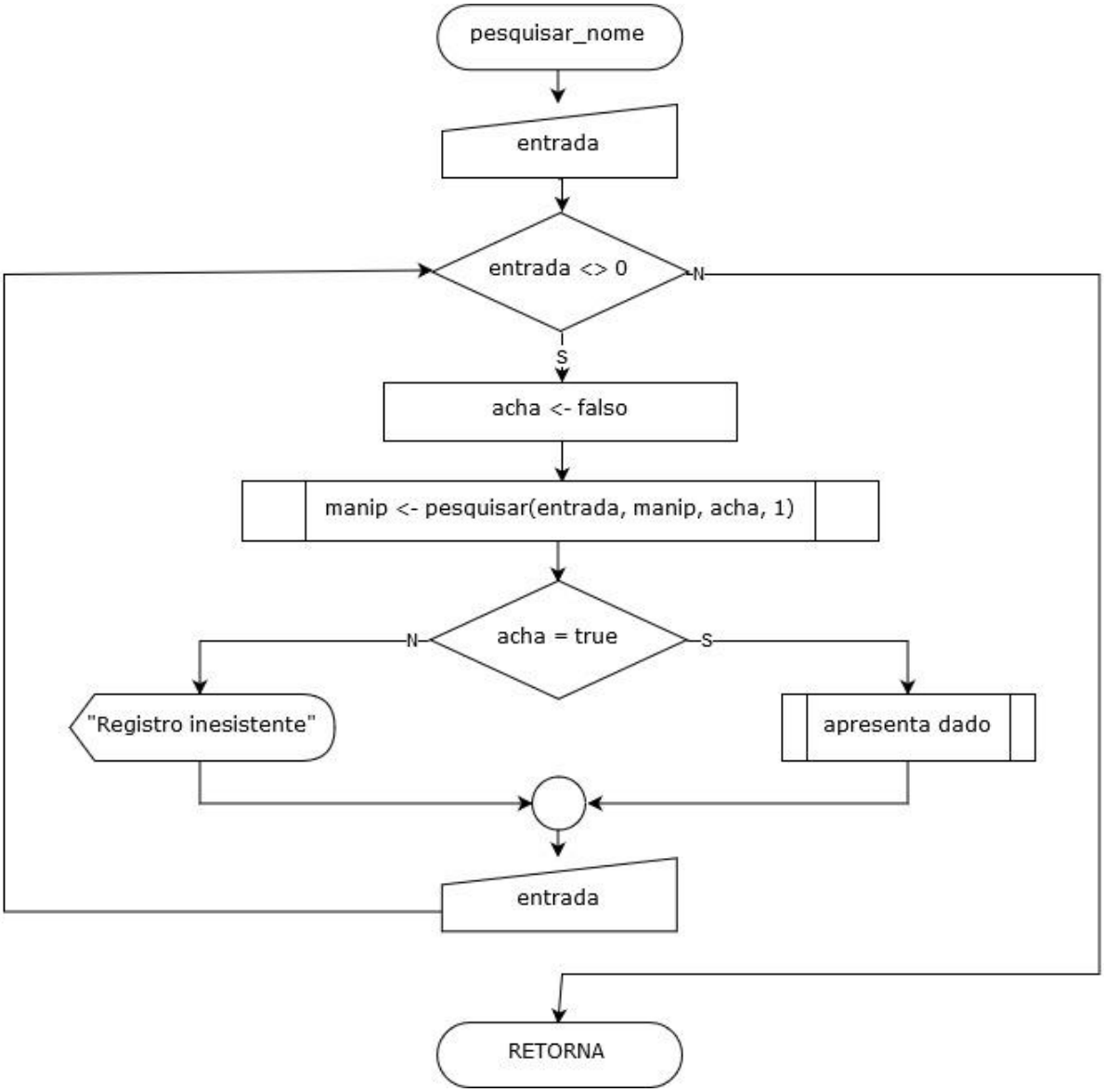

remover(pesquisa: inteiro)
entrada
entrada <> 0
N
S
acha <- falso
manip <- pesquisar(entrada, manip, acha, pesquisa)
acha = true
N
S
"Registro inesistente"
apresenta dado
entrada
entrada = 'S'
N
S
manip.status <- false
ABRE
ARQUIVO = CADASTRO.TXT
arquivo <- manip
FECHA
ARQUIVO = CADASTRO.TXT
S
entrada
RETORNA


alterar(acao: cadeia, pesquisa: cadeia)
entrada
entrada <> 0
N
S
acha <- falso
manip <- pesquisar(entrada, manip, acha, 1)
acha = true
N
S
"Registro inesistente"
apresenta dado
entrada
entrada = 'S'
N
S
entrada
ABRE
ARQUIVO = CADASTRO.TXT
arquivo <- manip
FECHA
ARQUIVO = CADASTRO.TXT
entrada
RETORNA


menu_alterar
acao
acao <> '3'
N
S
pesquisa
acao = '1'
N
S
pesquisa = '1'
S
alterar(1,1)
N
pesquisa = '2'
S
alterar(1,1)
N
acao = '2'
N
S
pesquisa = '1'
S
alterar(2,1)
N
pesquisa = '2'
S
alterar(2,1)
N
acao <> 3
N
S
RETORNA


menu_remover
entrada
opcao = '1'
S
remover(1)
N
opcao = '2'
S
remover(2)
N
opcao <> '3'
S
N
RETORNA


pesquisar_nascimento
entrada
entrada <> 0
N
S
acha <- falso
manip <- pesquisar(entrada, manip, acha, 2)
acha = true
N
S
"Registro inesistente"
apresenta dado
entrada
RETORNA


pesquisar_nome
entrada
entrada <> 0
N
S
acha <- falso
manip <- pesquisar(entrada, manip, acha, 1)
acha = true
N
S
"Registro inesistente"
apresenta dado
entrada
RETORNA

CODIFICAÇÃO

```
// Projeto_Main.cpp

#include "Projeto_Biblio.h"

int main(void){
    testa_arquivo();
    char opcao;

    do{
        tela(1,0);
        cin >> opcao;
        cin.ignore(80, '\n');
        cout << endl;

        switch (opcao){
            case '1': cadastrar();             break;
            case '2': pesquisar_nome();        break;
            case '3': pesquisar_nascimento(); break;
            case '4': menu_remover();          break;
            case '5': menu_alterar();          break;
        }
    }
    while (opcao != '6');
    return 0;
}
```

```
// Projeto_ Biblio.h

#include <iostream>
#include <fstream>
#include <cstring>
#include <clocale>
#include <cmath>
#include <ctime>
#include <windows.h>
using namespace std;

#define arquivo_nome "Cadastro.txt"     // Nome do arquivo a
ser criado.
#define nome_tamanho 41                 // Número de
caracteres que o campo-membro "nome" deve contemplar.
