UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL ALTHAUS CAPRI CASTELO BRANCO LISBOA

ESPECIFICAÇÃO DE REQUISITOS:
SISTEMA DE LOCAÇÃO DE VEÍCULOS

TRABALHO DE ENGENHARIA DE SOFTWARE

PONTA GROSSA

2017

RAFAEL ALTHAUS CAPRI CASTELO BRANCO LISBOA

CASOS DE TESTES:

SISTEMA DE LOCAÇÃO DE VEÍCULOS

Trabalho apresentado como requisito parcial à obtenção da nota da disciplina de Engenharia de Software, do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – Campus Ponta Grossa.

Prof. MSc. Vinícius Camargo Andrade.

PONTA GROSSA

2017

SUMÁRIO

1 **TESTES DE CAIXA BRANCA 4**
1.1 FUNÇÃO COUNTLEAPYEARS 4
1.2 FUNÇÃO CHARTOINT 5
1.3 FUNÇÃO GETDIFFERENCEDAYS 6
1.4 FUNÇÃO IMPRIMECABECALHO 7
1.5 FUNÇÃO GERARID 7
1.6 FUNÇÃO LISTARCLIENTES 8
1.7 FUNÇÃO CADASTRARCLIENTE 9
1.8 FUNÇÃO CLIENTES 10
1.9 FUNÇÃO LISTARVEICULOS 11
1.10 FUNÇÃO CADASTRARVEICULO 12
1.11 FUNÇÃO VEICULOS 13
1.12 FUNÇÃO PROCURARCLIENTE 14
1.13 FUNÇÃO PROCURARVEICULO 15
1.14 FUNÇÃO LOCAR 17
1.15 FUNÇÃO PROCURARLOCACAO 18
1.16 FUNÇÃO GETVALORVEICULO 19
1.17 FUNÇÃO INICIO 20
1.18 FUNÇÃO DEVOLVER 21
1.19 FUNÇÃO MAIN 24

1 TESTES DE CAIXA-BRANCA

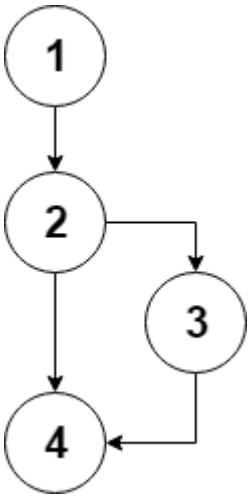
1.1 FUNÇÃO COUNTLEAPYEARS

V(G): 4 – 4 + 2 = 2

Caminho 1: 1 – 2 – 3 – 4

Caminho 2: 1 – 2 – 4

```
70  int countLeapYears(Date d) {
71      int years = d.y;
72
73      if (d.m <= 2)
74          years--;
75
76      return years / 4 - years / 100 + years / 400;
77  }
```



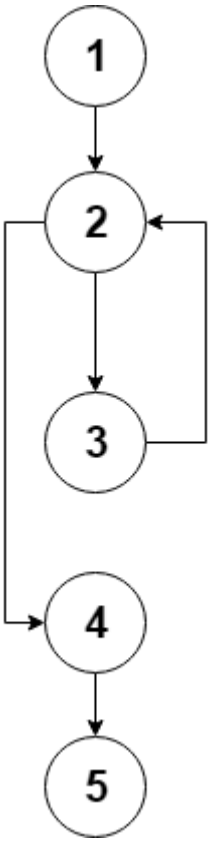
1.2 FUNÇÃO CHARTOINT

V(G): 5 – 5 + 2 = 2

Caminho 1: 1 – 2 – 3 – 2 – 4 – 5

Caminho 2: 1 – 2 – 4 – 5

```
79  int charToInt(char data[], int qtd, int step) {
80      char dia[50];
81      for (int i = 0; i < 3; i++) {
82          dia[i] = "";
83      }
84      strncpy(dia, &data[step], qtd);
85      return atoi(dia);
86  }
```



1.3 Função getDifferenceDays

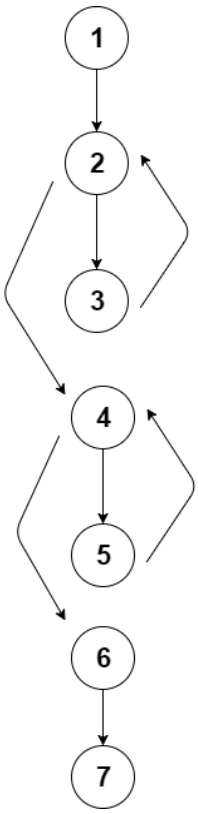
V(G): 8 – 7 + 2 = 3

Caminho 1: 1 – 2 – 3 – 2 – 4 – 5 – 4 – 6 – 7

Caminho 2: 1 – 2 – 4 – 6 – 7

Caminho 3: 1 – 2 – 3 – 2 – 4 – 6 – 7

```
88  int getDifferenceDays(char data1[], char data2[]) {
89
90      int day1 = charToInt(data1, 2, 0);
91      int mon1 = charToInt(data1, 2, 3);
92      int year1 = charToInt(data1, 4, 6);
93
94      int day2 = charToInt(data2, 2, 0);
95      int mon2 = charToInt(data2, 2, 3);
96      int year2 = charToInt(data2, 4, 6);
97
98      Date dt1 = {day1, mon1, year1};
99      Date dt2 = {day2, mon2, year2};
100
101     long int n1 = dt1.y * 365 + dt1.d;
102
103     for (int i = 0; i < dt1.m - 1; i++)
104         n1 += monthDays[i];
105
106     n1 += countLeapYears(dt1);
107
108     long int n2 = dt2.y * 365 + dt2.d;
109     for (int i = 0; i < dt2.m - 1; i++)
110         n2 += monthDays[i];
111     n2 += countLeapYears(dt2);
112
113     return (n2 - n1);
114 }
```

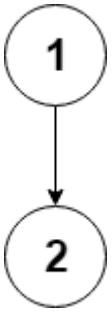


1.4 Função imprimeCabecalho

V(G): 1 – 2 + 2 = 1

Caminho 1: 1 – 2

```
116  void imprimeCabecalho(char *titulo) {
117      system("cls||clear");
118      printf("\n----------------------------------------
119  }
120
```

