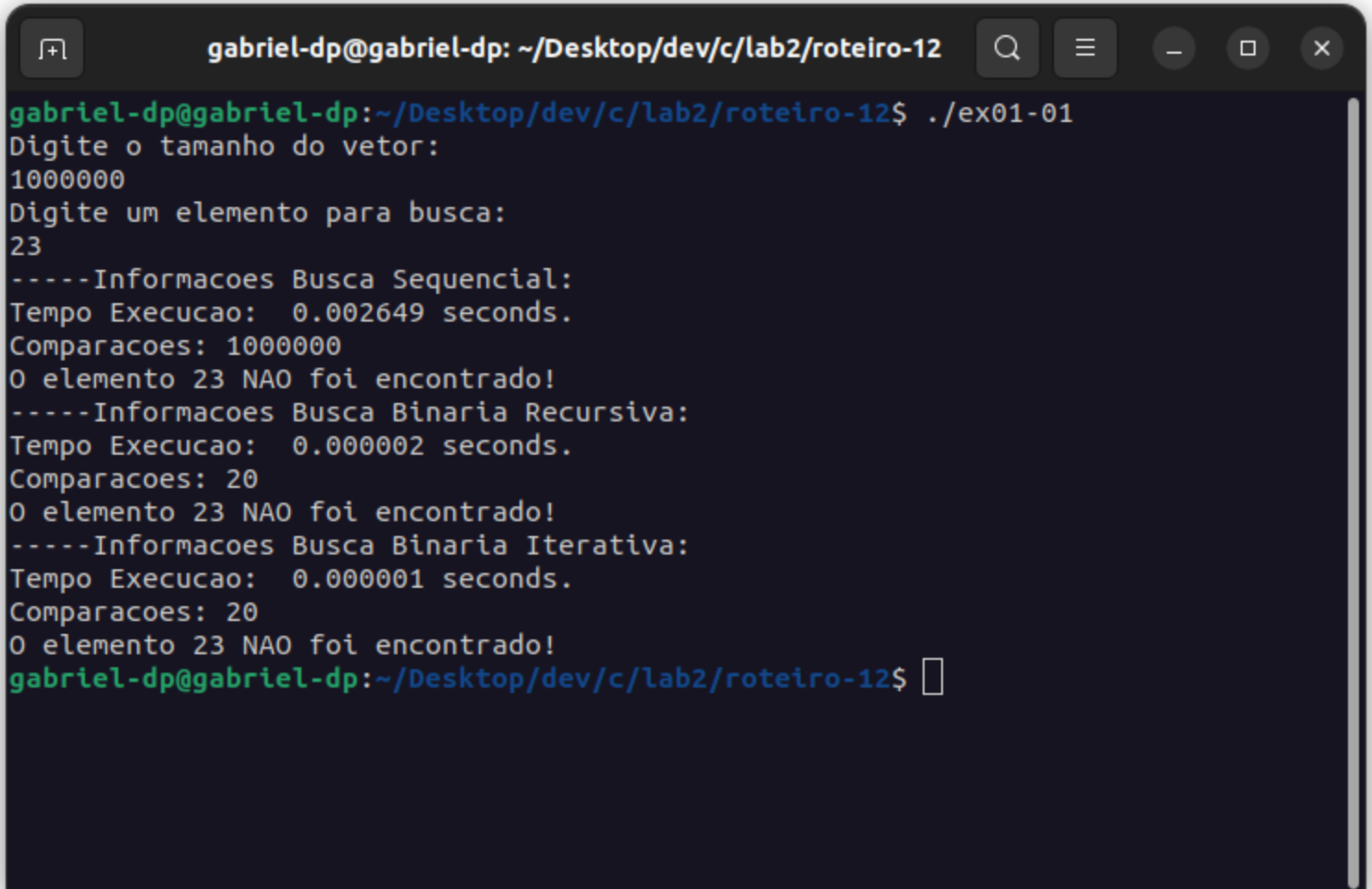
roteiro-12/ex01-01.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int comp;

void preencheAleatorio(int *v, int n, int ini, int fim) {
    int i;
    for (i = 0; i < n; i++)
        v[i] = ini + rand() % (fim - ini + 1);
}

void troca(int *a, int *b) {
    int aux = *a;
    *a = *b;
    *b = aux;
}

int buscaSequencial(int *v, int n, int elem) {
    int i;
    for (i = 0; i < n; i++) {
        comp++;
        if (v[i] == elem)
            return i;  // Elemento encontrado
    }
    return -1;  // Elemento nao encontrado
}

int particao(int *v, int ini, int fim) {
    int i = ini, j = fim;
    int pivo = v[(ini + fim) / 2];
    while (1) {
        while (v[i] < pivo) {
            i++;
        }  // procura algum >= pivo do lado esquerdo
        while (v[j] > pivo) {
            j--;
        }  // procura algum <= pivo do lado direito

        if (i < j) {
            troca(&v[i], &v[j]);  // troca os elementos encontrados
            i++;
            j--;
        } else
            return j;  // retorna o local onde foi feita a particao
    }
}

void QuickSort(int *v, int ini, int fim) {
    if (ini < fim) {
        int q = particao(v, ini, fim);
        QuickSort(v, ini, q);
        QuickSort(v, q + 1, fim);
    }
}

```