#define nascimento_tamanho 11


#if defined _WIN32 || defined _WIN64


// Rotinas para formatação de tela

// Limpa tela

void limpa(void){
    HANDLE TELA;
    DWORD ESCRITA = 0;
    COORD POS;
    TELA = GetStdHandle(STD_OUTPUT_HANDLE);
    POS.X = 0;
    POS.Y = 0;
    FillConsoleOutputCharacter(TELA, 32, 100 * 100, POS,
&ESCRITA);
}


// Posicionar cursor nos limites 80 x 24

void position(int LINHA, int COLUNA){
    if (COLUNA >= 1 and COLUNA <= 80 and LINHA >= 1 and LINHA
<= 24) {
        HANDLE TELA;
        COORD POS;
        TELA = GetStdHandle(STD_OUTPUT_HANDLE);
        POS.X = COLUNA - 1;
        POS.Y = LINHA - 1;
        SetConsoleCursorPosition(TELA, POS);
    }
}
```

```
// Limpar linha a partir da posicao do cursor

void clearline(void) {
    HANDLE TELA;
    COORD POS;
    CONSOLE_SCREEN_BUFFER_INFO VIDEO;
    DWORD ESCRITA = 0;
    TELA = GetStdHandle(STD_OUTPUT_HANDLE);
    GetConsoleScreenBufferInfo(TELA, &VIDEO);
    POS.Y = VIDEO.dwCursorPosition.Y;
    POS.X = VIDEO.dwCursorPosition.X;
    FillConsoleOutputCharacter(TELA, 32, 80 - POS.X, POS,
&ESCRITA);
}

#else

// ACOES EXCLUSIVAS PARA MODO: TERMINAL ANSI

// LIMPAR TELA

void limpa(void){
cout << "\033[2J";
}

// LIMPAR LINHA A PARTIR DA POSICAO DO CURSOR

void position(int LINHA, int COLUNA){
if (COLUNA >= 1 && COLUNA <= 80 && LINHA >= 1 && LINHA <= 24)
    cout << "\033[" << LINHA << ";" << COLUNA << "H";
}

// LIMPAR LINHA A PARTIR DA POSICAO DO CURSOR

void clearline(void) {
    cout << "\033[K";
}

#endif

#ifndef __Projeto_Biblio_h
#define __Projeto_Biblio_h

// Classe e sub-rotinas membro

class agenda{
private:
    char nome[nome_tamanho];
    char nascimento[nascimento_tamanho];
public:
    void entra_nome(char *e_nome);
```

```
    void entra_nascimento(char *e_nascimento);
    char *mostra_nome(void);
    char *mostra_nascimento(void);
    int posicao;
    bool status = false;
};

void agenda::entra_nome(char *e_nome){
    strcpy(nome, e_nome);
    return;
}

void agenda::entra_nascimento(char *e_nascimento){
    strcpy(nascimento, e_nascimento);
    return;
}

char *agenda::mostra_nome(void){
    return nome;
}

char *agenda::mostra_nascimento(void){
    return nascimento;
}

// Rotinas para tratamento de data e idade

void rmvachr(char CADEIA[], char CARACTERE){
    char *POS1 = CADEIA;
    char *POS2 = CADEIA;
    while (*POS1){
        *POS2 = *POS1++;
        POS2 += (*POS2 != CARACTERE);
    }
    *POS2 = '\0';
    return;
}

short sday(char DT[])
{
    return atoi(&DT[0]);
}

short smonth(char DT[])
{
    return atoi(&DT[3]);
}

short syear(char DT[])
{
    return atoi(&DT[6]);
```

```
}

bool leapyear(char DT[])
{
  if (syear(DT) % 400 == 0)
    return true;
  if (syear(DT) % 4 == 0 and syear(DT) % 100 != 0)
    return true;
  return false;
}

short lastday(char DT[])
{
  if (smonth(DT) == 2) // FEV
    return (leapyear(DT)) ? 29 : 28;
  if (smonth(DT) <= 7) // JUL
    return (smonth(DT) % 2 != 0) ? 31 : 30;
  return (smonth(DT) % 2 != 0) ? 30 : 31;
}

bool validate(char DT[])
{
  return syear(DT)  >=    1 and
         syear(DT)  <= 9999 and
