INSTITUTO FEDERAL DE SÂO PAULO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DAVY PAULINO DA SILVA DANTAS

PROJETO AGENDA MÉDICA

PROGRAMA (SOFTWARE)

CAMPOS DO JORDÂO

2019

DAVY PAULINO DA SILVA DANTAS

PROJETO AGENDA MÉDICA

Programa (software) desenvolvimento para a conclusão da matéria obrigatória, linguagem de programação I, para a conclusão de Tecnólogo em Análise e Desenvolvimento de Sistemas, no Instituto Federal de São Paulo.

Orientador: Prof. ME. Jose Augusto Navarro Garcia Manzano

CAMPOS DO JORDÃO

2019

RESUMO

Desenvolver programa para o gerenciamento de agenda para atendimento médico (BORGES, 1985, p. 86-93). O programa deve mostrar um menu contendo as opções: marcar atendimento, desmarcar atendimento, listar marcações do dia, clientes marcados no dia e mapa dos horários livres na agenda.

Para facilitar as operações do programa, considere que todos os meses do ano possuem 31 dias e que os horários de atendimentos serão realizados de hora em hora sempre das 8 às 17 horas sem intervalo para almoço. A agenda deve ser gravada em arquivo binário para posterior uso.

LISTA DE ILUSTRAÇÕES

Figura 1 - Estrutura hierárquica do programa agenda médica. 6
Figura 2 - Estrutura do registro de matriz CAD_AGENDA. 6
Figura 3 - Menu programa agenda médica 7
Figura 4 - Rotina de entrada do dia. 8
Figura 5 - Rotina de entrada da hora. 8
Figura 6 - Rotina de marcação de data de atendimento. 9
Figura 7 - Rotina de remoção de atendimento marcado. 10
Figura 8 - Rotina de remoção de atendimento marcado. 11
Figura 9 - Rotina de remoção de atendimento marcado. 11
Figura 10 - Rotina de exibição de todos os atendimentos do mês. 12

SUMÁRIO

1 DIAGRAMAS.....06
2 PROGRAMA 13
3 CONCLUSÕES FINAIS.....24
REFERÊNCIAS.....25

1 DIAGRAMAS

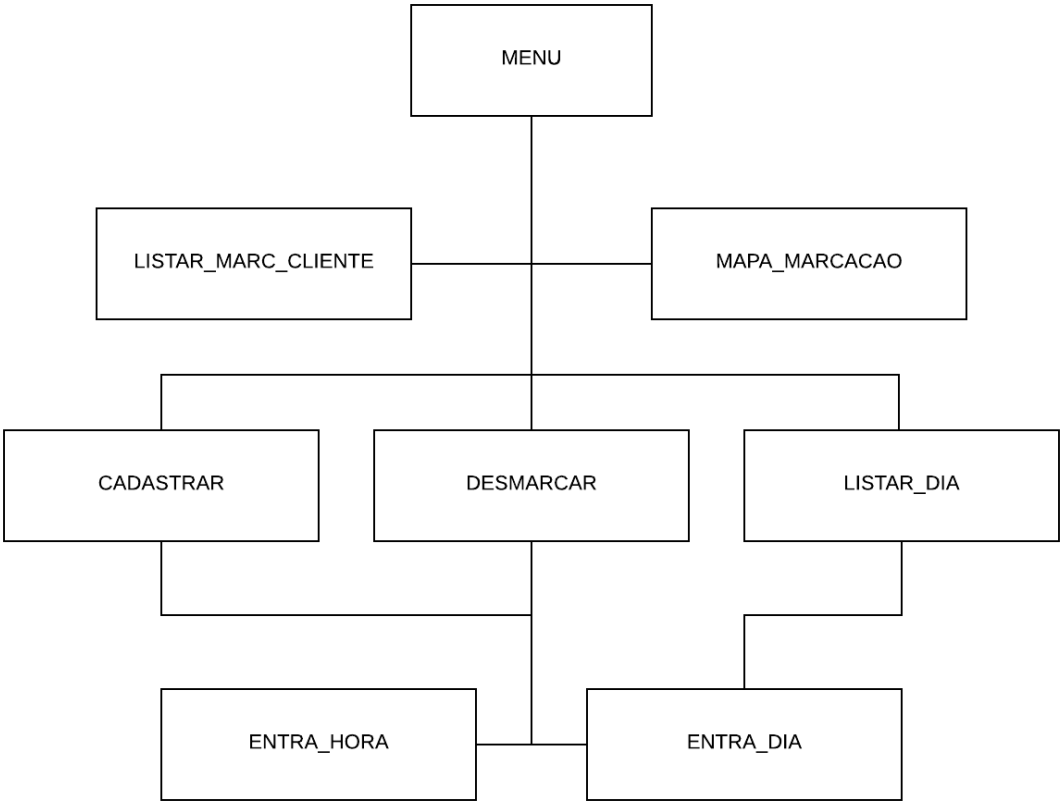

Figura 1 - Estrutura hierárquica do programa agenda médica.

Registro: CAD_AGENDA	
Campo	Tipo
NOME	cadeia

Figura 2 - Estrutura do registro de matriz CAD_AGENDA.