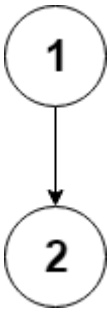


1.5 Função gerarId

V(G) 1 – 2 + 2 = 1

Caminho 1: 1 – 2

```
121  int gerarId() {
122      srand(time(NULL));
123      return rand() % (ID_MAX - ID_MIN + 1) + ID_MIN;
124  }
125
```



1.6 Função listarClientes

V(G): 16 – 13 + 2 = 5

Caminho 1: 1 – 2 – 4 – 5 – 8 – 9 – 12 – 13

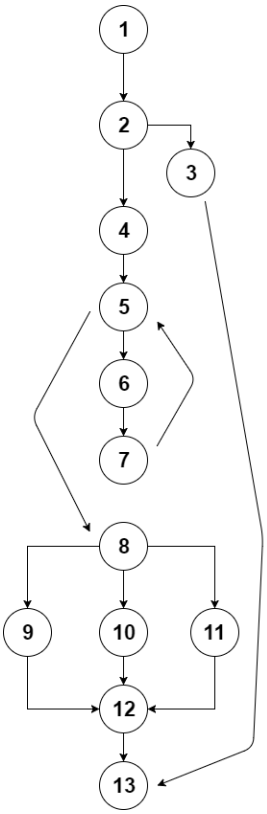
Caminho 2: 1 – 2 – 4 – 5 – 8 – 10 – 12 – 13

Caminho 3: 1 – 2 – 4 – 5 – 8 – 11 – 12 – 13

Caminho 4: 1 – 2 – 4 – 5 – 6 – 7 – 5 ...

Caminho 5: 1 – 2 – 3 – 13

```c
void listarClientes() {
    Cliente c;
    FILE *infile;
    int opcao;

    imprimeCabecalho("LISTA DE CLIENTES");

    infile = fopen(FILE_CLIENTE, "r");
    if (infile == NULL) {
        fprintf(stderr, MSG_ERRO_ARQUIVO);
        exit(1);
    }

    printf("    ID %9s %13s", "Nasc.", "Nome\n\n");
    while (fread(&c, sizeof (Cliente), 1, infile)) {
        printf("    %d   %s   %s\n", c.cliId, c.dataNascimento, c.nome);
    }

    fclose(infile);

    printf("\nSelecione uma opção:\n1. Listar novamente\n2. Voltar\n\nOPÇÃO: ");
    scanf("%d", &opcao);

    switch (opcao) {
        case 1:
            listarClientes();
            break;
        case 2:
            clientes();
            break;
        default:
            printf("\nOpção inválida.\n");
            listarClientes();
            break;
    }
}
```



1.7 Função cadastrarCliente

V(G): 11 – 9 – 2 = 4

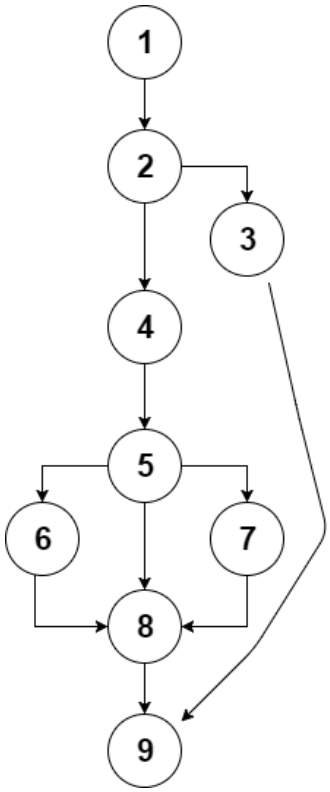
Caminho 1: 1 – 2 – 4 – 5 – 6 – 8 – 9

Caminho 2: 1 – 2 – 4 – 5 – 8 – 9

Caminho 3: 1 – 2 – 4 – 5 – 7 – 8 – 9

Caminho 4: 1 – 2 – 3 – 9

```c
void cadastrarCliente() {
    Cliente c;
    FILE *outfile;

    imprimeCabecalho("CADASTRAR CLIENTE");

    printf("Nome: ");
    scanf(" %[^\n]", &c.nome);

    printf("Data de Nascimento (dd/mm/yyyy): ");
    scanf(" %[^\n]", &c.dataNascimento);

    printf("RG: ");
    scanf(" %[^\n]", &c.rg);

    printf("CPF: ");
    scanf(" %[^\n]", &c.cpf);

    printf("Nacionalidade (Ex.: BR): ");
    scanf(" %[^\n]", &c.nacionalidade);

    printf("Nº Habilitação: ");
    scanf(" %[^\n]", &c.habilitacao);

    printf("Validade CNH (dd/mm/yyyy): ");
    scanf(" %[^\n]", &c.validadeCnh);

    outfile = fopen(FILE_CLIENTE, "a");

    if (outfile == NULL) {
        fprintf(stderr, MSG_ERRO_ARQUIVO);
        exit(1);
    }

    c.cliId = gerarId();
    fwrite(&c, sizeof (Cliente), 1, outfile);

    if (fwrite != 0) {
        printf("\nCliente cadastrado com sucesso! Aguarde...\n");
    } else {
        printf("\nErro ao cadastrar o cliente. Reinicie o programa e tente novamente.\n");
    }

    fclose(outfile);

    sleep(2);
    clientes();

}
```