         smonth(DT) >=    1 and
         smonth(DT) <=   12 and
         sday(DT)   >=    1 and
         sday(DT)   <= lastday(DT);
}

unsigned long int dateansi(char DT[]){
    char  DATATEMP1[11] = {};
    char  DATATEMP2[11] = {};
    short I;
    strcpy(DATATEMP1, DT);
    for (I = 0; I <= 9; I++){
        if (I < 4)
            DATATEMP2[I] = DATATEMP1[I + 6];
        if (I == 4)
            DATATEMP2[4] = '/';
        if (I > 4 and I < 7)
            DATATEMP2[I] = DATATEMP1[I - 2];
        if (I == 7)
            DATATEMP2[7] = '/';
        if (I > 7)
            DATATEMP2[I] = DATATEMP1[I - 8];
    }
    rmvachr(DATATEMP2, '/');
    return atol(DATATEMP2);
}
```

```
unsigned long int julianday(char DT[]){
    short DIA, MES, ANO;
    float A, B, C, D, E, F;
    DIA = sday(DT);
    MES = smonth(DT);
    ANO = syear(DT);
    if (MES < 3){
        ANO = ANO - 1;
        MES = MES + 12;
    }
    if (dateansi(DT) >= 2299161){
        A = floor(ANO / 100);
        B = floor(A / 4);
        C = 2 - A + B;
    }
    else
        C = 0;
    D = floor(365.25 * (ANO + 4716));
    E = floor(30.6001 * (MES + 1));
    F = D + E + DIA + 0.5 + C - 1524.5;
    return (unsigned long int) F;
}

char *jultodate(unsigned long int DJ){
    static char DATA[11];
    float A, B, C, D, E, F, G, H, I, J, K;
    A = (float)DJ;
    if (A > (float)2299160){
        B = (float)floor((A - 1867216.25) / 36524.25);
        C = A + 1 + B - (float)floor(B / 4);
    }
    else
        C = A;
    D = C + 1524;
    E = (float)floor((D - 122.1) / 365.25);
    F = (float)floor(E * 365.25);
    G = (float)floor((D - F) / 30.6001);
    H = D - F - floor(G * 30.6001);
    if (G < 14)
        I = G - 1;
    else
        I = G - 13;
    if (I > 2)
        J = E - 4716;
    else
        J = E - 4715;
    if (J > 0)
        K = J;
    else
        K = (J + 1) * -1;
    sprintf(DATA, "%2.f/%2.f/%4.f", H, I, K);
```

```
    return DATA;
}

short mes_char_int(char *mes){
    short n_mes = 0;
    if (strcmp(mes, "Jan") == 0)
        n_mes = 1;
    else if (strcmp(mes, "Feb") == 0)
        n_mes = 2;
    else if (strcmp(mes, "Mar") == 0)
        n_mes = 3;
    else if (strcmp(mes, "Apr") == 0)
        n_mes = 4;
    else if (strcmp(mes, "May") == 0)
        n_mes = 5;
    else if (strcmp(mes, "Jun") == 0)
        n_mes = 6;
    else if (strcmp(mes, "Jul") == 0)
        n_mes = 7;
    else if (strcmp(mes, "Aug") == 0)
        n_mes = 8;
    else if (strcmp(mes, "Sep") == 0)
        n_mes = 9;
    else if (strcmp(mes, "Oct") == 0)
        n_mes = 10;
    else if (strcmp(mes, "Nov") == 0)
        n_mes = 11;
    else if (strcmp(mes, "Dec") == 0)
        n_mes = 12;
    return n_mes;
}

char *formata_data(char *agora_char){
    short dia, mes, ano;
    char mes_char[4];
    for (int i = 0; i < 3; i++)
        mes_char[i] = agora_char[i + 4];

    mes = mes_char_int(mes_char);
    dia = atoi(&agora_char[8]);
    ano = atoi(&agora_char[20]);

    sprintf(agora_char, "%2.2d/%2.2d/%4.4d", dia, mes, ano);

    return agora_char;
}

char *calcula_hoje(void){
    time_t agora = time(NULL);
    char *agora_char = ctime(&agora);