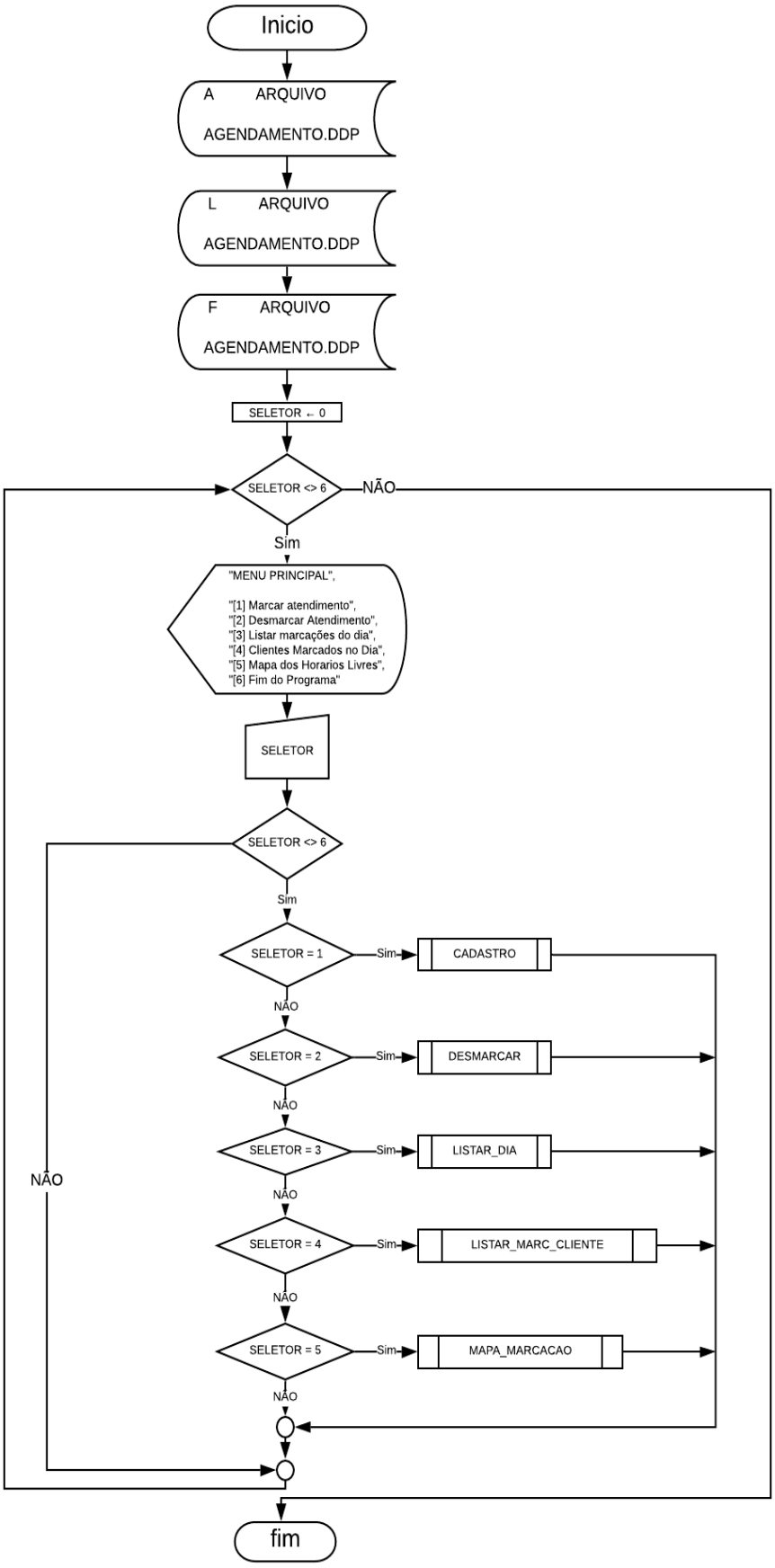

Figura 3 - Menu programa agenda médica

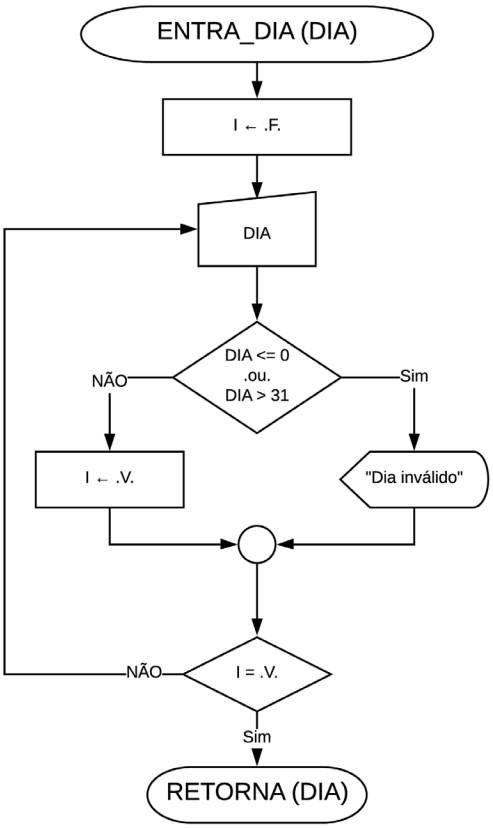

Figura 4 - Rotina de entrada do dia.

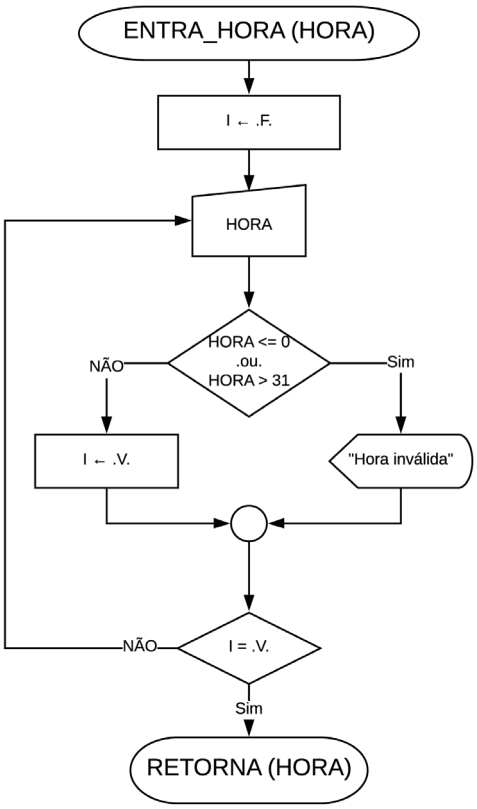

Figura 5 - Rotina de entrada da hora.

