LISTA 2 – LUCAS EDUARDO LEITE COSTA

1.1

exerc1.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "conta.c"

int main(){
    ContaBancaria *c = (ContaBancaria*)malloc(sizeof(ContaBancaria));
    int numero;
    double saque, deposito;
    char titular[100];

    scanf("%s", &titular);
    scanf("%d", &numero);
    criarConta(c, numero, titular);
    scanf("%lf", &deposito);
    depositar(c, deposito);
    scanf("%lf", &saque);
    sacar(c, saque);
    printf("Seu saldo é: R$%.2lf\n", consultarSaldo(c));
    imprimirInfo(c);
    free(c);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[lucascosta@fedora roteiro_2]$ ./teste
Lucas
10
758.60
100.10
Seu saldo é: R$658.50
Titular = Lucas - Número da Conta = 10 - Saldo = R$658.50
[lucascosta@fedora roteiro_2]$
```

conta.h > imprimirInfo(ContaBancaria *)

```c
#ifndef conta_h
#define conta_h

typedef struct ContaBancaria
{
    int num_conta;
    double saldo;
    char titular[100];
}ContaBancaria;

void criarConta(ContaBancaria* c, int numero, char *titular);
void depositar(ContaBancaria* c, double valor);
void sacar(ContaBancaria* c, double valor);
double consultarSaldo(ContaBancaria* c);
void imprimirInfo(ContaBancaria* c);


#endif
```

conta.c > ...

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

typedef struct ContaBancaria
{
   int num_conta;
   double saldo;
   char titular[100];
}ContaBancaria;

void criarConta(ContaBancaria* c, int numero, char *titular){
    strcpy(c->titular, titular);
    c->saldo = 0;
    c->num_conta = numero;
}

void depositar(ContaBancaria* c, double valor){
    c->saldo += valor;
}

void sacar(ContaBancaria* c, double valor){
    if(c->saldo >= valor){
        c->saldo -= valor;
    }else{
        printf("Saldo insuficiente!\n");
    }
}

double consultarSaldo(ContaBancaria* c){
    return c->saldo;
}

void imprimirInfo(ContaBancaria* c){
    printf("Titular = %s - Número da Conta = %d - Saldo = R$%.2lf\n", c->titular , c->num_conta, c->saldo)
}
```

1.2

main.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "catalogo.h"

int main(){
    char nome[100], produto[100];
    double preco;
    int quantidade, total;
    CatalogoProdutos *c = (CatalogoProdutos*)malloc(sizeof(CatalogoProduto
    criarCatalogo(c);
    scanf("%s %lf %d", nome, &preco, &quantidade);
    adicionarProduto(c, nome, preco, quantidade);
    scanf("%s %lf %d", nome, &preco, &quantidade);
    adicionarProduto(c, nome, preco, quantidade);
    scanf("%s", produto);
    printf("Quantidade de %s no estoque: %d\n", produto ,verificarEstoque(
    imprimirCatalogo(c);
    free(c);
    return 0;
}
```

makefile

```
all: catalogo.o main.o
    gcc catalogo.o main.o -o main
main.o: main.c
    gcc main.c -c
catalogo.o: catalogo.h
    gcc -c catalogo.c
clean: rm -rf *.o
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[lucascosta@fedora roteiro_2]$ ./main
caneta 1.50 6
lapis 1.10 7
lapis
Quantidade de lapis no estoque: 7

 - CATÁLOGO -
Nome: caneta - Preço: R$1.50 - Quantidade:6
Nome: lapis - Preço: R$1.10 - Quantidade:7
[lucascosta@fedora roteiro_2]$
```

catalogo.c > imprimirCatalogo(CatalogoProdutos *)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "catalogo.h"

void criarCatalogo(CatalogoProdutos *c){
    c->total = 0;
}

void adicionarProduto(CatalogoProdutos *c, char *nome, double preco, int quantidade){
    strcpy(c->produtos[c->total].nome, nome);
    c->produtos[c->total].preco = preco;
    c->produtos[c->total].quantidade = quantidade;
    c->total += 1;
}

int verificarEstoque(CatalogoProdutos *c, char *nome){
    for(int i = 0; i < 100; i++){
        if(strcmp(c->produtos[i].nome, nome) == 0){
            return c->produtos[i].quantidade;
        }
    }
    return 0;
}

void imprimirCatalogo(CatalogoProdutos *c){
    printf("\n - CATÁLOGO - \n");
    for(int i = 0; i < c->total; i++){
        printf("Nome: %s - Preço: R$%.2lf - Quantidade:%d\n", c->produtos[i].nome, c->produtos[i].preco, c->produtos[i].quantidade);
    }
}
```

catalogo.h > CatalogoProdutos > total

```c
#ifndef catalogo_h
#define catalogo_h

typedef struct Produto{
    char nome[100];
    double preco;
    int quantidade;
}Produto;

typedef struct CatalogoProdutos{
    Produto produtos[100];
    int total;
}CatalogoProdutos;

void criarCatalogo(CatalogoProdutos *c);
void adicionarProduto(CatalogoProdutos *c, char *nome, double preco, int quantidade);
int verificarEstoque(CatalogoProdutos *c, char *nome);
void imprimirCatalogo(CatalogoProdutos *c);

#endif
```

2.1 A partir do n >= 64, a ordenação por inserção é pior do que a por intercalação.

2.2 Pelo o resultado da desigualdade, temos que para o algoritmo com tempos de execução de $100n^2$ funcione mais rápido o n deverá ser menor que 1/50.

2.3 Dizemos que g(n) é O(f(n)) quando g(n) é dominada assintoticamente por f(n), com n tendendo ao infinito.

2.4 Quando g(n) é dominada inferiormente por f(n), com n tendendo ao infinito.

2.5 A expressão se contradiz, pois “O” é usado para denotar o limite superior em questão de análise de complexidade, sendo assim dizer “é no mínimo $O(n^2)$” não faz sentido.

2.6 A partir da resolução da igualdade das expressões, temos as raízes 12 e 35, interpretamos como um intervalo que dentro dele o algoritmo A é mais rápido, mas fora desse intervalo o B é mais rápido.

2.7 O pior caso é $O(n^3)$, no momento em que os três laços são percorridos.

2.8 Temos que a complexidade é constante pelo fato que em todos os casos o algoritmo faz n-1 comparações, mesmo o maior elemento estando nas primeiras posições ou em qualquer outra, em todos os casos ele vai percorrer o vetor inteiro, fazendo n-1 comparações, para confirmar o maior elemento, portanto a afirmação é válida.