```

```
    char *p_hoje;

    p_hoje = formata_data(agora_char);

    return p_hoje;
}

void calcula_idade(char *p_nascimento, int &anos, int &meses,
int &dias){
    unsigned long int diferenca = (julianday(calcula_hoje()))
- (julianday(p_nascimento));

    anos  = (diferenca / 365.25);
    meses = ((diferenca - (anos * 365.25)) / 30.4375);
    dias = (diferenca - (anos * 365.25) - (meses * 30.4375));

    return;
}

// Rotinas secundarias

void maiusculo(char *p_entrada){
    while(*p_entrada){
        if(*p_entrada >= 97 && *p_entrada <= 122)
            *p_entrada -= 32;
        p_entrada++;
    }
}

bool vazia(){
    bool vazia = true;
    agenda manip;
    manip.status = false;
    fstream arquivo(arquivo_nome, ios_base::in |
ios_base::binary);
    while(!arquivo.eof() && manip.status == false){
        arquivo.read(reinterpret_cast<char*>(&manip),
sizeof(manip));
    }
    if (manip.status == true)
        vazia = false;

    arquivo.flush();
    arquivo.close();
    return vazia;
}

void testa_arquivo(void){
    fstream arquivo(arquivo_nome, ios_base::in |
ios_base::binary);
```

```
    if (arquivo.fail())
        fstream arquivo(arquivo_nome, ios_base::out |
ios_base::binary);

    arquivo.flush();
    arquivo.close();
    return;
}

unsigned int nova_posicao(void){
    fstream arquivo(arquivo_nome, ios_base::in |
ios_base::binary);
    arquivo.seekg(0, ios_base::end);

    unsigned int tamanho_arquivo;
    if (arquivo.tellg() <= 0)
        tamanho_arquivo = 0;
    else
        tamanho_arquivo = arquivo.tellg();
    unsigned int n_posicao = tamanho_arquivo / sizeof(agenda);

    arquivo.flush();
    arquivo.close();

    return n_posicao;
}

void apresenta_dado_pesquisa(agenda manip){
    int dias, meses, anos;
    calcula_idade(manip.mostra_nascimento(), anos, meses,
dias);

    cout << "          Nome.....................: " <<
manip.mostra_nome() << endl;
    cout << "          Data de nascimento.......: " <<
manip.mostra_nascimento() << endl;
    cout << "          Anos de idade............: " << anos
<< endl;
    cout << "          Meses de idade...........: " <<
meses << endl;
    cout << "          Dias de idade............: " << dias
<< endl;
    cout << endl;
    return;
}

void apresenta_dado(agenda manip){
    int dias, meses, anos;
    calcula_idade(manip.mostra_nascimento(), anos, meses,
dias);
```

```
    cout << "          Nome.....................: " <<
manip.mostra_nome() << endl;
    cout << "          Data de nascimento.......: " <<
manip.mostra_nascimento() << endl;
    cout << endl;
    return;
}

agenda pesquisar(char *entrada, agenda manip, bool &acha, int
opcao){
    fstream arquivo(arquivo_nome, ios_base::in | ios_base::
binary);

    if (opcao == 1){    // Pesquisa por nome.
        while (!(arquivo.eof()) && acha == false){
            arquivo.read(reinterpret_cast<char*>(&manip),
sizeof(manip));
            if (strcmp(entrada, manip.mostra_nome()) == 0 &&
manip.status == true){
                acha = true;
            }
        }
    }
    else{               // Pesquisa por data de nascimento.