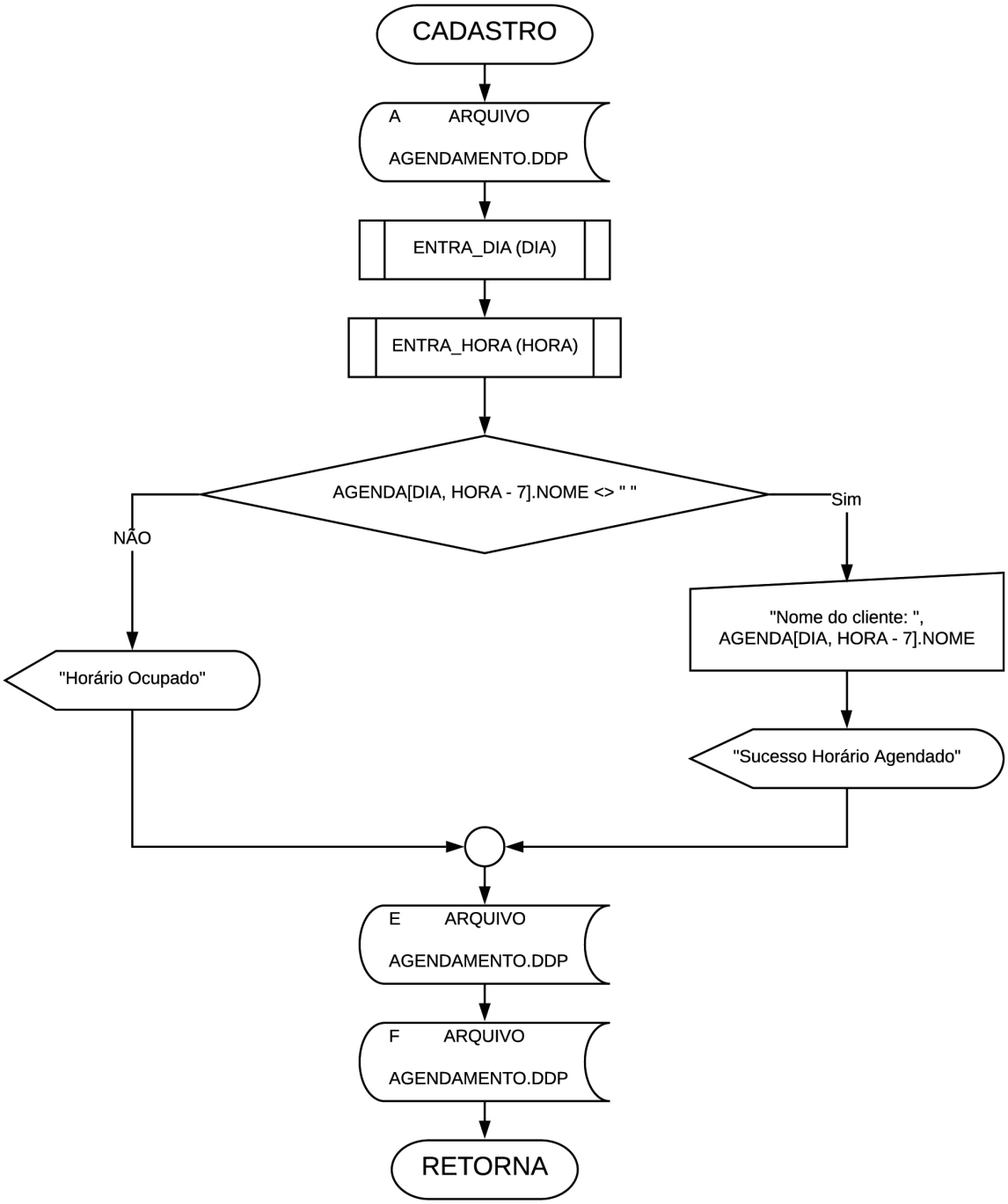

Figura 6 - Rotina de marcação de data de atendimento.

