Universidade Federal de São João del Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Roteiro 2

Rodrigo José Zonzin Esteves
212050002

1 Resolução

1.1 Conta Bancária

```h
#ifndef _CONTABANCARIA_H
#define _CONTABANCARIA_H

struct contaBancaria_t{
    int num_conta;
    double saldo;
    char nome_titular[20];
};
typedef struct contaBancaria_t ContaBancaria;


ContaBancaria* criarConta(int, char*);
void depositar(ContaBancaria*, double);
void sacar(ContaBancaria*, double);
double consultarSaldo(ContaBancaria*);
void imprimirInfo(ContaBancaria*);

#endif
```

../ex11/contaBancaria.h

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

#include "contaBancaria.h"


ContaBancaria* criarConta(int num, char *titular){
    ContaBancaria *c = (ContaBancaria*)malloc(sizeof(ContaBancaria));

    strcpy(c->nome_titular, titular);
    c->num_conta = num;
    c->saldo = 0.00;

    return c;
}

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

void sacar(ContaBancaria *c, double valor){
    if(c->saldo < valor) printf("VALOR INSUFICIENTE PARA SALDO!\n");
    else{
        c->saldo = c->saldo - valor; ;
    }

}

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

void imprimirInfo(ContaBancaria *c){
    printf("TITULAR: %s\n", c->nome_titular);
    printf("NUMERO DA CONTA: %d\n", c->num_conta);
    printf("SALDO: %.2lf\n", c->saldo);
    printf("\n");
}
```

../ex11/contaBancaria.c

```c
#include "contaBancaria.h"

int main(){
    ContaBancaria *c = criarConta(1,
        "Rodrigo Zonzin");
    imprimirInfo(c);

    depositar(c, 1000.05);
    imprimirInfo(c);

    sacar(c, 2.0333);
    imprimirInfo(c);
}
```

../ex11/main.c

```
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/pr2/ex11$ ./main
TITULAR: Rodrigo Zonzin
NUMERO DA CONTA: 1
SALDO: 0.00

TITULAR: Rodrigo Zonzin
NUMERO DA CONTA: 1
SALDO: 1000.05

TITULAR: Rodrigo Zonzin
NUMERO DA CONTA: 1
SALDO: 998.02

zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/pr2/ex11$
```

Figura 1: Saída

1.2 CatalogoProduto

```c
#define _PRODUTO_H


struct produto_t{
    char nome[20];
    double preco;
    int quantidade;
};
typedef struct produto_t Produto;
```

../ex12/Produto.h

```c
#define CATALOGOPRODUTO_H

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

#include "Produto.h"

struct catalogoProdutos_t{
    int n;
    Produto itens[100];
};

typedef struct catalogoProdutos_t
    CatalogoProdutos;


CatalogoProdutos* criarCatalogo();
void
    adicionarProduto(CatalogoProdutos*,
    char*, double, int);
int verificarEstoque(CatalogoProdutos*,
    char*);
void imprimirCatalogo(CatalogoProdutos
    *c);
```

../ex12/CatalogoProduto.h

```c
#include "CatalogoProduto.h"
#include <string.h>

CatalogoProdutos* criarCatalogo(){
    CatalogoProdutos *cat =
        (CatalogoProdutos*)malloc(sizeof(Ca

    cat->n = 0;
    return cat;
}

void adicionarProduto(CatalogoProdutos
    *c, char *nome, double preco, int
    quantidade){
    strcpy(c->itens[c->n].nome, nome);
    c->itens[c->n].preco  = preco;
```

```
14      c->itens[c->n].quantidade =
            quantidade;
15
16      (c->n)++;
17  }
18  int verificarEstoque(CatalogoProdutos
        *c, char *nome){
19      for(int i = 0; i< c->n; i++){
20          if(!strcmp(c->itens[i].nome,
                nome)){
21              return
                    c->itens[i].quantidade;
22          }
23      }
24
25      return -1;
26  }
27  void imprimirCatalogo(CatalogoProdutos
        *c){
28      for(int  i =0; i<c->n; i++){
29          printf("%s\n",
                c->itens[i].nome);
30          printf("%.2lf\n",
                c->itens[i].preco);
31          printf("%d\n",
                c->itens[i].quantidade);
32
33          printf("\n");
34      }
35  }
36
37
38  int main(){
39      CatalogoProdutos *c =
            criarCatalogo();
40      imprimirCatalogo(c);
41
42      adicionarProduto(c, "COMPUTADOR",
            2000.50, 3);
43      adicionarProduto(c, "CELULAR",
            1500.00, 2);
44      adicionarProduto(c, "TV", 4000.00,
            1);
45
46      imprimirCatalogo(c);
47
48      printf("ESTOQUE: %d\n",
            verificarEstoque(c, "CELULAR"));
49      return 0;
50  }
```

../ex12/CatalogoProduto.c