```c
int rec_buscaBinaria(int *v, int ini, int fim, int elem) {
    if (ini > fim) return -1;
    int meio = (ini + fim) / 2;
    comp++;
    if (v[meio] == elem)
        return meio;
    else if (elem < v[meio])
        return rec_buscaBinaria(v, ini, meio - 1, elem);
    else
        return rec_buscaBinaria(v, meio + 1, fim, elem);
}

int it_buscaBinaria(int *v, int ini, int fim, int elem) {
    int meio;
    while (ini <= fim) {
        meio = (ini + fim) / 2;
        comp++;
        if (elem == v[meio])
            return meio;
        else if (elem < v[meio])
            fim = meio - 1;
        else
            ini = meio + 1;
    }
    return -1;
}

int main() {
    srand(time(NULL));
    comp = 0;
    clock_t t;

    int *v;
    int n, x;
    printf("Digite o tamanho do vetor:\n");
    scanf("%d", &n);
    v = (int *)malloc(n * sizeof(int));

    preencheAleatorio(v, n, 1, n);
    QuickSort(v, 0, n - 1);

    printf("Digite um elemento para busca:\n");
    scanf("%d", &x);

    int ind;

    t = clock();
    ind = buscaSequencial(v, n, x);
    t = clock() - t;
    printf("-----Informacoes Busca Sequencial:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento %d foi encontrado na pos %d.\n", x, ind);
    else
        printf("O elemento %d NAO foi encontrado!\n", x);

    comp = 0;
```

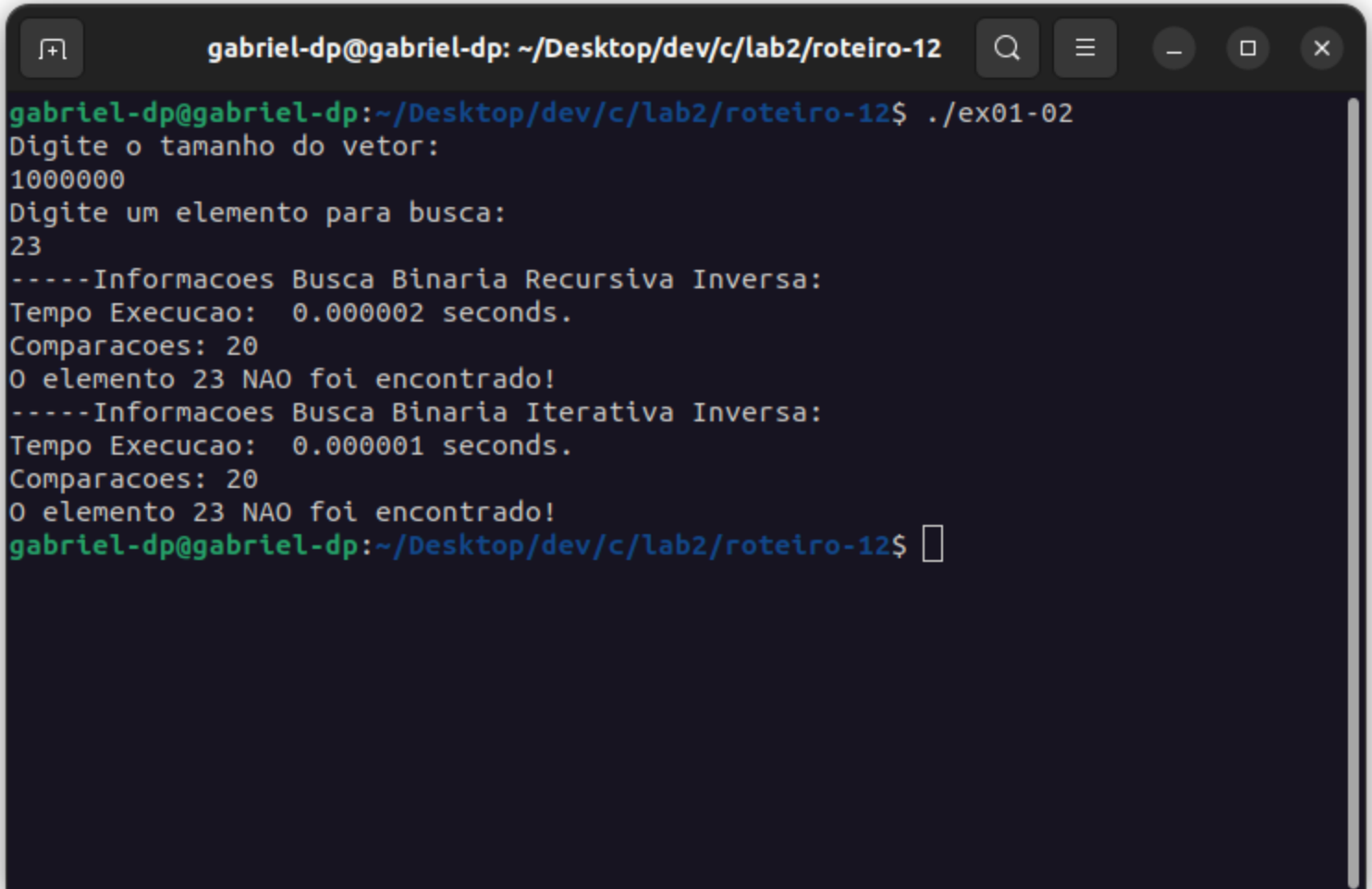
```
    t = clock();
    ind = rec_buscaBinaria(v, 0, n - 1, x);
    t = clock() - t;
    printf("-----Informacoes Busca Binaria Recursiva:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento %d foi encontrado na pos %d.\n", x, ind);
    else
        printf("O elemento %d NAO foi encontrado!\n", x);

    comp = 0;
    t = clock();
    ind = it_buscaBinaria(v, 0, n - 1, x);
    t = clock() - t;
    printf("-----Informacoes Busca Binaria Iterativa:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento %d foi encontrado na pos %d.\n", x, ind);
    else
        printf("O elemento %d NAO foi encontrado!\n", x);