        while (!(arquivo.eof()) && acha == false){
            arquivo.read(reinterpret_cast<char*>(&manip),
sizeof(manip));
            if (strcmp(entrada, manip.mostra_nascimento()) ==
0 && manip.status == true){
                acha = true;
            }
        }
    }

    arquivo.flush();
    arquivo.close();
    return manip;
}

// Rotinas para tratamento de tela

void traca_linha(void){
for (int i = 0; i < 80; i++)
    cout << "-";
return;
}

void tela(short tela, short mensagem){
switch (tela){
case 0:
    switch (mensagem){
```

```
    case 0:
        position(18,1);     cout << "Acao invalida, a agenda
esta vazia! Tecle <Enter> para voltar.";
        cin.get();
        break;
    }
    break;

case 1:
    switch (mensagem){
    case 0:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,33);     cout << "Menu principal";
        position(4,1);      traca_linha();

        position(7,15);     cout << "CADASTRAR
REGISTRO.........................: [1]";
        position(8,15);     cout << "PESQUISAR REGISTRO POR
NOME.................: [2]";
        position(9,15);     cout << "PESQUISAR REGISTRO POR
DATA DE NASCIMENTO....: [3]";
        position(10,15);    cout << "REMOVER
REGISTRO.............................: [4]";
        position(11,15);    cout << "ALTERAR
REGISTRO.............................: [5]";
        position(12,15);    cout << "SAIR DO
PROGRAMA.............................: [6]";
        position(16,1);     cout << "Selecione uma opcao: ";
        break;
    }
    break;

case 2:
    switch (mensagem){
    case 0:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,35);     cout << "Cadastrar";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre um nome ou <0> para
sair: ";
        position(8,1);
        break;

    case 1:
        position(10,1);     cout << "Entre a data de
nascimento no formato \"DD/MM/AAAA\"";
```

```
        position(11,1);
        break;

    case 3:
        position(16,1);     cout << "Entrada incorreta! Por
favor, entre uma data valida.";
        cin.get();
        position(11,1);     clearline();
        position(16,1);     clearline();
        break;
    }
    break;

case 3:
    switch (mensagem){
    case 0:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,26);     cout << "Pesquisar registro por
nome";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre um nome ou <0> para
sair: ";
        position(8,1);
        break;
    }
    break;

case 4:
    switch (mensagem){
    case 0:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,23);     cout << "Pesquisar registro por
nascimento";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre uma data de
nascimento ou <0> para sair: ";
        position(8,1);
        break;
    }
    break;

case 5:
    switch (mensagem){
    case 0:
        limpa();
```

```
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Remover";
        position(4,1);      traca_linha();

        position(7,15);     cout << "PESQUISAR REGISTRO POR
NOME.................: [1]";
        position(8,15);     cout << "PESQUISAR REGISTRO POR
DATA DE NASCIMENTO....: [2]";
        position(9,15);     cout << "VOLTAR AO MENU
ANTERIOR.....................: [3]";
        position(16,1);     cout << "Selecione uma opcao: ";
        break;

    case 1:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,32);     cout << "Remover por nome";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre o nome do contato a
ser removido ou <0> para sair: ";
        position(8,1);
        break;

    case 2:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,29);     cout << "Remover por nascimento";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre a data de
nascimento do contato a ser removido ou <0> para sair: ";
        position(8,1);
        break;

    }
    break;

case 6:
    switch(mensagem){
    case 0:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();
```

```
        position(7,15);     cout << "ALTERAR
NOME.............................: [1]";
        position(8,15);     cout << "ALTERAR DATA DE
NASCIMENTO.................: [2]";
        position(9,15);     cout << "VOLTAR AO MENU
ANTERIOR....................: [3]";
        position(16,1);     cout << "Selecione uma opcao: ";
        break;

    case 1:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();

        position(7,15);     cout << "PESQUISAR REGISTRO POR
NOME.................: [1]";
        position(8,15);     cout << "PESQUISAR REGISTRO POR
DATA DE NASCIMENTO....: [2]";
        position(9,15);     cout << "VOLTAR AO MENU