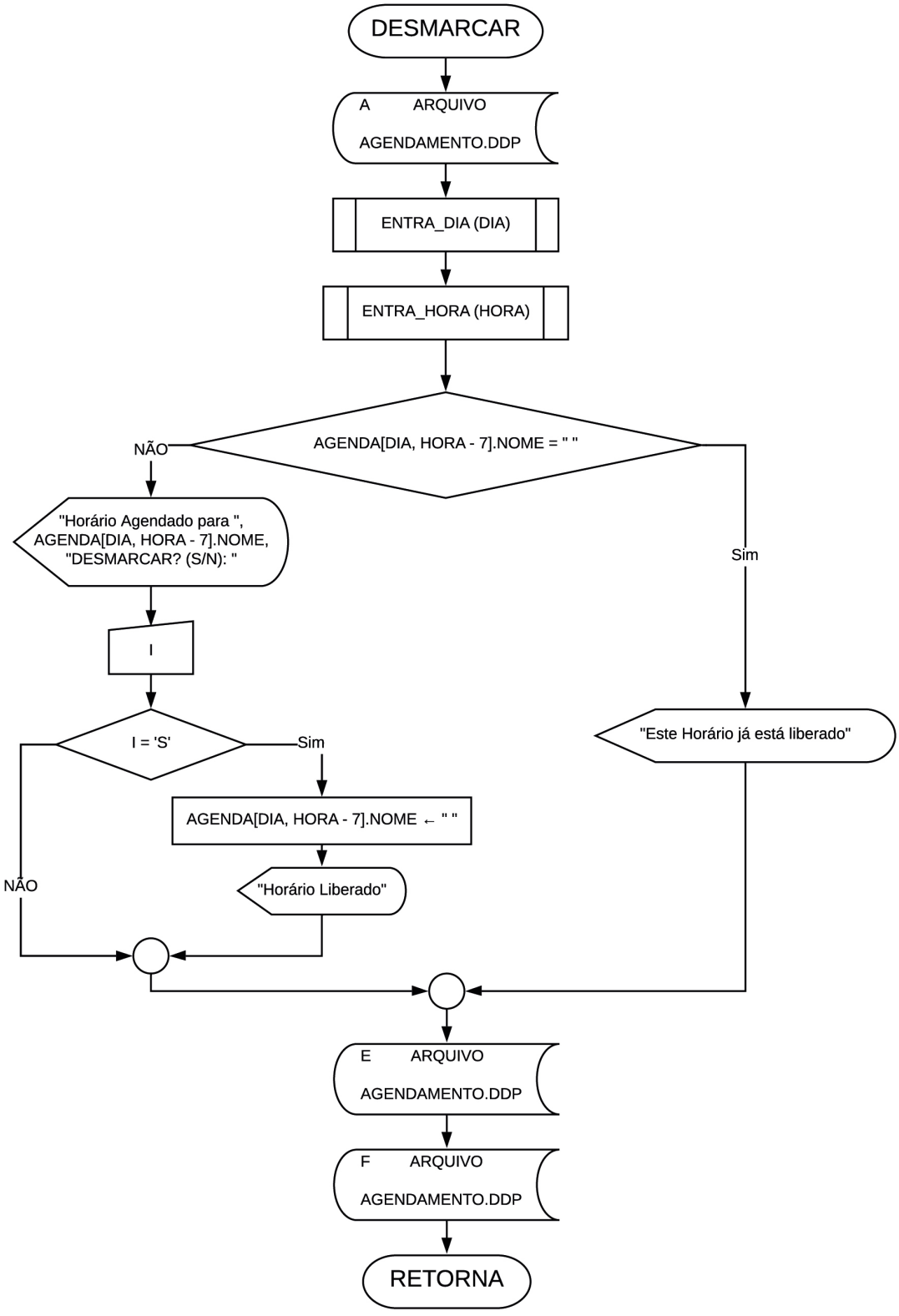

Figura 7 - Rotina de remoção de atendimento marcado.

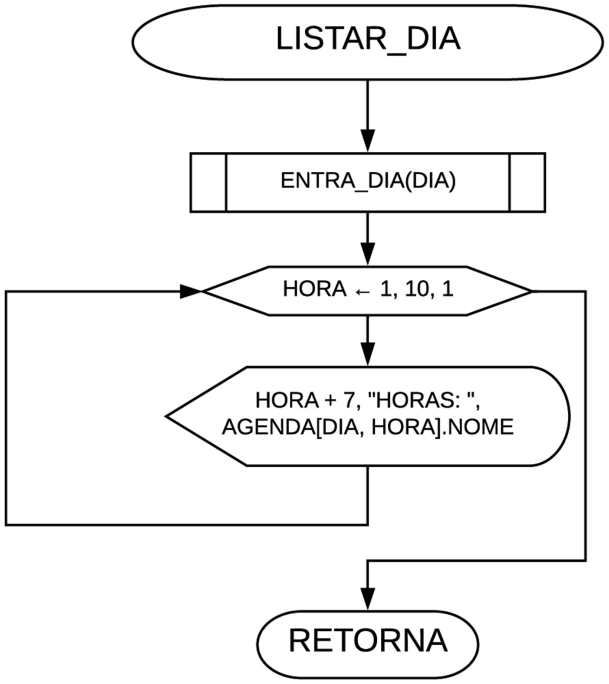

Figura 8 - Rotina de remoção de atendimento marcado.

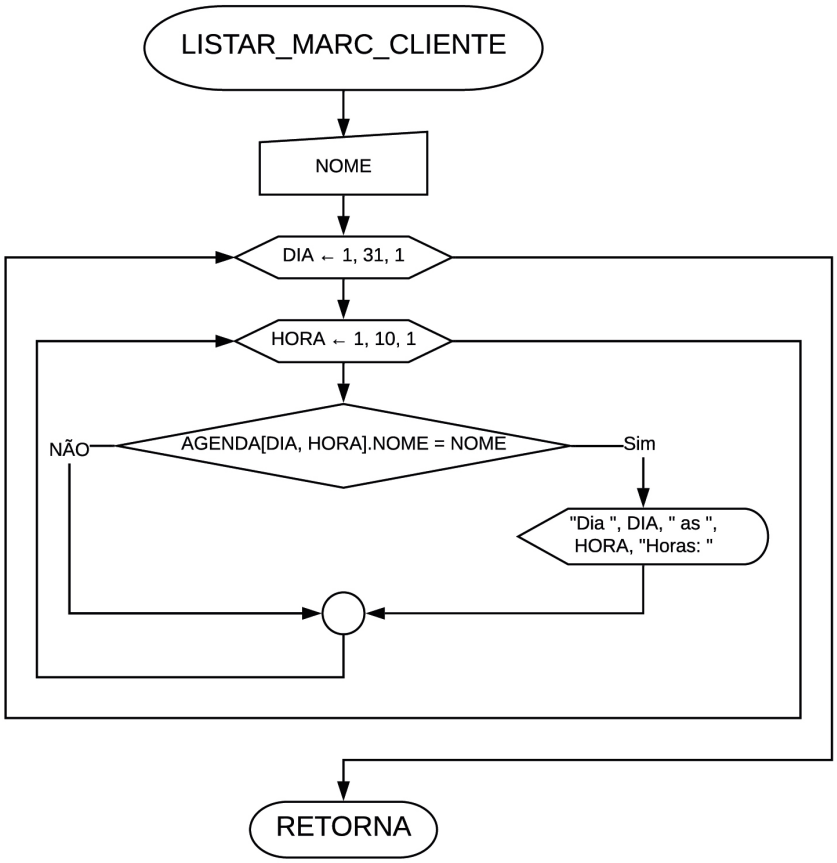

Figura 9 - Rotina de remoção de atendimento marcado.