    free(v);
    return 0;
}
```


gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex01-01
Digite o tamanho do vetor:
1000000
Digite um elemento para busca:
23
-----Informacoes Busca Sequencial:
Tempo Execucao: 0.002649 seconds.
Comparacoes: 1000000
O elemento 23 NAO foi encontrado!
-----Informacoes Busca Binaria Recursiva:
Tempo Execucao: 0.000002 seconds.
Comparacoes: 20
O elemento 23 NAO foi encontrado!
-----Informacoes Busca Binaria Iterativa:
Tempo Execucao: 0.000001 seconds.
Comparacoes: 20
O elemento 23 NAO foi encontrado!
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$

roteiro-12/ex01-02.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int comp;

void preencheAleatorio(int *v, int n, int ini, int fim) {
    int i;
    for (i = 0; i < n; i++)
        v[i] = ini + rand() % (fim - ini + 1);
}

void troca(int *a, int *b) {
    int aux = *a;
    *a = *b;
    *b = aux;
}

int particao(int *v, int ini, int fim) {
    int i = ini, j = fim;
    int pivo = v[(ini + fim) / 2];
    while (1) {
        while (v[i] > pivo) {
            i++;
        }
        while (v[j] < pivo) {
            j--;
        }

        if (i < j) {
            troca(&v[i], &v[j]);
            i++;
            j--;
        } else
            return j;
    }
}

void QuickSort(int *v, int ini, int fim) {
    if (ini < fim) {
        int q = particao(v, ini, fim);
        QuickSort(v, ini, q);
        QuickSort(v, q + 1, fim);
    }
}

int rec_buscaBinaria(int *v, int ini, int fim, int elem) {
    if (ini > fim) return -1;
    int meio = (ini + fim) / 2;
    comp++;
    if (v[meio] == elem)
        return meio;
    else if (elem > v[meio])
        return rec_buscaBinaria(v, ini, meio - 1, elem);
    else
        return rec_buscaBinaria(v, meio + 1, fim, elem);
}

int it_buscaBinaria(int *v, int ini, int fim, int elem) {
```

```c
    int meio;
    while (ini <= fim) {
        meio = (ini + fim) / 2;
        comp++;
        if (elem == v[meio])
            return meio;
        else if (elem > v[meio])
            fim = meio - 1;
        else
            ini = meio + 1;
    }
    return -1;
}

int main() {
    srand(time(NULL));
    comp = 0;
    clock_t t;

    int *v;
    int n, x;
    printf("Digite o tamanho do vetor:\n");
    scanf("%d", &n);
    v = (int *)malloc(n * sizeof(int));

    preencheAleatorio(v, n, 1, n);
    QuickSort(v, 0, n - 1);

    printf("Digite um elemento para busca:\n");
    scanf("%d", &x);

    int ind;

    comp = 0;
    t = clock();
    ind = rec_buscaBinaria(v, 0, n - 1, x);
    t = clock() - t;
    printf("-----Informacoes Busca Binaria Recursiva Inversa:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento %d foi encontrado na pos %d.\n", x, ind);
    else
        printf("O elemento %d NAO foi encontrado!\n", x);

    comp = 0;
    t = clock();
    ind = it_buscaBinaria(v, 0, n - 1, x);
    t = clock() - t;
    printf("-----Informacoes Busca Binaria Iterativa Inversa:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento %d foi encontrado na pos %d.\n", x, ind);
    else
        printf("O elemento %d NAO foi encontrado!\n", x);

    free(v);
    return 0;
}
```