ANTERIOR....................: [3]";
        position(16,1);     cout << "Selecione uma opcao: ";
        break;

    case 2:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre o nome do contato a
ser alterado ou <0> para sair: ";
        position(8,1);
        break;

    case 3:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre a data de
nascimento do contato a ser alterado ou <0> para sair: ";
        position(8,1);
        break;

    case 4:
        limpa();
```

```
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre o novo nome: ";
        position(8,1);
        break;

    case 5:
        limpa();
        position(1,1);      traca_linha();
        position(2,32);     cout << "PROGRAMA AGENDA";
        position(3,36);     cout << "Alterar";
        position(4,1);      traca_linha();

        position(7,1);      cout << "Entre a nova data de
nascimento: ";
        position(8,1);
        break;

    case 6:
        position(16,1);     cout << "Entrada incorreta! Por
favor, entre uma data valida.";
        cin.get();
        position(11,1);     clearline();
        position(16,1);     clearline();
        break;

    };
    break;
}

return;
}

// Rotinas principais

void cadastrar(void){
    tela(2,0);
    agenda manip;
    char entrada[nome_tamanho];

    cin.getline(entrada, sizeof(entrada));
    maiusculo(entrada);

    while (!(strcmp(entrada, "0") == 0)){
        manip.entra_nome(entrada);
        cout << endl;

        do{
```

```
            tela(2,1);
            cin >> entrada;
            cin.ignore(80, '\n');
            if (!validate(entrada))
                tela(2,3);
        }
        while (!validate(entrada));
        manip.entra_nascimento(entrada);
        cout << endl;

        manip.posicao = nova_posicao();

        apresenta_dado(manip);
        cout << "Confirmar cadastro? [S/N]: ";
        cin >> entrada;
        cin.ignore(80, '\n');
        cout << endl;

        if (toupper(entrada[0]) == 'S'){
            manip.status = true;
            fstream arquivo(arquivo_nome, ios_base::out |
ios_base::app | ios_base::binary);
            arquivo.write(reinterpret_cast<char*>(&manip),
sizeof(manip));
            arquivo.flush();
            arquivo.close();
            cout << "Contato cadastrado com sucesso! Tecle
<Enter para continuar." << endl;
            cin.get();
            cout << endl;
        }

        tela(2,0);
        cin.getline(entrada, sizeof(entrada));
        maiusculo(entrada);
    }
    cout << endl;
    return;
}

void pesquisar_nome(void){
    tela(3,0);
    agenda manip;
    char entrada[nome_tamanho];
    bool acha;

    cin.getline(entrada, sizeof(entrada));
    maiusculo(entrada);
    cout << endl;

    while (!(strcmp(entrada, "0") == 0)){
```

```
        acha = false;
        manip = pesquisar(entrada, manip, acha, 1);

        if (acha == true){
            apresenta_dado_pesquisa(manip);
            cout << "Tecle <Enter> para continuar.";
            cin.get();
        }
        else{
            cout << "Registro inesistente! Tecle <Enter> para
continuar.";
            cin.get();
        }

        tela(3,0);
        cin.getline(entrada, sizeof(entrada));
        maiusculo(entrada);
        cout << endl;
    }
    return;
}

void pesquisar_nascimento(void){
    agenda manip;
    char entrada[nome_tamanho];
    bool acha;

    tela(4,0);
    cin >> entrada;
    cin.ignore(80, '\n');
    cout << endl;

    while (!(strcmp(entrada, "0") == 0)){
        acha = false;
        manip = pesquisar(entrada, manip, acha, 2);

        if (acha == true){
            apresenta_dado_pesquisa(manip);
            cout << "Tecle <Enter> para continuar.";
            cin.get();
        }
        else{
            cout << "Registro inesistente! Tecle <Enter> para
continuar." << endl;
            cin.get();
        }

        tela(4,0);
        cin >> entrada;
        cin.ignore(80, '\n');
        cout << endl;
```

```
    }
    return;
}

void remover(short pesquisa){
    agenda manip;
    char entrada[nome_tamanho];
    bool acha;


    if (pesquisa == 1){
        tela(5,1);
        cin.getline(entrada, sizeof(entrada));
        maiusculo(entrada);
        cout << endl;
    }
    else{
        tela(5,2);
        cin >> entrada;
        cin.ignore(80, '\n');
        cout << endl;
    }

    while (!(strcmp(entrada, "0")) == 0){
        acha = false;
        manip = pesquisar(entrada, manip, acha, pesquisa);

        if (acha == true){
            apresenta_dado(manip);