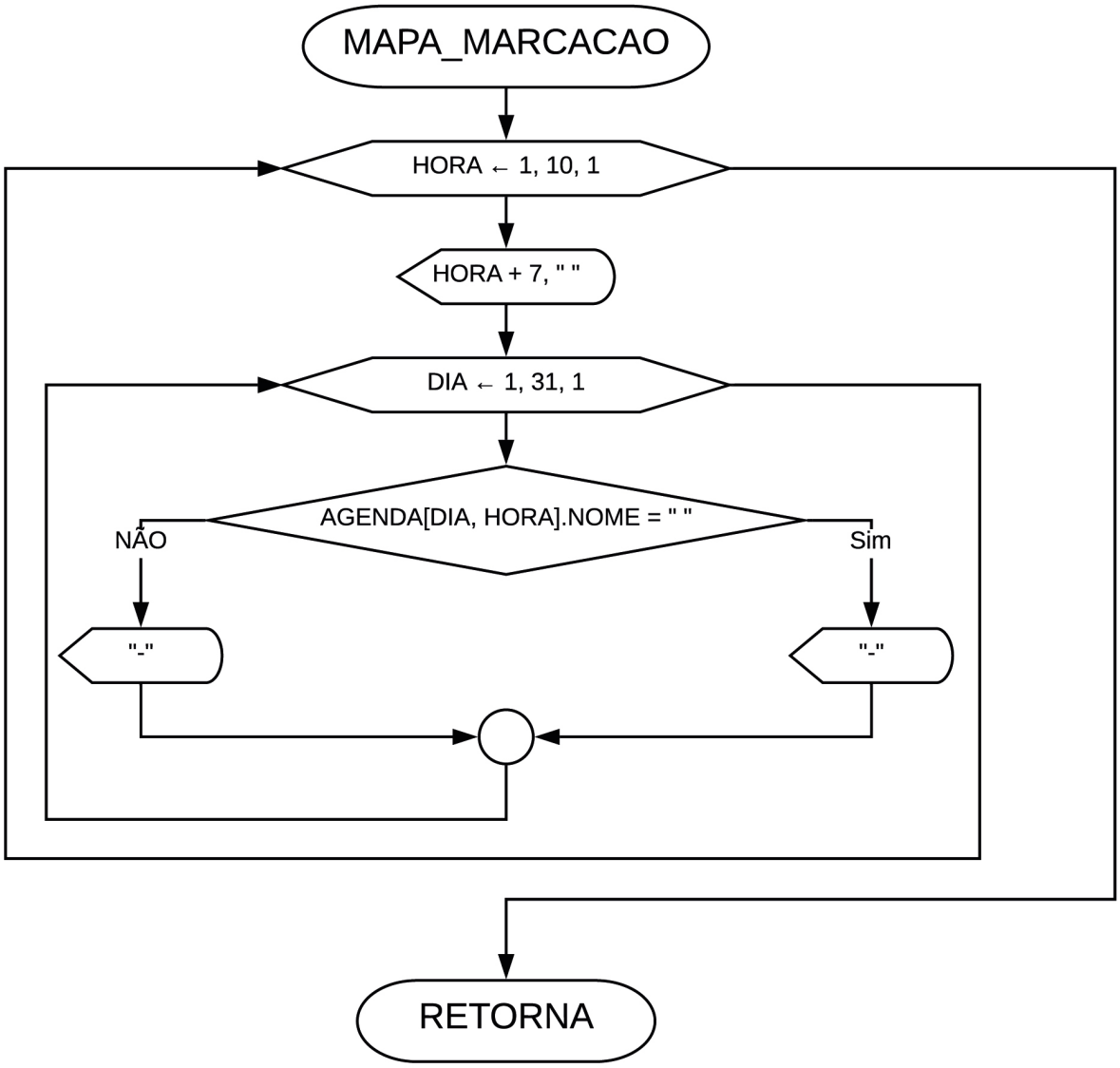

Figura 10 - Rotina de exibição de todos os atendimentos do mês.