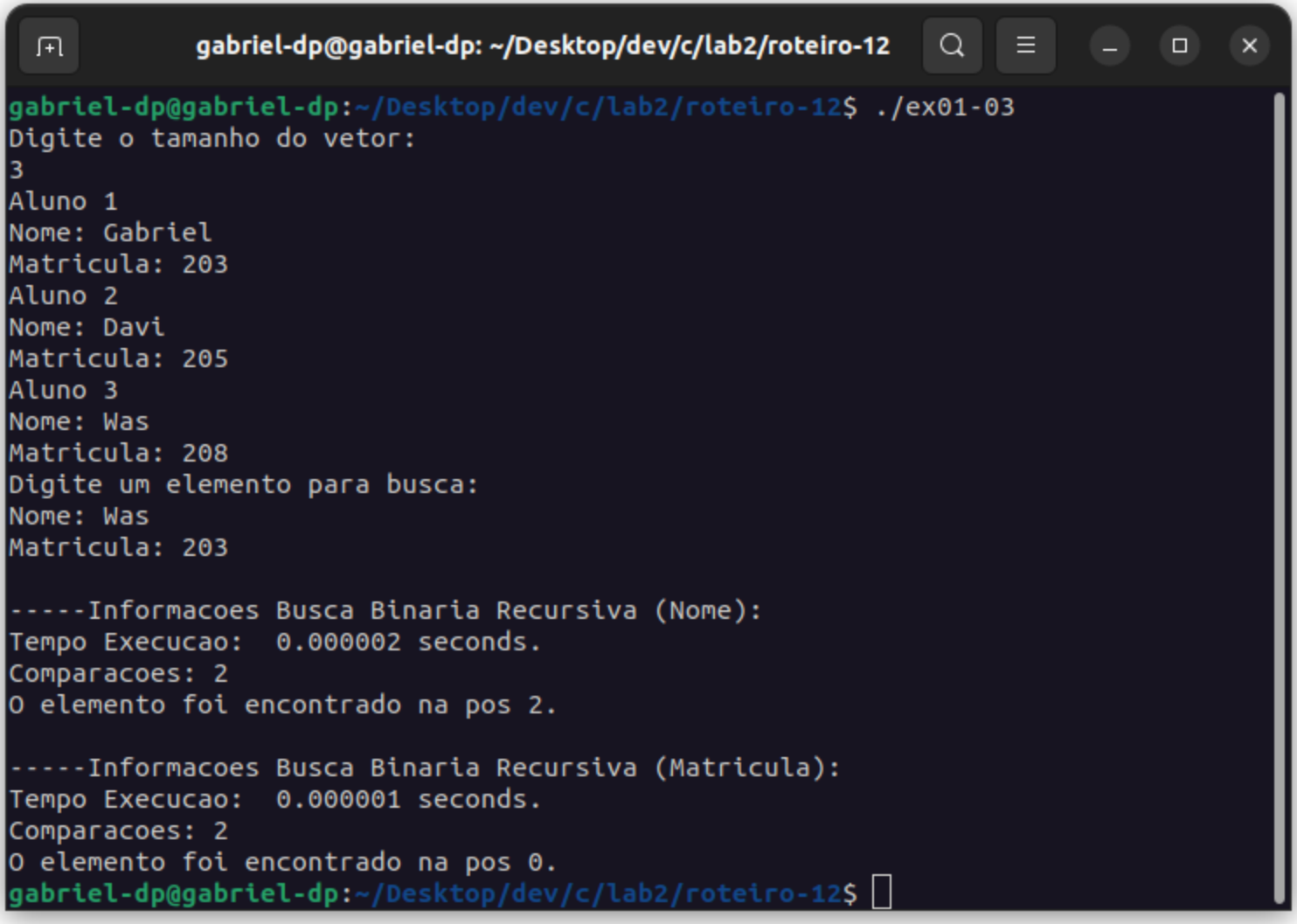

gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex01-02
Digite o tamanho do vetor:
1000000
Digite um elemento para busca:
23
-----Informacoes Busca Binaria Recursiva Inversa:
Tempo Execucao: 0.000002 seconds.
Comparacoes: 20
O elemento 23 NAO foi encontrado!
-----Informacoes Busca Binaria Iterativa Inversa:
Tempo Execucao: 0.000001 seconds.
Comparacoes: 20
O elemento 23 NAO foi encontrado!
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$

roteiro-12/ex01-03.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define NOME_TAMANHO 50

typedef struct {
    char nome[50 + 1];
    int matricula;
} Aluno;

int comp;

void troca(Aluno *a, Aluno *b) {
    Aluno aux = *a;
    *a = *b;
    *b = aux;
}

int particao(Aluno *v, int ini, int fim, int (*compara)(Aluno *, Aluno *)) {
    int i = ini, j = fim;
    Aluno pivo = v[(ini + fim) / 2];
    while (1) {
        while (compara(&(v[i]), &pivo) < 0) {
            i++;
        }
        while (compara(&(v[j]), &pivo) > 0) {
            j--;
        }

        if (i < j) {
            troca(&v[i], &v[j]);
            i++;
            j--;
        } else
            return j;
    }
}

void QuickSort(Aluno *v, int ini, int fim, int (*compara)(Aluno *, Aluno *)) {
    if (ini < fim) {
        int q = particao(v, ini, fim, compara);
        QuickSort(v, ini, q, compara);
        QuickSort(v, q + 1, fim, compara);
    }
}

int rec_buscaBinaria(Aluno *v, int ini, int fim, Aluno elem, int (*compara)
(Aluno *, Aluno *)) {
    if (ini > fim) return -1;
    int meio = (ini + fim) / 2;
    comp++;
    if (compara(&(v[meio]), &elem) == 0)
        return meio;
    else if (compara(&(v[meio]), &elem) > 0)
```

```
56          return rec_buscaBinaria(v, ini, meio - 1, elem, compara);
57      else
58          return rec_buscaBinaria(v, meio + 1, fim, elem, compara);
59  }
60
61  int comparaNome(Aluno *a, Aluno *b) {
62      return strcmp(a->nome, b->nome);
63  }
64
65  int comparaMatricula(Aluno *a, Aluno *b) {
66      return a->matricula - b->matricula;
67  }
68
69  int main() {
70      // Atribuicoes iniciais
71      srand(time(NULL));
72      comp = 0;
73      clock_t t;
74
75      Aluno *v, x;
76      int n;
77      printf("Digite o tamanho do vetor:\n");
78      scanf("%d", &n);
79      v = (Aluno *)malloc(n * sizeof(Aluno));
80
81      for (int i = 0; i < n; i++) {
82          printf("Aluno %d\n", i + 1);
83          printf("Nome: ");
84          fgets(v[i].nome, NOME_TAMANHO, stdin);
85          fgets(v[i].nome, NOME_TAMANHO, stdin);
86          printf("Matricula: ");
87          scanf("%d", &(v[i].matricula));
88      }
89
90      printf("Digite um elemento para busca:\n");
91      printf("Nome: ");
92      fgets(x.nome, NOME_TAMANHO, stdin);
93      fgets(x.nome, NOME_TAMANHO, stdin);
94      printf("Matricula: ");
95      scanf("%d", &(x.matricula));
96
97      int ind;
98
99      QuickSort(v, 0, n - 1, &comparaNome);
100     comp = 0;
101     t = clock();
102     ind = rec_buscaBinaria(v, 0, n - 1, x, &comparaNome);
103     t = clock() - t;
104     printf("\n-----Informacoes Busca Binaria Recursiva (Nome):\n");
105     printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
106     printf("Comparacoes: %d\n", comp);
107
108     if (ind != -1)
109         printf("O elemento foi encontrado na pos %d.\n", ind);
110     else
111         printf("O elemento NAO foi encontrado!\n");
112
113     QuickSort(v, 0, n - 1, comparaMatricula);
114     comp = 0;
```