1.8 Função clientes

V(G): 10 – 8 + 2 = 4

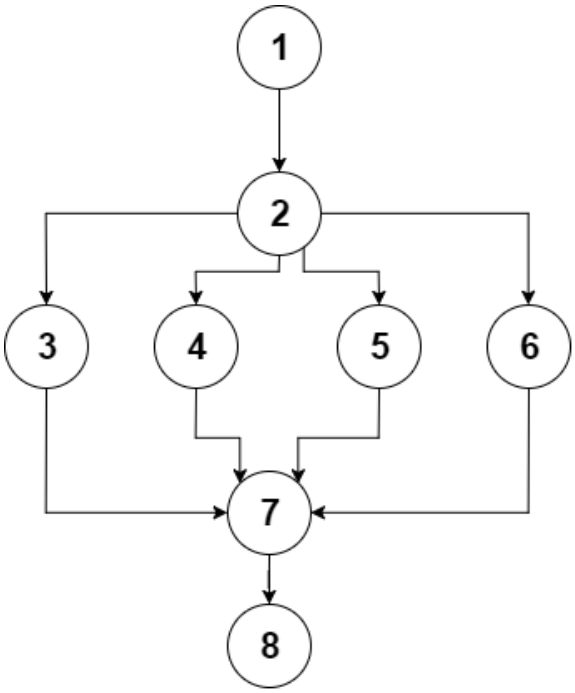
Caminho 1: 1 – 2 – 3 – 7 – 8

Caminho 2: 1 – 2 – 4 – 7 – 8

Caminho 3: 1 – 2 – 5 – 7 – 8

Caminho 4: 1 – 2 – 6 – 7 – 8

```
213  void clientes() {
214      int opcao;
215
216      imprimeCabecalho("CADASTRO DE CLIENTES");
217      printf("Selecione uma opção:\n1. Listar clientes\n2. Cadastrar um cliente\n3. Voltar\n\nOPÇÃO: ");
218      scanf("%d", &opcao);
219
220      switch (opcao) {
221          case 1:
222              listarClientes();
223              break;
224          case 2:
225              cadastrarCliente();
226              break;
227          case 3:
228              inicio();
229              break;
230          default:
231              printf("\nOpção inválida.\n");
232              clientes();
233              break;
234      }
235  }
236
```



1.9 Função listarVeiculos

V(G): 16 – 13 + 2 = 5

Caminho 1: 1 – 2 – 4 – 5 – 8 – 9 – 12 – 13

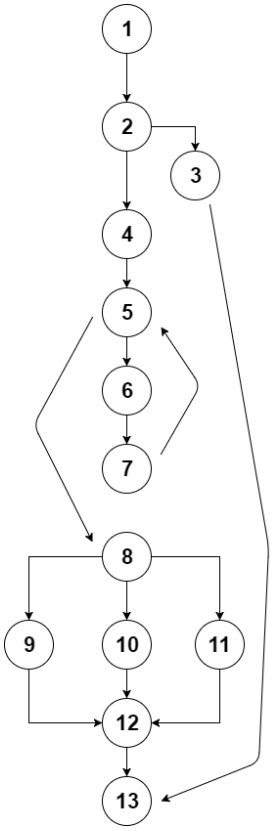
Caminho 2: 1 – 2 – 4 – 5 – 8 – 10 – 12 – 13

Caminho 3: 1 – 2 – 4 – 5 – 8 – 11 – 12 – 13

Caminho 4: 1 – 2 – 4 – 5 – 6 – 7 – 5 ...

Caminho 5: 1 – 2 – 3 – 13

```c
void listarVeiculos() {
    Veiculo c;
    FILE *infile;
    int opcao;

    imprimeCabecalho("LISTA DE VEÍCULOS");

    infile = fopen(FILE_VEICULO, "r");
    if (infile == NULL) {
        fprintf(stderr, MSG_ERRO_ARQUIVO);
        exit(1);
    }

    printf("    ID %9s %11s", "Placa", "Nome\n\n");
    while (fread(&c, sizeof (Veiculo), 1, infile)) {
        printf("    %d   %s   %s - %s\n", c.veiId, c.placa, c.marca, c.modelo);
    }

    fclose(infile);

    printf("\nSelecione uma opção:\n1. Listar novamente\n2. Voltar\n\nOPÇÃO: ");
    scanf("%d", &opcao);

    switch (opcao) {
        case 1:
            listarVeiculos();
            break;
        case 2:
            veiculos();
            break;
        default:
            printf("\nOpção inválida.\n");
            listarVeiculos();
            break;
    }
}
```



1.10 Função cadastrarVeiculo

V(G): 11 – 9 + 2 = 4

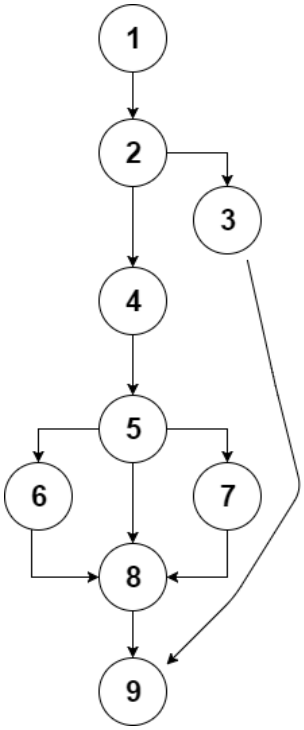
Caminho 1: 1 – 2 – 4 – 5 – 6 – 8 – 9

Caminho 2: 1 – 2 – 4 – 5 – 8 – 9

Caminho 3: 1 – 2 – 4 – 5 – 7 – 8 – 9

Caminho 4: 1 – 2 – 3 – 9

```c
void cadastrarVeiculo() {
    Veiculo v;
    FILE *outfile;

    imprimeCabecalho("CADASTRAR VEÍCULO");

    printf("Marca: ");
    scanf(" %[^\n]", &v.marca);

    printf("Modelo: ");
    scanf(" %[^\n]", &v.modelo);

    printf("Ano: ");
    scanf(" %[^\n]", &v.ano);

    printf("Cor: ");
    scanf(" %[^\n]", &v.cor);

    printf("Combustível (A)lcool | (G)asolina | (F)lex: ");
    scanf(" %[^\n]", &v.combustivel);

    printf("Placa: ");
    scanf(" %[^\n]", &v.placa);

    printf("Renavan: ");
    scanf(" %[^\n]", &v.renavan);

    printf("Categoria (1) Econômico | (2) Intermediário | (3) Luxo: ");
    scanf("%d", &v.categoria);

    outfile = fopen(FILE_VEICULO, "a");

    if (outfile == NULL) {
        fprintf(stderr, MSG_ERRO_ARQUIVO);
        exit(1);
    }

    v.veiId = gerarId();
    fwrite(&v, sizeof (Veiculo), 1, outfile);

    if (fwrite != 0) {
        printf("\nVeículo cadastrado com sucesso! Aguarde...\n");
    } else {
        printf("\nErro ao cadastrar o veículo. Reinicie o programa e tente novamente.\n");
    }

    fclose(outfile);

    sleep(2);
    veiculos();

}
```