2 PROGRAMA

```
/// PROGRAMA AGENDA
/*
  Programa ..................: Agenda de Atendimento
  Autor .....................: Davy P. S. Dantas
  Data ......................: 18 / 05 / 2019
  Versão ....................: v1.0.0
  Contato ...................: davypaulinocbjr@outlook.com
  Linkedin ..................: @davypaulinodsd
  Github ....................: @ThreeDP
  Portifolio ................: davypaulino.myportfolio.com
*/

// INÍCIO: BIBLIOTECAS .............................|
#include <iostream>
#include <iomanip>
#include <fstream>
#include <clocale>
#include <cctype>
#include <windows.h>
// FIM: BIBLIOTECAS ................................|

// INICIO: Constantes de posicionamento do cursor ..|
#define ROD                                       25
#define PADRAO                                     1
#define ESCRIT                                    23
#define LIMPAT                                     4
#define XPAD                                       0
#define CABEC                                      3
#define L_RE                                       8
// FIM: Constantes de posicionamento do cursor .....|

// INÍCIO: Constantes de Cores .....................|
#define PRETO                                      0
#define AZUL                                       9
#define VERDE                                     10
#define VERMEL                                     4
#define BRANCO                                    15
#define AMAREL                                    14
// FIM: Constantes de Cores ........................|

// INÍCIO: Constantes de dimensão TELA .............|
#define COLUN 93
#define LINH 30
// FIM: Constantes de dimensão TELA ................|

// INÍCIO: Constantes de reset .....................|
#define VAZIO "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
```

```
// Constante para verificação de vazio
#define ESPACO "                                                                "
// Constante para limpar espaço coloridos
// FIM: Constantes de reset .....................|

/// Registro AGENDA variável global
using namespace std;

struct CAD_AGENDA
// Registro para armazenar o nome do cliente no dia e horário indicados

  {
    char NOME[41];
  }; CAD_AGENDA AGENDA[31][10];

/// INÍCIO: Biblioteca Programador ======================================================
 void DIMENCIONAR_TELA()
{
    COORD POS;
    POS.X = COLUN;
    POS.Y = LINH;

    SMALL_RECT DIMENCAO;
    DIMENCAO.Top = 0;
    DIMENCAO.Left = 0;
    DIMENCAO.Bottom = LINH - 1;
    // Determina a parte inferior da tela com o valor da constante LINH (LINHA)
    DIMENCAO.Right = COLUN - 1;
    // Determina a direita da tela com o valor da constante COLUN (COLUNA)
    /* - 1 é para que o valor permaneça correto transformando o número em sequencial
    cardinal */
    HANDLE TELA = GetStdHandle(STD_OUTPUT_HANDLE);
    // Determina a variável TELA como indicador de saída no vídeo
    SetConsoleScreenBufferSize(TELA, POS);
    // Muda o tamanho do console
    SetConsoleWindowInfo(TELA, true, &DIMENCAO);
    // Exibi o tamanho do console alterado
}

void LIMPAR_LINHA(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);
```

```
  return;
}

void ATRIBUI_COR(int COR_FUNDO, int COR_TEXTO)
{
  HANDLE TELA;
  int COR;
  TELA = GetStdHandle(STD_OUTPUT_HANDLE);

  if (COR_FUNDO < 0 and COR_FUNDO > 15 and COR_TEXTO < 0 and COR_TEXTO > 15)
    COR = 15;
  else
    COR = COR_TEXTO - 1 + 16 * COR_FUNDO + 1;
  SetConsoleTextAttribute(TELA, COR);
}

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

int POSICIONA_CURSOR(int LINHA ,int COLUNA)
{
  HANDLE TELA;
  COORD POS;
  TELA = GetStdHandle(STD_OUTPUT_HANDLE);
  POS.X = COLUNA - 1;
  POS.Y = LINHA - 1;
  SetConsoleCursorPosition(TELA, POS);
}
/// FIM: Biblioteca Programador ============================================================

/// INICIO: Rotina RESET_MATRIZ ------------------------------------------------------------
void RESET_MATRIZ() // Certifica que o registro AGENDA esteja vazio
{
  int I, J;

  for (I = 1; I <= 31; I++)
  {
    for (J = 1; J <= 10; J++)
    {
      strcpy(AGENDA[I][J].NOME, VAZIO); // Adiciona caracteres nulos ao registro AGENDA
    }
  }
```

```
  return;
}
/// FIM: Rotina RESET_MATRIZ ------------------------------------------------------------

/// INICIO: Rotina Entrar DIA -----------------------------------------------------------
int ENTRA_DIA(int *DIA)          // Função de Entrada e verificação do dado dia
{
  bool I = false;          // Variável para verificação da validade do número inserido
  char C_DIA[10];
  // Char com o número de caracteres numéricos possíveis de entrada sem ocorrer um erro

  cout << " Informe o DIA .....: ";

  do
  {
    cin.getline(C_DIA, sizeof(C_DIA));
    // Realiza a entrada do DIA em Char para evitar Entrada de caracteres especiais
    *DIA = atoi(C_DIA);
    // Converte o valor de char para inteiro e atribui a variável DIA;
    if (*DIA <= 0 or *DIA > 31)
    // Verifica se o número convertido se encontra na faixa válida de dias
      {
        POSICIONA_CURSOR(ROD, PADRAO);
        // Posiciona Cursor na 1° linha e na 2° coluna
        ATRIBUI_COR(VERMEL, BRANCO);
        // Atribui cor vermelha ao fundo e branca para a letra

        cout << " Dia Inválido " << endl;

        ATRIBUI_COR(PRETO, BRANCO);
        // Restaura cor de fundo e texto para o padrão
        POSICIONA_CURSOR(L_RE, ESCRIT);
        // Posiciona o cursor na 8° linha e em frente ao texto "Informe o DIA"
        LIMPAR_LINHA();
        // Limpa os caracteres a partir da localização do cursor
      }
    else
      {
        I = true;
        POSICIONA_CURSOR(ROD, PADRAO);    // Posiciona o Cursor no rodapé
        cout << ESPACO<< endl;            // Insere caracteres vazios
        POSICIONA_CURSOR(L_RE + 1, PADRAO);
        // Volta o cursor para a linha posterior ao texto "Informe o DIA"
      }
  }
  while (not(I == true));
}
/// FIM: Rotina Entrar DIA --------------------------------------------------------------

/// INICIO: Rotina Entrar HORA ----------------------------------------------------------
```

```
int ENTRA_HORA(int *HORA)
{
  bool I = false;
  char C_HORA[10];
// Char com o número de caracteres numéricos possíveis de se entrar sem ocorrer um erro.

  cout << " Informe a HORA ....: ";

  do
  {
    cin.getline(C_HORA, sizeof(C_HORA));       // Realiza a entrada do HORA
    *HORA = atoi(C_HORA);
    // Converte o valor de HORA para inteiro e atribui a variável HORA

    if (*HORA < 8 or *HORA > 17)               // Verifica a entrada de hora
    {
      POSICIONA_CURSOR(ROD, PADRAO);           // Posiciona o Cursor no rodapé
      ATRIBUI_COR(VERMEL, BRANCO);
      // Atribui cor vermelha ao fundo e branca para a letra
      cout << " Hora Inválida " << endl;
      ATRIBUI_COR(PRETO, BRANCO);              // Volta as cores ao padrão
      POSICIONA_CURSOR(L_RE + 1, ESCRIT);
      // Posiciona o cursor frente ao texto " Informe a Hora
      LIMPAR_LINHA();               // Limpa a linha onde se encontra o cursor
    }
    else
    {
      I = true;
      POSICIONA_CURSOR(ROD, PADRAO);           // Posiciona o cursor na linha de rodapé
      cout << ESPACO << endl;                  // Insere caracteres em branco
      POSICIONA_CURSOR(L_RE + 2, PADRAO);
      // Volta o cursor para a linha posterior ao texto "Informe a HORA"
    }
  }
  while (not(I == true));
}
/// FIM: Rotina Entrar HORA ------------------------------------------------------------

/// INÍCIO: Rotina Marcar atendimento ------------------------------------------------
int CADASTRO(int *SELETOR)
{
  int DIA, HORA;
  char I, NOME[41];
  bool VALIDO = false;
  short RETORNO;

  ofstream ARQUIVO("AGENDAMENTO.DDP", ios::binary);
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);   // Posiciona Cursor abaixo da linha de cabeçalho
```

```
cout << "\n" << setw(58) << "MARCAR ATENDIMENTO\n\n\n";

ENTRA_DIA(&DIA);
ENTRA_HORA(&HORA);


if (strcmp(AGENDA[DIA][HORA-7].NOME, VAZIO) == 0)
// Verifica se o dia e hora escolhidos estão disponíveis
{
  cout << " Nome do cliente ...: ";
  do
  {
    LIMPAR_LINHA();
    cin.getline(NOME, sizeof(NOME));
    RETORNO = isalpha(NOME[1]);
    // verifica se o primeiro caractere da cadeia não é um caractere vazio ou especial
    if (RETORNO != 0)
    // Caso não seja um caractere vazio ou especial cadastra com sucesso.
    {
      strcpy(AGENDA[DIA][HORA - 7].NOME, NOME);