```
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/pr2/ex12$ ./main
COMPUTADOR
2000.50
3

CELULAR
1500.00
2

TV
4000.00
1

ESTOQUE: 2
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/pr2/ex12$
```

Figura 2: Saída

2 Análise de Complexidade

2.1

$$insercao = 8n^2$$

$$intercalacao = 64n \cdot lg(n)$$

Podemos analisar a seguinte inequação

$$insercao \geq intercalacao$$

$$8n^2 \geq 64n \cdot lg(n)$$

$$n^2 \geq 8n \cdot lg(n)$$
$$\frac{8}{n} \geq lg(n) \forall n \geq 44$$

2.2

$f(n) = 100n^2$
$g(n) = 2^n$
Portanto,

$$f(n) \leq g(n)$$
$$100n^2 \leq 2^n$$
$$lg(100n^2) \leq lg(2^n)$$
$$lg(100) + 2lg(n) \leq nlg(2)$$
$$lg(100) + 2lg(n) \leq n$$
$$2lg(10) + 2lg(n) \leq n$$
$$6.64 + 2lg(n) \leq n$$

É fácil perceber que $f(n)$ é pior que $g(n)$ $\forall n \in [15, \infty)$

Sejam f e g duas funções de complexidade quaisquer em $\mathbb{R}$.

2.3

Definição 1. *Notação O: Limite assintótico superior*
$\exists c, m \in \mathbb{R}^+, |g(n)| \leq c \cdot |f(n)| \forall n > m \Rightarrow g(n) = O(f(n))$

Para qualquer valor de $n \geq m$, g sempre terá imagem menor que a imagem de $c \cdot f$ para tais constantes m e c.

2.4

Definição 2. *Notação Ω: Limite assintótico inferior*
$\exists c, m \in \mathbb{R}^+, |g(n)| \geq c \cdot |f(n)| \forall n > m \Rightarrow g(n) = \Omega(f(n))$

Para qualquer valor de $n \geq m$, g sempre terá imagem maior que a imagem de $c \cdot f$ para tais constantes m e c.

2.5

A notação O é capaz de descrever o "teto"de desempenho de um algoritmo, mas não seu limite inferior. Dessa maneira, só é possível dizer: *O tempo de execução do algoritmo A é no máximo* $O(n^2)$.

2.6

$a(n) = n^2 - n + 500$
$b(n) = 47n - 47$
Fazemos,

$$a(n) \leq b(n)$$
$$n^2 - n + 500 \leq 47n - 47$$
$$n^2 - 48n + 453 \leq 0$$

Resolvendo para a igualdade, obtemos $n_1 = 35,09$ e $n_2 = 12,91$. Como n é obrigatoriamente pertencente aos inteiros positivos, temos:

$$a(n) \leq b(n) \Leftrightarrow n \in [13, 35]$$

2.7

$\sum_{i=0}^{N-1} \sum_{j=i}^{N} \sum_{k=1}^{j} s$
$= \sum_{i=0}^{N-1} \sum_{j=i}^{N} js$
$= \sum_{i=0}^{N-1} s(\sum_{j=i}^{N} j - \sum_{j=1}^{i-1} j)$
$= s \sum_{i=0}^{N-1} \frac{N^2-N}{2}$
$= sN \frac{N^2-N}{2}$
$= s \frac{N^3-N}{2}$
$= O(N^3)$

2.8

É necessário percorrer os $n-1$ elementos do array e efetuar a operação lógica em todas as iterações.

Logo, $O(n)$.

Como sempre é necessário passar pelos $n-1$ elementos, temos $\Omega(n)$. Se ele é limitado superiormente e inferiormente pela mesma função, sabemos que $\exists c \in \mathbb{R}$ que multiplica torna $O(n)$ em $\Omega(n)$ e $\Omega(n)$ em $O(n)$.

$$g(n) = O(f(n)) = \Omega(f(n)) \Leftrightarrow g(n) = \Theta(f(n))$$