```c
    t = clock();
    ind = rec_buscaBinaria(v, 0, n - 1, x, comparaMatricula);
    t = clock() - t;
    printf("\n-----Informacoes Busca Binaria Recursiva (Matricula):\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t) / CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", comp);

    if (ind != -1)
        printf("O elemento foi encontrado na pos %d.\n", ind);
    else
        printf("O elemento NAO foi encontrado!\n");

    free(v);
    return 0;
}
```


gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex01-03
Digite o tamanho do vetor:
3
Aluno 1
Nome: Gabriel
Matricula: 203
Aluno 2
Nome: Davi
Matricula: 205
Aluno 3
Nome: Was
Matricula: 208
Digite um elemento para busca:
Nome: Was
Matricula: 203
-----Informacoes Busca Binaria Recursiva (Nome):
Tempo Execucao: 0.000002 seconds.
Comparacoes: 2
O elemento foi encontrado na pos 2.
-----Informacoes Busca Binaria Recursiva (Matricula):
Tempo Execucao: 0.000001 seconds.
Comparacoes: 2
O elemento foi encontrado na pos 0.
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$

roteiro-12/hash.h

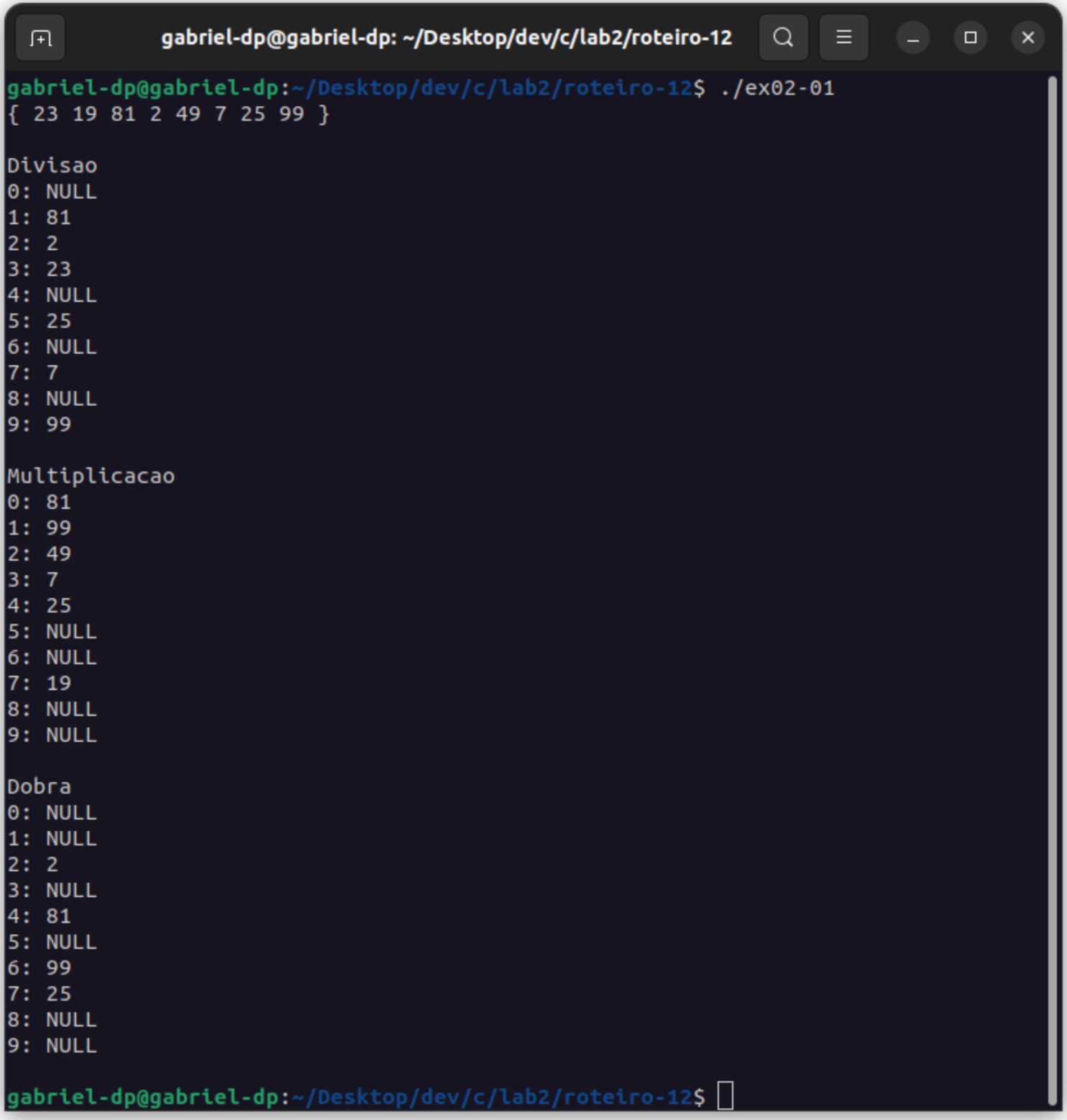
```c
#ifndef HASH_H
#define HASH_H

#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int** tabela;
    int tam, qtd;
} Hash;