1.11 Função veículos

V(G): 10 – 8 + 2 = 4

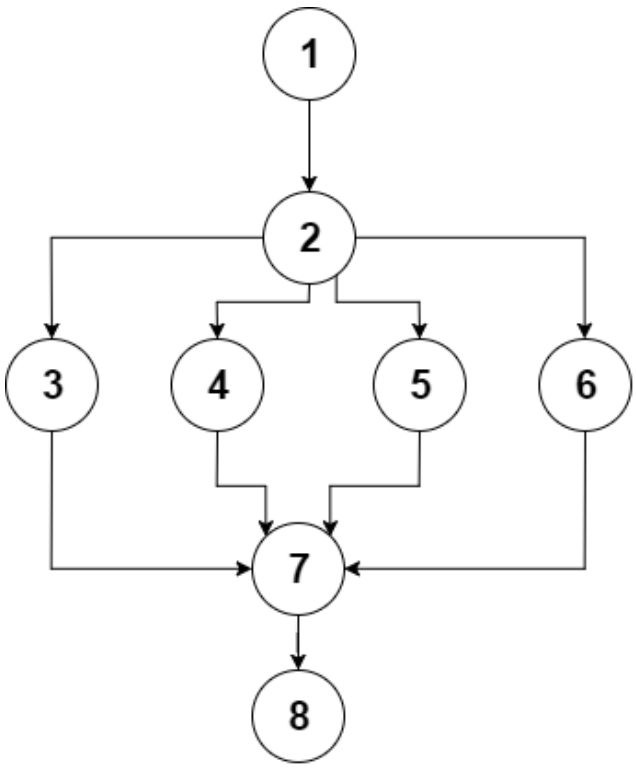
Caminho 1: 1 – 2 – 3 – 7 – 8

Caminho 2: 1 – 2 – 4 – 7 – 8

Caminho 3: 1 – 2 – 5 – 7 – 8

Caminho 4: 1 – 2 – 6 – 7 – 8

```
327 void veiculos() {
328     int opcao;
329 
330     imprimeCabecalho("CADASTRO DE VEÍCULOS");
331     printf("Selecione uma opção:\n1. Listar veículos\n2. Cadastrar um veículo\n3. Voltar\n\nOPÇÃO: ");
332     scanf("%d", &opcao);
333 
334     switch (opcao) {
335         case 1:
336             listarVeiculos();
337             break;
338         case 2:
339             cadastrarVeiculo();
340             break;
341         case 3:
342             inicio();
343             break;
344         default:
345             printf("\nOpção inválida.\n");
346             veiculos();
347             break;
348     }
349 }
```



1.12 Função procurarCliente

V(G): 19 – 15 + 2 = 6

Caminho 1: 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10 – 6 – 11 – 12 – 13 – 14 – 2 - 15

Caminho 2: 1 – 2 – 3 – 5 – 6 – 7 – 9 – 10 – 6 – 11 – 12 – 13 – 14 – 2 – 15

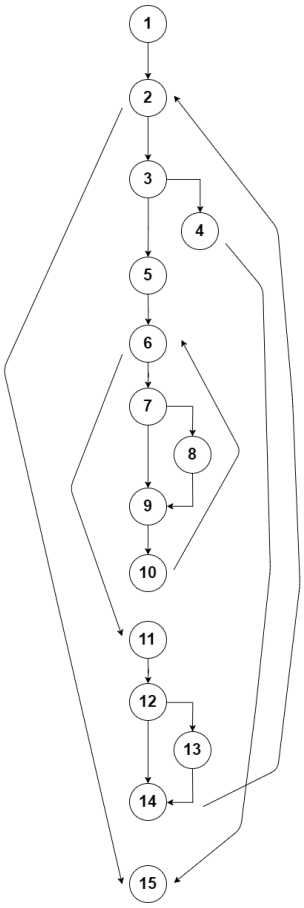
Caminho 3: 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10...

Caminho 4: 1 – 2 – 3 – 5 – 6 – 11 – 12 – 13 – 14

Caminho 5: 1 – 2 – 3 – 4 – 15

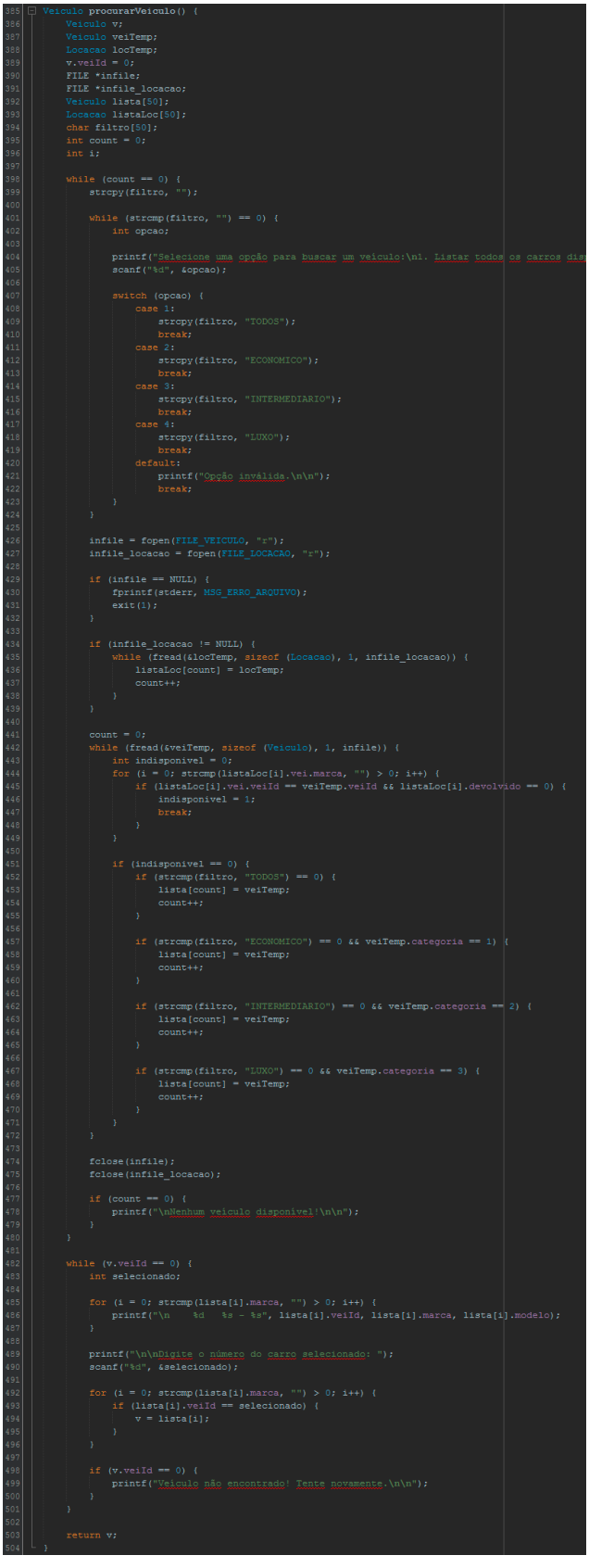
Caminho 6: 1 – 2 – 15

```c
Cliente procurarCliente() {
    Cliente cliTemp;
    Cliente cliente;
    FILE *infile_cliente;
    char cpf[12];

    cliente.cliId = 0;

    while (cliente.cliId == 0) {
        printf("Busque um cliente por CPF: ");
        scanf(" %[^\n]", &cpf);

        infile_cliente = fopen(FILE_CLIENTE, "r");
        if (infile_cliente == NULL) {
            fprintf(stderr, MSG_ERRO_ARQUIVO);
            exit(1);
        }

        while (fread(&cliTemp, sizeof (Cliente), 1, infile_cliente)) {
            if (strcmp(cliTemp.cpf, cpf) == 0) {
                cliente = cliTemp;
            }
        }

        fclose(infile_cliente);

        if (cliente.cliId == 0) {
            printf("Cliente não encontrado! Tente novamente.\n\n");
        }
    }

    return cliente;
}
```