      POSICIONA_CURSOR(ROD, PADRAO);
      ATRIBUI_COR(VERDE, BRANCO);
      cout << " Sucesso: Horário Agendado!" << endl;
      ATRIBUI_COR(PRETO, BRANCO);    // Restaura cor de fundo e texto para o padrão
      VALIDO = true;
    }
    else
    {
      POSICIONA_CURSOR(ROD, PADRAO);
      ATRIBUI_COR(AZUL, BRANCO);
      cout << " Nome Inválido";
      ATRIBUI_COR(PRETO, BRANCO);

      cout << " (Entre somente valores alfanuméricos de A-Z ou a-z)" << endl;
      cout << " Deseja Entrar novamente? (S/N) ";
      cin >> I;
      cin.ignore(80, '\n');

      VALIDO = (toupper(I) == 'S') ? false : true;
      POSICIONA_CURSOR(ROD, PADRAO);
      cout << ESPACO;
    }
    LIMPAR_LINHA();
    POSICIONA_CURSOR(ROD + 1, PADRAO);
    LIMPAR_LINHA();
    POSICIONA_CURSOR(L_RE + 2, ESCRIT);

  }while (VALIDO == false);
}
else
```

```
  {
    POSICIONA_CURSOR(ROD, PADRAO);
    ATRIBUI_COR(VERMEL, BRANCO
    cout << " Horário Ocupado!" << endl;
    ATRIBUI_COR(PRETO, BRANCO);
  }
  ARQUIVO.write(reinterpret_cast<char*>(&AGENDA), sizeof(AGENDA));
  ARQUIVO.flush();
  ARQUIVO.close();
  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
  cout << " Tecle <Enter> para ir ao menu\n\n";
  *SELETOR = 0;       // Atribui Zero ao SELETOR para resetar seleção do swicth case
  cin.get();

  ATRIBUI_COR(PRETO, BRANCO);
  POSICIONA_CURSOR(ROD, PADRAO);
  cout << ESPACO;
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);
}
/// FIM: Rotina Marcar atendimento ------------------------------------------------------

/// INÍCIO: Rotina Desmarcar atendimento ------------------------------------------------
int DESMARCAR(int *SELETOR)
{
  ofstream ARQUIVO("AGENDAMENTO.DDP", ios::binary);
  // Abre o arquivo Agendamento como modo de Escrita
  int DIA, HORA;
  char I;

  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);

  cout << "\n" << setw(59) << "DESMARCAR ATENDIMENTO\n\n\n";

  ENTRA_DIA(&DIA);
  ENTRA_HORA(&HORA);

  if (strcmp(AGENDA[DIA][HORA - 7].NOME, VAZIO) == 0)  // Executa as ações caso a posição em
AGENDA já esteja vazia;
  {
    POSICIONA_CURSOR(ROD, PADRAO);
    ATRIBUI_COR(VERDE, BRANCO);
    cout << "Este Horário já está liberado \n";
    ATRIBUI_COR(PRETO, BRANCO);
  }
  else
  {
    cout << "Horário Agendado para " << AGENDA[DIA][HORA - 7].NOME;
    cout << "\n" << "DESMARCAR? (S/N): ";
```

```
    cin >> I;
    cin.ignore(80, '\n');
    if (toupper(I) == 'S')
    {
      strcpy(AGENDA[DIA][HORA - 7].NOME, VAZIO);
      POSICIONA_CURSOR(ROD, PADRAO);
      ATRIBUI_COR(VERDE, BRANCO);
      cout << "Horário liberado";
    }
  }

  ARQUIVO.write(reinterpret_cast<char*>(&AGENDA), sizeof(AGENDA));
  // Grava em AGENDAMENTO.DDP o Registro AGENDA
  ARQUIVO.flush();
  // Certifica que os dados foram realmente gravados em AGENDAMENTO.DDP
  ARQUIVO.close();
// Fecha o arquivo AGENDAMENTO.DDP

  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
  cout << "Tecle <Enter> para ir ao menu" << "\n";
  *SELETOR = 0;
  cin.get();

  ATRIBUI_COR(PRETO, BRANCO);
  POSICIONA_CURSOR(ROD, PADRAO);
  cout << ESPACO;
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);
}
/// FIM: Rotina Desmarcar atendimento ------------------------------------------------

/// INÍCIO: Rotina Listar clientes de um dia ------------------------------------------
int LISTAR_DIA(int *SELETOR)
{
  int DIA, HORA;
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);

  cout << "\n" << setw(54) << "CLIENTES DO DIA" << "\n\n\n";
  ENTRA_DIA(&DIA);
  cout << endl;

  for (HORA = 1; HORA <= 10; HORA++)
  {
    cout << setw(8) << HORA + 7 << " Horas: " << AGENDA[DIA][HORA].NOME << endl;
  }

  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
```

```
  cout << "Tecle <Enter> para ir ao menu";
  ATRIBUI_COR(PRETO, BRANCO);
  *SELETOR = 0;
  cin.get();

  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);
}
/// FIM: Rotina Listar clientes de um dia ------------------------------------------

/// INÍCIO: Rotina Marcações de um cliente -----------------------------------------
int LISTAR_MARC_CLIENTE(int *SELETOR)
{
  int DIA, HORA;
  char NOME[41];

  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);

  cout << endl << setw(58) << "HORÁRIOS DE UM CLIENTE" << "\n\n\n";
  cout << " Entre o nome do cliente: ";
  cin.getline(NOME, sizeof(NOME));
  // Entra Nome para comparar como AGENDA.NOME

  if (strcmp(NOME, VAZIO) != 0)
  // Executa ação se o Nome inserido for diferente de vazio
  {
    cout << "\n" << " Horários Agendados para:  " << NOME << "\n\n";
    for (DIA = 1; DIA <= 31; DIA++)
    {
      for (HORA = 1; HORA <= 10; HORA++)
      {
        if (strcmp(AGENDA[DIA][HORA].NOME, NOME) == 0)
          cout << "DIA " << DIA << " as " << HORA + 7 << " Horas\n";
      }
    }
  }
  else
  {
    POSICIONA_CURSOR(ROD, PADRAO);
    ATRIBUI_COR(VERMEL, BRANCO);
    cout << " É preciso entrar um nome para exibir!!";
    ATRIBUI_COR(PRETO, BRANCO);
  }

  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
  cout << "Tecle <Enter> para ir ao menu";
  ATRIBUI_COR(PRETO, BRANCO);
  *SELETOR = 0;
```

```
  cin.get();

  POSICIONA_CURSOR(ROD, PADRAO);
  cout << ESPACO;
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);
}
/// FIM: Rotina Marcações de um cliente ------------------------------------------------

/// INÍCIO: Rotina Mapa dos dias Marcados ----------------------------------------------
int MAPA_MARCACAO(int *SELETOR)
{
  int DIA, HORA;
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);

  /* Configuração de apresentação da Agenda; */
  cout << endl << setw(60) << "MAPA DOS HORÁRIOS OCUPADOS" << "\n\n\n";
  cout << setw(31) << " HORA" << setw(11) << "1" << setw(10) << "2" << setw(10) << "3" <<
setw(10) << endl;
  cout << setw(63) << "123456789012345678901234567890l" << endl;

  for(HORA = 1; HORA <= 10; HORA++)
  {
    cout << setw(31) << HORA + 7 << " ";
    for(DIA = 1; DIA <= 31; DIA++)
    {
      (strcmp(AGENDA[DIA][HORA].NOME, VAZIO) == 0) ? cout << "-" : cout << "*";
    }
    cout << endl;
  }

  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
  cout << "Tecle <Enter> para ir ao menu";
  ATRIBUI_COR(PRETO, BRANCO);
  *SELETOR = 0;
  cin.get();
  LIMPAR_TELA(XPAD, CABEC);
  POSICIONA_CURSOR(LIMPAT, PADRAO);
}
/// FIM: Rotina Mapa dos dias Marcados -------------------------------------------------

/// INÍCIO: Rotina Principal do Programa -----------------------------------------------
int main(void)
{
  DIMENCIONAR_TELA();                  // Ajusta as proporções da tela do programa
  RESET_MATRIZ();

  int SELETOR;
  ifstream ARQUIVO("AGENDAMENTO.DDP", ios::binary);
```

```
  // Define Variável ARQUIVO como variável de acesso a AGENDAMENTO.DPP e abre o arquivo

  cout << setlocale(LC_ALL, "") << "\n";
  // Habilita o idioma padrão do computador (programa configurado para linguagem PT_BR)
  cout << setiosflags(ios::right);
  cout << setiosflags(ios::uppercase);
  LIMPAR_TELA(0, 0);
  POSICIONA_CURSOR(0, 0);

  ARQUIVO.read(reinterpret_cast<char*>(&AGENDA), sizeof(AGENDA));
  // Le todos os dados do arquivo e grava na matriz agenda
  ARQUIVO.close();     // Fecha o arquivo
  ATRIBUI_COR(PRETO, AZUL);
  cout << setw(69) << "*** AGENDA PARA MARCAÇÂO DE ATENDIMENTO ***" << "\n\n";
  ATRIBUI_COR(PRETO, BRANCO);

  while (not(SELETOR == 6))
  {
    cout << endl << setw(54) << "MENU PRINCIPAL" << "\n\n";
    cout << setw(30) << " " << "[1] Marcar Atendimento" << "\n";
    cout << setw(30) << " " << "[2] Desmarcar Atendimento" << "\n";
    cout << setw(30) << " " << "[3] Listar Marcações do Dia" << "\n";
    cout << setw(30) << " " << "[4] Clientes Marcados no Dia" << "\n";
    cout << setw(30) << " " << "[5] Mapa dos Horários Livres" << "\n";
    cout << setw(30) << " " << "[6] Fim do Programa" << "\n\n";

    cout << setw(30) << " " << "==> ";
    cin >> SELETOR;
    cin.ignore(80, '\n');
    if (not(SELETOR == 6))
    {
       switch (SELETOR)
       {
        case 1: CADASTRO(&SELETOR); break;
        case 2: DESMARCAR(&SELETOR); break;
        case 3: LISTAR_DIA(&SELETOR); break;
        case 4: LISTAR_MARC_CLIENTE(&SELETOR); break;
        case 5: MAPA_MARCACAO(&SELETOR); break;
        case 6: break;
        default: break;
       }
    }
  }
  POSICIONA_CURSOR(ROD + 1, PADRAO);
  ATRIBUI_COR(PRETO, AZUL);
  cout << "Tecle <Enter> para ir ao menu";
  cin.get();
  return 0;
}
/// FIM: Rotina Principal do Programa ----------------------------------------------------
```