Hash* criaHash(int t) {
    Hash* h;
    h = (Hash*)malloc(sizeof(Hash));
    if (h != NULL) {
        h->tam = t;
        h->qtd = 0;
        h->tabela = (int**)malloc(t * sizeof(int*));
        if (h->tabela == NULL) return NULL;
        int i;
        for (i = 0; i < t; i++)
            h->tabela[i] = NULL;
    }
    return h;
}

void destroiHash(Hash* h) {
    if (h != NULL) {
        int i;
        for (i = 0; i < h->tam; i++)
            if (h->tabela[i] != NULL)
                free(h->tabela[i]);
        free(h->tabela);
        free(h);
    }
}

int chaveDivisao(int chave, int tam) {
    return (chave & 0x7FFFFFFF) % tam;
}

int chaveMultiplicacao(int chave, int tam) {
    float A = 0.6180339887;  // constante: 0 < A < 1
    float val = chave * A;
    val = val - (int)val;
    return (int)(tam * val);
}

int chaveDobra(int chave, int tam) {
    int pos, n_bits = 30;

    int p = 1;
    int r = p << n_bits;
    while ((chave & r) != r) {
        n_bits--;
        r = p << n_bits;
```

```
57      }
58
59      n_bits++;
60      pos = chave;
61      while (pos > tam) {
62          int metade_bits = n_bits / 2;
63          int parte1 = pos >> metade_bits;
64          parte1 = parte1 << metade_bits;
65          int parte2 = pos ^ parte1;
66          parte1 = pos >> metade_bits;
67          pos = parte1 ^ parte2;
68          n_bits = n_bits / 2;
69      }
70      return pos;
71  }
72
73  int insereHash_semTratar(Hash* h, int elem, int (*funChave)(int, int)) {
74      if (h == NULL) return 0;
75      int pos = funChave(elem, h->tam);
76
77      if (h->tabela[pos] == NULL) {
78          int* novo = (int*)malloc(sizeof(int));
79          if (novo == NULL) return 0;
80          *novo = elem;
81          h->tabela[pos] = novo;
82          h->qtd++;
83      } else
84          *(h->tabela[pos]) = elem;
85      return 1;
86  }
87
88  int buscaHash_semTratar(Hash* h, int elem, int (*funChave)(int, int), int* p) {
89      if (h == NULL) return 0;
90      int pos = funChave(elem, h->tam);
91      if (h->tabela[pos] == NULL) return 0;
92      if (*(h->tabela[pos]) == elem) {
93          *p = *(h->tabela[pos]);
94          return 1;
95      }
96      return 0;
97  }
98
99  int sondagemLinear(int pos, int chave, int i, int tam, int (*funChave)(int,
    int)) {
100     return ((pos + i) & 0x7FFFFFFF) % tam;
101 }
102
103 int sondagemQuadratica(int pos, int chave, int i, int tam, int (*funChave)(int,
    int)) {
104     pos = pos + 2 * i + 5 * i * i;
105     return (pos & 0x7FFFFFFF) % tam;
106 }
107
108 int sondagemDuploHash(int H1, int chave, int i, int tam, int (*funChave)(int,
    int)) {
109     int H2 = funChave(chave, tam - 1) + 1;
110     return ((H1 + i * H2) & 0x7FFFFFFF) % tam;
111 }
112
```

```c
int insereHash_EnderAberto(Hash* h, int elem, int (*funChave)(int, int), int (*funSondagem)(int, int, int, int, int (*)(int, int))) {
    if (h == NULL) return 0;
    int i, pos, newPos;
    pos = funChave(elem, h->tam);
    for (i = 0; i < h->tam; i++) {
        newPos = funSondagem(pos, elem, i, h->tam, funChave);
        if (h->tabela[newPos] == NULL) {
            int* novo = (int*)malloc(sizeof(int));
            if (novo == NULL) return 0;
            *novo = elem;
            h->tabela[newPos] = novo;
            h->qtd++;
            return 1;
        }
    }
    return 0;
}

int buscaHash_EnderAberto(Hash* h, int elem, int* p, int (*funChave)(int, int), int (*funSondagem)(int, int, int, int, int (*)(int, int))) {
    if (h == NULL) return 0;
    int i, pos, newPos;
    pos = funChave(elem, h->tam);
    for (i = 0; i < h->tam; i++) {
        newPos = funSondagem(pos, elem, i, h->tam, funChave);
        if (h->tabela[newPos] == NULL) return 0;
        if (*(h->tabela[newPos]) == elem) {
            *p = *(h->tabela[newPos]);
            return 1;
        }
    }
    return 0;
}

void imprimeHash(Hash* h) {
    if (h == NULL) return;
    int i;
    for (i = 0; i < h->tam; i++) {
        printf("%d: ", i);
        if (h->tabela[i] == NULL)
            printf("NULL\n");
        else
            printf("%d\n", *(h->tabela[i]));
    }
}