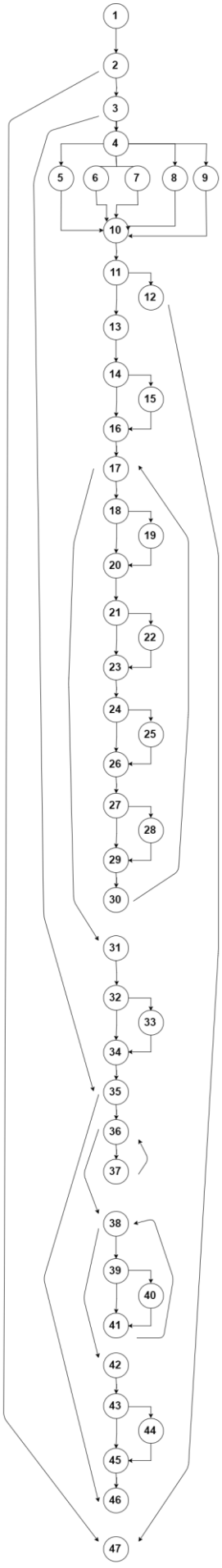


1.13 Função procurarVeiculo

V(G): 64 - 47 + 2 = 19



```
Veiculo procurarVeiculo() {
    Veiculo v;
    Veiculo veiTemp;
    Locacao locTemp;
    v.veiId = 0;
    FILE *infile;
    FILE *infile_locacao;
    Veiculo lista[50];
    Locacao listaLoc[50];
    char filtro[50];
    int count = 0;
    int i;

    while (count == 0) {
        strcpy(filtro, "");

        while (strcmp(filtro, "") == 0) {
            int opcao;

            printf("Selecione uma opção para buscar um veiculo:\n1. Listar todos os carros dis
            scanf("%d", &opcao);

            switch (opcao) {
                case 1:
                    strcpy(filtro, "TODOS");
                    break;
                case 2:
                    strcpy(filtro, "ECONOMICO");
                    break;
                case 3:
                    strcpy(filtro, "INTERMEDIARIO");
                    break;
                case 4:
                    strcpy(filtro, "LUXO");
                    break;
                default:
                    printf("Opção inválida.\n\n");
                    break;
            }
        }

        infile = fopen(FILE_VEICULO, "r");
        infile_locacao = fopen(FILE_LOCACAO, "r");

        if (infile == NULL) {
            fprintf(stderr, MSG_ERRO_ARQUIVO);
            exit(1);
        }

        if (infile_locacao != NULL) {
            while (fread(&locTemp, sizeof (Locacao), 1, infile_locacao)) {
                listaLoc[count] = locTemp;
                count++;
            }
        }

        count = 0;
        while (fread(&veiTemp, sizeof (Veiculo), 1, infile)) {
            int indisponivel = 0;
            for (i = 0; strcmp(listaLoc[i].vei.marca, "") > 0; i++) {
                if (listaLoc[i].vei.veiId == veiTemp.veiId && listaLoc[i].devolvido == 0) {
                    indisponivel = 1;
                    break;
                }
            }

            if (indisponivel == 0) {
                if (strcmp(filtro, "TODOS") == 0) {
                    lista[count] = veiTemp;
                    count++;
                }

                if (strcmp(filtro, "ECONOMICO") == 0 && veiTemp.categoria == 1) {
                    lista[count] = veiTemp;
                    count++;
                }

                if (strcmp(filtro, "INTERMEDIARIO") == 0 && veiTemp.categoria == 2) {
                    lista[count] = veiTemp;
                    count++;
                }

                if (strcmp(filtro, "LUXO") == 0 && veiTemp.categoria == 3) {
                    lista[count] = veiTemp;
                    count++;
                }
            }
        }

        fclose(infile);
        fclose(infile_locacao);

        if (count == 0) {
            printf("\nNenhum veiculo disponivel!\n\n");
        }
    }

    while (v.veiId == 0) {
        int selecionado;

        for (i = 0; strcmp(lista[i].marca, "") > 0; i++) {
            printf("\n    %d   %s - %s", lista[i].veiId, lista[i].marca, lista[i].modelo);
        }

        printf("\n\nDigite o número do carro selecionado: ");
        scanf("%d", &selecionado);

        for (i = 0; strcmp(lista[i].marca, "") > 0; i++) {
            if (lista[i].veiId == selecionado) {
                v = lista[i];
            }
        }

        if (v.veiId == 0) {
            printf("Veiculo não encontrado! Tente novamente.\n\n");
        }
    }

    return v;
}
```