3 CONCLUSÕES FINAIS

O projeto foi concluído com êxito comprido todas as funcionalidades exigidas, alguns ajustes foram realizados pensando na melhor usabilidade para o usuário, muitas mudanças podem ser feitas para um melhor atendimento do usuário e podem ser contempladas futuramente em uma atualização de versão do software.

REFERÊNCIAS

MANZANO, JOSÉ AUGUSTO N. G. **C++ Underground programmer:** 6, ed. São Paulo: Propes Vivens/Clube de Autores, 2018.

MANZANO, JOSÉ AUGUSTO N. G. **Algoritmos - Lógica para desenvolvimento de programação de computadores:** José Augusto N. G. Manzano, Jayr Figueiredo de Oliveira – 28, ed. – São Paulo : Érica, 2016.

SetConsoleScreenBufferSize function: Microsoft Docs, 07 de nov. 2018, Disponível em: https://docs.microsoft.com/en-us/windows/console/setconsolescreenbuffersize. Acesso em: 18 jun. 2019.

SetConsoleWindowInfo function: Microsoft Docs, 07 de nov. 2018, Disponível em: <https://docs.microsoft.com/en-us/windows/console/setconsolewindowinfo>. Acesso em: 18 jun. 2019.

SMALL_RECT structure: Microsoft Docs, 07 de nov. 2018, Disponível em: < https://docs.microsoft.com/en-us/windows/console/small-rect-str>. Acesso em: 18 jun. 2019.