            cout << "Remover registro? [S/N]: ";
            cin >> entrada;
            cin.ignore(80, '\n');
            cout << endl;

            if (toupper(entrada[0]) == 'S'){
                manip.status = false;
                fstream arquivo(arquivo_nome, ios_base::in |
ios_base::out | ios_base::binary);
                arquivo.seekp(manip.posicao*(sizeof(manip)),
ios_base::beg);
                arquivo.write(reinterpret_cast<char*>(&manip),
sizeof(manip));
                arquivo.flush();
                arquivo.close();
                cout << "Contato removido com sucesso! Tecle
<Enter> para continuar.";
                cin.get();
                cout << endl;
            }
        }
        else{
```

```
            cout << "Registro insistente! Tecle <Enter> para
continuar.";
            cin.get();
        }

        if (pesquisa == 1){
            tela(5,1);
            cin.getline(entrada, sizeof(entrada));
            maiusculo(entrada);
            cout << endl;
        }
        else{
            tela(5,2);
            cin >> entrada;
            cin.ignore(80, '\n');
            cout << endl;
        }
    }
}

void alterar(char acao, short pesquisa){
    agenda manip;
    char entrada[nome_tamanho];
    bool acha;

    if (pesquisa == 1){
        tela(6,2);
        cin.getline(entrada, sizeof(entrada));
        maiusculo(entrada);
        cout << endl;
    }
    else{
        tela(6,3);
        cin >> entrada;
        cin.ignore(80, '\n');
        cout << endl;
    }

    while (!(strcmp(entrada, "0")) == 0){
        acha = false;
        manip = pesquisar(entrada, manip, acha, pesquisa);

        if (acha == true){
            apresenta_dado(manip);
            cout << "Alterar registro? [S/N]: ";
            cin >> entrada;
            cin.ignore(80, '\n');
            cout << endl;

            if (toupper(entrada[0]) == 'S'){
                if (acao == 1){
```

```
                    tela(6,4);
                    cin.getline(entrada, sizeof(entrada));
                    maiusculo(entrada);
                    manip.entra_nome(entrada);
                    cout << endl;
                }
                else{
                    do{
                        tela(6,5);
                        cin >> entrada;
                        cin.ignore(80, '\n');
                        if (!validate(entrada))
                            tela(6,6);
                    }
                    while(!validate(entrada));
                    manip.entra_nascimento(entrada);
                    cout << endl;
                }

                fstream arquivo(arquivo_nome, ios_base::in |
ios_base::out | ios_base::binary);
                arquivo.seekp(manip.posicao*(sizeof(manip)),
ios_base::beg);
                arquivo.write(reinterpret_cast<char*>(&manip),
sizeof(manip));
                arquivo.flush();
                arquivo.close();
                cout << "Contato alterado com sucesso! Tecle
<Enter> para continuar." << endl;
                cin.get();
            }
        }
        else{
            cout << "Registro insistente! Tecle <Enter> para
continuar.";
            cin.get();
        }

        if (pesquisa == 1){
            tela(6,2);
            cin.getline(entrada, sizeof(entrada));
            maiusculo(entrada);
            cout << endl;
        }
        else{
            tela(6,3);
            cin >> entrada;
            cin.ignore(80, '\n');
            cout << endl;
        }
    }
```

```
}

void menu_remover(void){
    char opcao;

    do{
        tela(5,0);
        cin >> opcao;
        cin.ignore(80, '\n');
        cout << endl;

        switch (opcao){
            case '1': remover(1);   break;
            case '2': remover(2);   break;
        }
    }
    while (opcao != '3');
    return;
}

void menu_alterar(void){
    char acao, pesquisa;

    do{
        tela(6,0);
        cin >> acao;
        cin.ignore(80, '\n');
        cout << endl;

        if (acao != '3'){
            tela(6,1);
            cin >> pesquisa;
            cin.ignore(80, '\n');
            cout << endl;
        }

        switch (acao){
            case '1': switch (pesquisa){
                case '1': alterar(1,1); break;
                case '2': alterar(1,2); break;
            }; break;
            case '2': switch (pesquisa){
                case '1': alterar(2,1); break;
                case '2': alterar(2,2); break;
            }; break;
        }
    }
    while (acao != '3');
    return;
}
```

```
#endif // __Projeto_Biblio_h
```