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
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

1.14 Função locar

V(G): 11 – 9 – 2 = 4

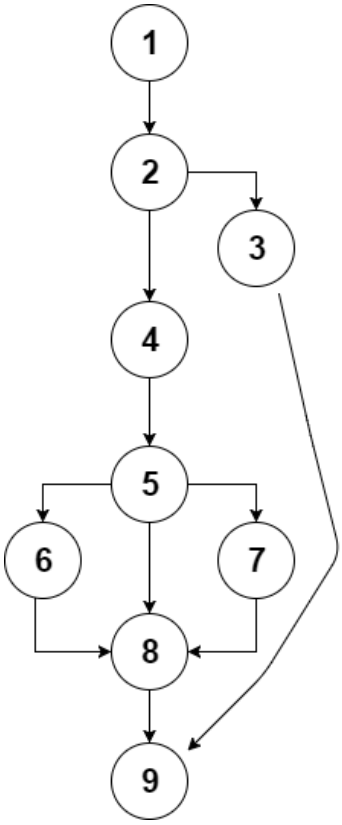
Caminho 1: 1 – 2 – 4 – 5 – 6 – 8 – 9

Caminho 2: 1 – 2 – 4 – 5 – 8 – 9

Caminho 3: 1 – 2 – 4 – 5 – 7 – 8 – 9

Caminho 4: 1 – 2 – 3 – 9

```c
void locar() {
    Locacao l;
    FILE *outfile;

    imprimeCabecalho("LOCAR VEÍCULO");

    l.cli = procurarCliente();
    printf("Cliente %s selecionado!\n\n", l.cli.nome);

    l.vei = procurarVeiculo();
    printf("Veículo %s %s selecionado!\n\n", l.vei.marca, l.vei.modelo);

    printf("Data da locação (dd/mm/yyyy): ");
    scanf(" %[^\n]", &l.dataLocacao);

    printf("Data da devolução (dd/mm/yyyy): ");
    scanf(" %[^\n]", &l.dataDevolucao);

    outfile = fopen(FILE_LOCACAO, "a");

    if (outfile == NULL) {
        fprintf(stderr, MSG_ERRO_ARQUIVO);
        exit(1);
    }

    l.locId = gerarId();
    l.devolvido = 0;

    fwrite(&l, sizeof (Locacao), 1, outfile);

    if (fwrite != 0) {
        printf("\nLocação cadastrada com sucesso! Aguarde...\n");
    } else {
        printf("\nErro ao cadastrar a locação. Reinicie o programa e tente novamente.\
    }

    fclose(outfile);

    sleep(2);
    inicio();
}
```



1.15 Função procurarLocacao

V(G): 19 – 15 + 2 = 6

Caminho 1: 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10 – 6 – 11 – 12 – 13 – 14 – 2 - 15

Caminho 2: 1 – 2 – 3 – 5 – 6 – 7 – 9 – 10 – 6 – 11 – 12 – 13 – 14 – 2 – 15

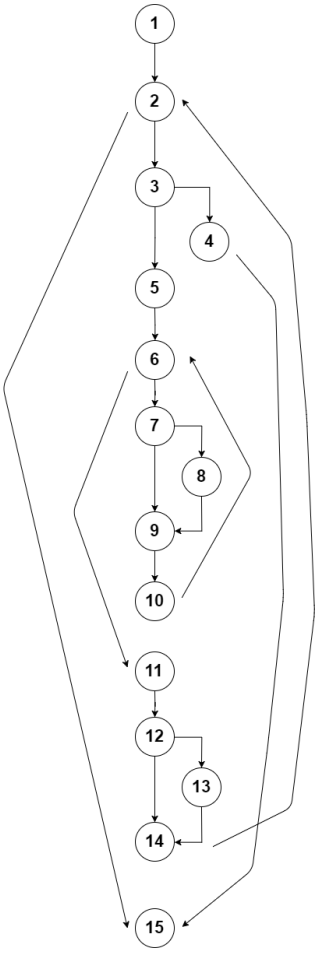
Caminho 3: 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10...

Caminho 4: 1 – 2 – 3 – 5 – 6 – 11 – 12 – 13 – 14

Caminho 5: 1 – 2 – 3 – 4 – 15

Caminho 6: 1 – 2 – 15

```
548  Locacao procurarLocacao() {
549      Locacao locTemp;
550      Locacao locacao;
551      FILE *infile;
552      char placa[10];
553
554      locacao.locId = 0;
555
556      while (locacao.locId == 0) {
557          printf("Digite a placa do veiculo: ");
558          scanf("%s", &placa);
559
560          infile = fopen(FILE_LOCACAO, "r");
561          if (infile == NULL) {
562              fprintf(stderr, MSG_ERRO_ARQUIVO);
563              exit(1);
564          }
565
566          while (fread(&locTemp, sizeof (Locacao), 1, infile)) {
567              if (strcmp(locTemp.vei.placa, placa) == 0 && locTemp.devolvido == 0) {
568                  locacao = locTemp;
569              }
570          }
571
572          fclose(infile);
573
574          if (locacao.locId == 0) {
575              printf("Locação não encontrada! Tente novamente.\n\n");
576          }
577      }
578
579      return locacao;
580  }
```