#endif
```

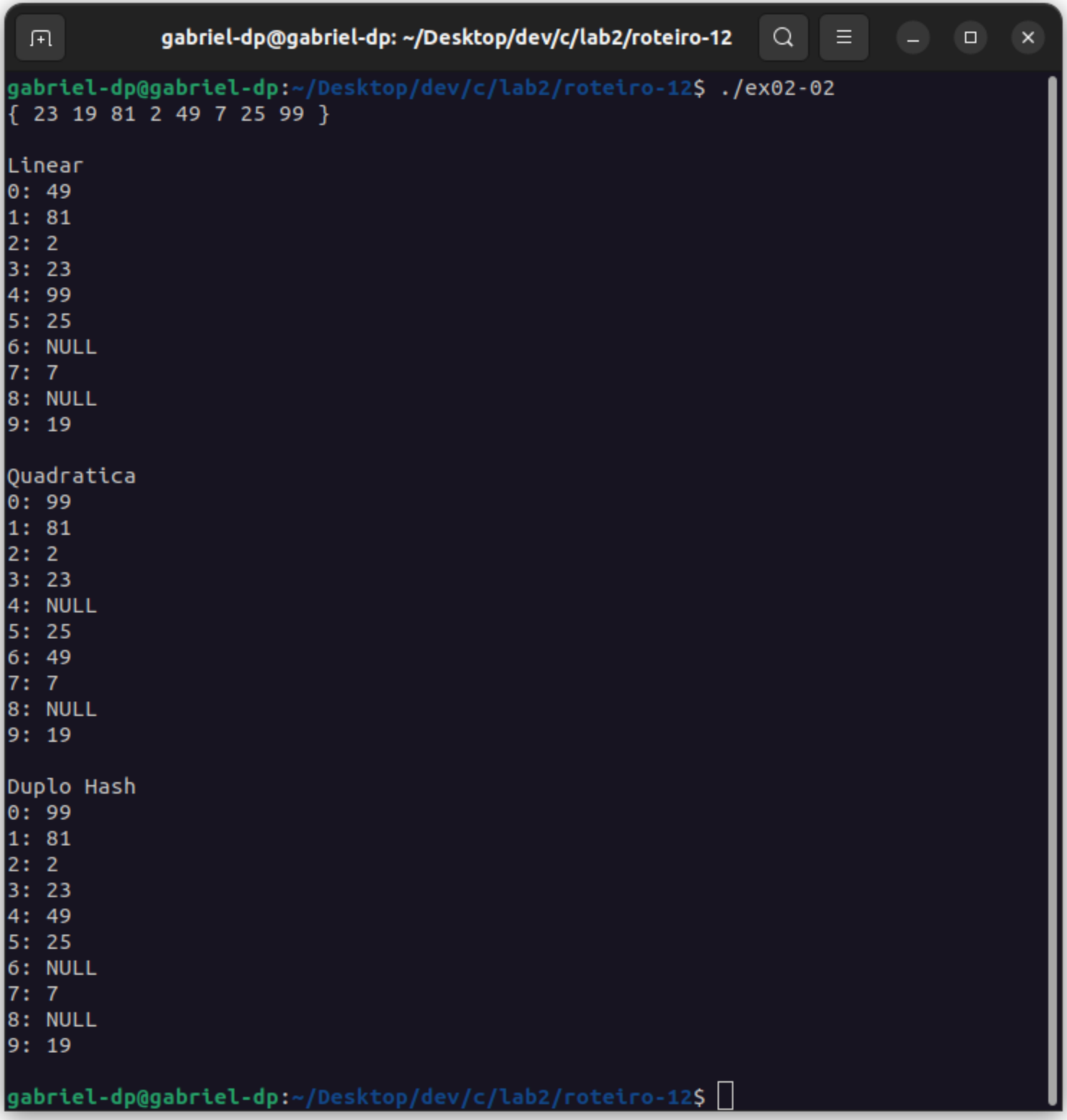
roteiro-12/ex02-01.c

```c
#include <stdio.h>

#include "hash.h"

int main() {
    int elementos[] = {23, 19, 81, 2, 49, 7, 25, 99};
    char* funsNome[] = {"Divisao", "Multiplicacao", "Dobra"};
    int (*funsChave[])(int, int) = {&chaveDivisao, &chaveMultiplicacao, &chaveDobra};

    printf("{ ");
    for (int j = 0; j < (sizeof(elementos) / sizeof(int)); j++) {
        printf("%d ", elementos[j]);
    }
    printf("}\n\n");

    for (int i = 0; i < (sizeof(funsChave) / sizeof(funsChave[0])); i++) {
        Hash* tabela = criaHash(10);

        for (int j = 0; j < (sizeof(elementos) / sizeof(int)); j++)
            insereHash_semTratar(tabela, elementos[j], funsChave[i]);

        printf("%s\n", funsNome[i]);
        imprimeHash(tabela);
        printf("\n");

        destroiHash(tabela);
    }

    return 0;
}
```


gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex02-01
{ 23 19 81 2 49 7 25 99 }
Divisao
0: NULL
1: 81
2: 2
3: 23
4: NULL
5: 25
6: NULL
7: 7
8: NULL
9: 99
Multiplicacao
0: 81
1: 99
2: 49
3: 7
4: 25
5: NULL
6: NULL
7: 19
8: NULL
9: NULL
Dobra
0: NULL
1: NULL
2: 2
3: NULL
4: 81
5: NULL
6: 99
7: 25
8: NULL
9: NULL
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$

roteiro-12/ex02-02.c

```c
#include <stdio.h>

#include "hash.h"

int main() {
    int elementos[] = {23, 19, 81, 2, 49, 7, 25, 99};
    char* funsNome[] = {"Linear", "Quadratica", "Duplo Hash"};
    int (*funsSondagem[])(int, int, int, int, int (*)(int, int)) = {&sondagemLinear, &sondagemQuadratica, &sondagemDuploHash};

    printf("{ ");
    for (int j = 0; j < (sizeof(elementos) / sizeof(int)); j++)
        printf("%d ", elementos[j]);
    printf("}\n\n");

    for (int i = 0; i < (sizeof(funsSondagem) / sizeof(funsSondagem[0])); i++) {
        Hash* tabela = criaHash(10);

        for (int j = 0; j < (sizeof(elementos) / sizeof(int)); j++)
            insereHash_EnderAberto(tabela, elementos[j], &chaveDivisao, funsSondagem[i]);

        printf("%s\n", funsNome[i]);
        imprimeHash(tabela);
        printf("\n");

        destroiHash(tabela);
    }

    return 0;
}
```


gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex02-02
{ 23 19 81 2 49 7 25 99 }
Linear
0: 49
1: 81
2: 2
3: 23
4: 99
5: 25
6: NULL
7: 7
8: NULL
9: 19
Quadratica
0: 99
1: 81
2: 2
3: 23
4: NULL
5: 25
6: 49
7: 7
8: NULL
9: 19
Duplo Hash
0: 99
1: 81
2: 2
3: 23
4: 49
5: 25
6: NULL
7: 7
8: NULL
9: 19
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$

roteiro-12/hash-lse.h