1.16 Função getValorVeiculo

V(G): 10 – 8 + 2 = 4

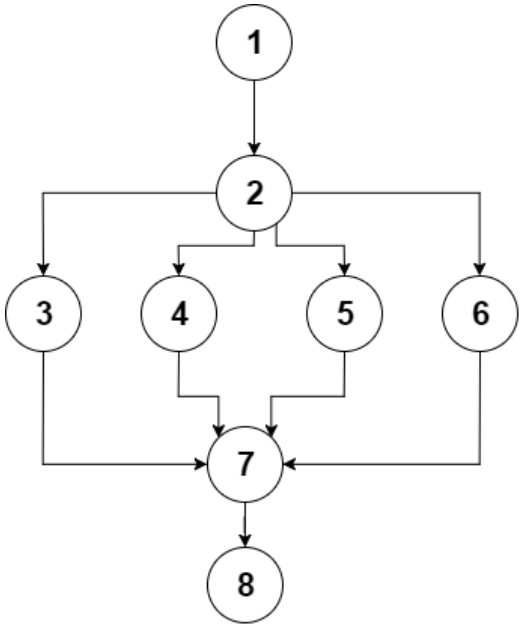
Caminho 1: 1 – 2 – 3 – 7 – 8

Caminho 2: 1 – 2 – 4 – 7 – 8

Caminho 3: 1 – 2 – 5 – 7 – 8

Caminho 4: 1 – 2 – 6 – 7 – 8

```
582  float getValorVeiculo(Veiculo v) {
583      switch (v.categoria) {
584          case 1:
585              return PRECO_ECONOMICO;
586              break;
587          case 2:
588              return PRECO_INTERMEDIARIO;
589              break;
590          case 3:
591              return PRECO_LUXO;
592              break;
593          default:
594              return 0;
595              break;
596      }
597
598      return 0;
599  }
```



1.17 Função inicio

V(G): 14 – 10 + 2 = 6

Caminho 1: 1 – 2 – 3 – 9 – 10

Caminho 2: 1 – 2 – 4 – 9 – 10

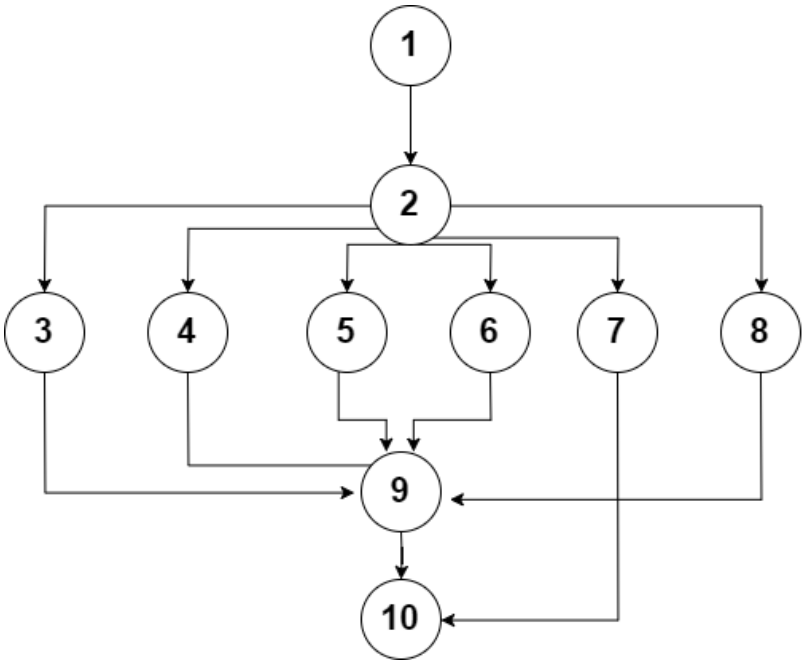
Caminho 3: 1 – 2 – 5 – 9 – 10

Caminho 4: 1 – 2 – 6 – 9 – 10

Caminho 5: 1 – 2 – 7 – 10

Caminho 6: 1 – 2 – 8 – 9 – 10

```
705  void inicio() {
706      int opcao;
707
708      imprimeCabecalho("LOCADORA DE VEÍCULOS");
709      printf("Selecione uma opção:\n1. Cadastro de Clientes\n2. Cadastro de Veículos\n3. Locar\n4. Devolver\n5. Sair\n\nOPÇÃO: ");
710      scanf("%d", &opcao);
711
712      switch (opcao) {
713          case 1:
714              clientes();
715              break;
716          case 2:
717              veiculos();
718              break;
719          case 3:
720              locar();
721              break;
722          case 4:
723              devolver();
724              break;
725          case 5:
726              exit(1);
727              break;
728          default:
729              printf("\nOpção inválida.");
730              inicio();
731              break;
732      }
733  }
```



1.18 Função devolver

V(G): 30 – 22 + 2 = 10

Caminho 1: 1 - 2 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 15 - 18 - 14 - 19 - 20 - 6 - 21 - 22

Caminho 2: 1 - 2 - 3 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 15 - 18 - 14 - 19 - 20 - 6 - 21 - 22

Caminho 3: 1 - 2 - 4 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 15 - 18 - 14 - 19 - 20 - 6 - 21 - 22

Caminho 4: 1 - 2 - 5 - 6 - 7 - 9 - 20 - 6 - 21 - 22

Caminho 5: 1 - 2 - 5 - 6 - 7 - 10 - 20 - 6 - 21 - 22

Caminho 6: 1 - 2 - 5 - 6 - 7 - 8 - 11 - 12 - 22

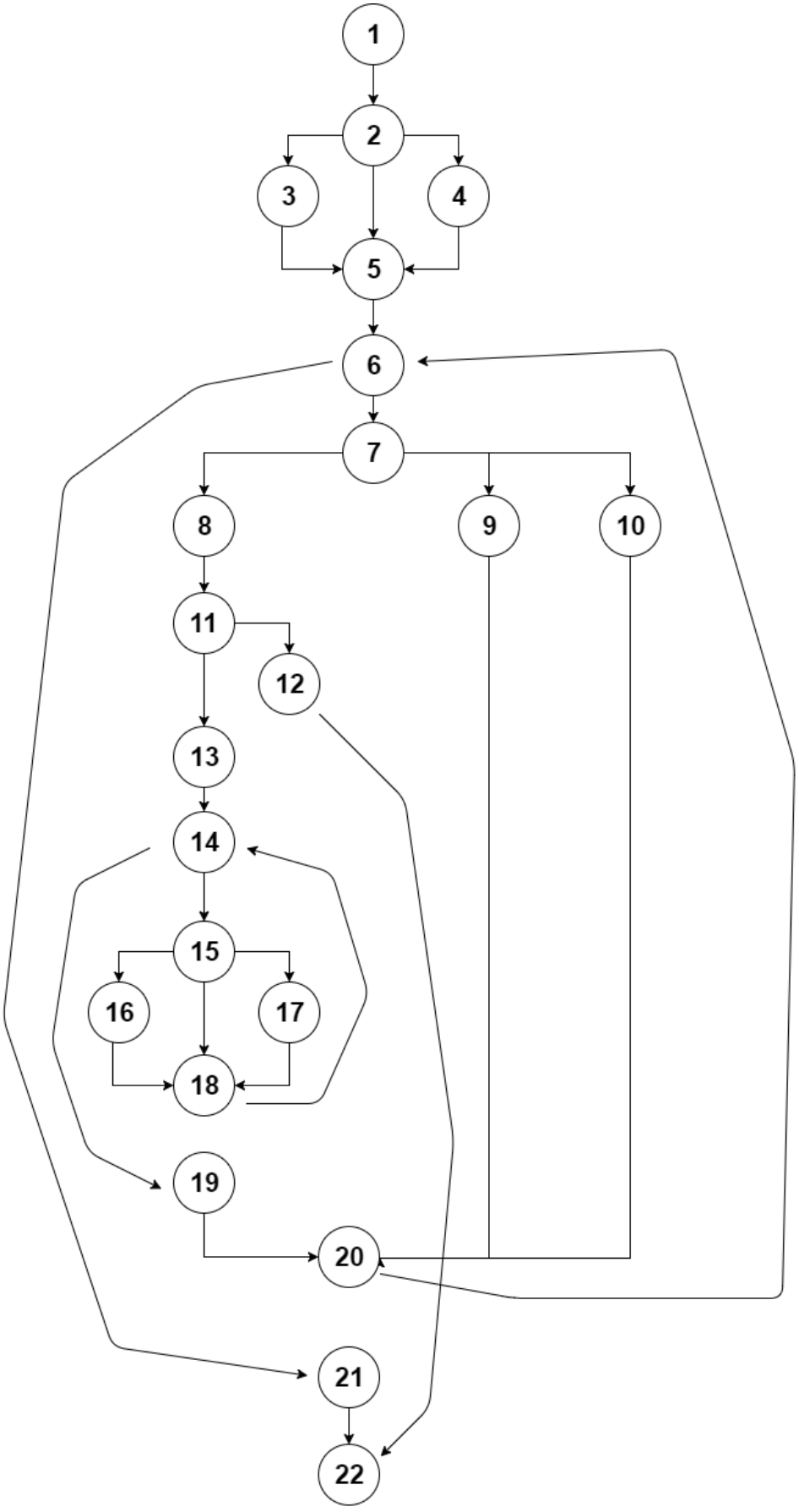
Caminho 7: 1 - 2 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 15 - 16 - 18 - 14 - 19 - 20 - 6 - 21 - 22

Caminho 8: 1 - 2 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 15 - 17 - 18 - 14 - 19 - 20 - 6 - 21 - 22

Caminho 9: 1 - 2 - 5 - 6 - 21 - 22

Caminho 10: 1 - 2 - 5 - 6 - 7 - 8 - 11 - 13 - 14 - 19 - 20 - 9 - 21 - 22

```c
void devolver() {
    Devolucao d;
    Locacao l;
    Locacao locTemp;
    char dataDevolucao[12];
    int difDatas;
    int difExtra;
    int difExtraTemp;
    int opcao = 0;
    int index = 0;
    float valorInicial;
    float valorMulta;
    float valorTotal;
    FILE *outfile;
    FILE *outfile_locacao;

    imprimeCabecalho("DEVOLVER VEÍCULO");

    l = procurarLocacao();
    printf("Veículo %s %s alugado para %s no período de %s a %s encontrado!\n\n", l.vei.marca, l.vei.modelo,

    printf("Data de devolução (dd/mm/yyyy): ");
    scanf("%s", &dataDevolucao);

    difDatas = getDifferenceDays(l.dataLocacao, l.dataDevolucao);
    difExtra = getDifferenceDays(l.dataDevolucao, dataDevolucao);

    valorInicial = difDatas * getValorVeiculo(l.vei);

    if (difExtra > 0) {
        valorMulta = (difExtra * getValorVeiculo(l.vei)) * 2;
        difExtraTemp = difExtra;
    } else {
        valorMulta = 0;
        difExtraTemp = 0;
    }

    valorTotal = valorInicial + valorMulta;

    printf("\nDias Locados: %d\n", difDatas);
    printf("Dias Extras: %d\n", difExtraTemp);

    printf("\nInicial: %7.2f\n", valorInicial);
    printf("Multas:  %7.2f\n", valorMulta);
    printf("TOTAL:   %7.2f\n", valorTotal);

    while (opcao == 0) {
        printf("\nSelecione uma opção:\n1. Confirmar devolução e pagamento\n2. Cancelar\n\nOPÇÃO: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                d.devId = gerarId();
                strcpy(d.dataDevolucao, dataDevolucao);
                d.loc = l;
                d.valorTotal = valorTotal;
                d.valorInicial = valorInicial;
                d.valorMulta = valorMulta;
                d.diasAtraso = difExtraTemp;

                outfile = fopen(FILE_DEVOLUCAO, "a");
                outfile_locacao = fopen(FILE_LOCACAO, "r+");

                if (outfile == NULL || outfile_locacao == NULL) {
                    fprintf(stderr, MSG_ERRO_ARQUIVO);
                    exit(1);
                }

                while (fread(&locTemp, sizeof (Locacao), 1, outfile_locacao)) {
                    if (locTemp.locId == l.locId) {
                        locTemp.devolvido = 1;
                        break;
                    } else {
                        index++;
                    }
                }

                fseek(outfile_locacao, index * sizeof (Locacao), SEEK_SET);
                fwrite(&locTemp, sizeof (Locacao), 1, outfile_locacao);
                fwrite(&d, sizeof (Devolucao), 1, outfile);

                if (fwrite != 0) {
                    printf("\nVeículo devolvido com sucesso! Aguarde...\n");
                } else {
                    printf("\nErro ao devolver o veículo. Reinicie o programa e tente novamente.\n");
                }

                fclose(outfile);
                fclose(outfile_locacao);

                sleep(2);
                inicio();
                break;
            case 2:
                inicio();
                break;
            default:
                opcao = 0;
                printf("Opção inválida!\n\n");
                break;
        }
    }
}
```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

1.19 Função main

V(G): 1 – 2 + 2 = 1

Caminho 1: 1 – 2

```
734
735  int main() {
736      inicio();
737  }
738
739
```