```c
#ifndef HASH_LSE_H
#define HASH_LSE_H

#include <stdio.h>
#include <stdlib.h>

#include "lse.h"

typedef struct {
    Lista **tabela;
    int tam, qtd;
} Hash;

Hash *criaHash(int t) {
    Hash *h;
    h = (Hash *)malloc(sizeof(Hash));
    if (h != NULL) {
        h->tam = t;
        h->qtd = 0;
        h->tabela = (Lista **)malloc(t * sizeof(Lista *));
        if (h->tabela == NULL) return NULL;
        int i;
        for (i = 0; i < t; i++)
            h->tabela[i] = NULL;
    }
    return h;
}

void destroiHash(Hash *h) {
    if (h != NULL) {
        int i;
        for (i = 0; i < h->tam; i++)
            if (h->tabela[i] != NULL)
                destroiLista(h->tabela[i]);
        free(h->tabela);
        free(h);
    }
}

int chaveDivisao(int chave, int tam) {
    return (chave & 0x7FFFFFFF) % tam;
}

int insereHashLSE(Hash *h, int elem) {
    if (h == NULL) return 0;
    int pos = chaveDivisao(elem, h->tam);
    if (h->tabela[pos] == NULL)
        h->tabela[pos] = criaLista();
    insereIni(h->tabela[pos], elem);
    h->qtd++;
    return 1;
}

int buscaHashLSE(Hash *h, int elem, int *p) {
    if (h == NULL) return 0;
    int pos = chaveDivisao(elem, h->tam);
```

```c
    if (h->tabela[pos] == NULL) return 0;
    return listaBuscaElem(h->tabela[pos], elem, p);
}

void imprimeHash(Hash *h) {
    if (h == NULL) return;
    int i;
    for (i = 0; i < h->tam; i++) {
        printf("%d: ", i);
        if (h->tabela[i] == NULL)
            printf("NULL\n");
        else
            imprimeLista(h->tabela[i]);
    }
}

#endif
```

roteiro-12/lse.h

```c
#ifndef LISTASE_H
#define LISTASE_H

#include <stdio.h>
#include <stdlib.h>

typedef struct NO {
    int info;
    struct NO* prox;
} NO;

typedef struct NO* Lista;

Lista* criaLista() {
    Lista* li;
    li = (Lista*)malloc(sizeof(Lista));
    if (li != NULL) {
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista* li) {
    if (li == NULL) return 1;
    if (*li == NULL) return 1;  // True - Vazia!
    return 0;                   // False - tem elemento!
}

NO* alocarNO() {
    return (NO*)malloc(sizeof(NO));
}

void liberarNO(NO* q) {
    free(q);
}

int listaBuscaElem(Lista* li, int elem, int* p) {
    if (li == NULL) return 0;
    NO* aux = *li;
    while (aux != NULL) {
        if (aux->info == elem) {
            *p = aux->info;
            return 1;
        }
        aux = aux->prox;
    }
    return 0;
}

int insereIni(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    *li = novo;
```

```c
    return 1;
}

int insereFim(Lista* li, int elem) {
    if (li == NULL) return 0;
    NO* novo = alocarNO();
    if (novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if (listaVazia(li)) {
        *li = novo;
    } else {
        NO* aux = *li;
        while (aux->prox != NULL)
            aux = aux->prox;
        aux->prox = novo;
    }
    return 1;
}

int removeIni(Lista* li) {
    if (li == NULL) return 0;
    if (listaVazia(li)) return 0;
    NO* aux = *li;
    *li = aux->prox;
    liberarNO(aux);
    return 1;
}

int removeFim(Lista* li) {
    if (li == NULL) return 0;
    if (listaVazia(li)) return 0;
    NO *ant, *aux = *li;
    while (aux->prox != NULL) {
        ant = aux;
        aux = aux->prox;
    }
    if (aux == *li)
        *li = aux->prox;
    else
        ant->prox = aux->prox;
    liberarNO(aux);
    return 1;
}

void imprimeLista(Lista* li) {
    if (li == NULL) return;
    if (listaVazia(li)) {
        printf("Lista Vazia!\n");
        return;
    }
    // printf("Elementos:\n");
    NO* aux = *li;
    while (aux != NULL) {
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("\n");
}
```

```
116
117 void destroiLista(Lista* li) {
118     if (li != NULL) {
119         NO* aux;
120         while ((*li) != NULL) {
121             aux = *li;
122             *li = (*li)->prox;
123             liberarNO(aux);
124         }
125         free(li);
126     }
127 }
128
129 #endif
130
```

roteiro-12/ex02-03.c

```c
#include <stdio.h>

#include "hash-lse.h"

int main() {
    int elementos[] = {23, 19, 81, 2, 49, 7, 25, 99};

    Hash* tabela = criaHash(10);

    printf("{ ");
    for (int j = 0; j < (sizeof(elementos) / sizeof(int)); j++) {
        printf("%d ", elementos[j]);
        insereHashLSE(tabela, elementos[j]);
    }
    printf("}\n\n");

    imprimeHash(tabela);

    destroiHash(tabela);

    return 0;
}
```


gabriel-dp@gabriel-dp: ~/Desktop/dev/c/lab2/roteiro-12
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$ ./ex02-03
{ 23 19 81 2 49 7 25 99 }
0: NULL
1: 81
2: 2
3: 23
4: NULL
5: 25
6: NULL
7: 7
8: NULL
9: 99 49 19
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-12$