UFRJ

INSTITUTO DE COMPUTAÇÃO
UFRJ

Programação de Computadores II

Laboratório -
Fila e Lista Ligada

Profa. Giseli Rabello Lopes

Instruções para resolução dos exercícios envolvendo Fila

- Sempre que possível, faça uso das funções já definidas referentes às operações básicas sobre a fila, vistas em aula, para resolução dos exercícios a seguir.
- Caso seja necessário, pode-se realizar mudanças para a fila armazenar outros tipos de dados.

Implementação da Fila

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

#define true 1
#define false 0

typedef int bool;

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
  //outros campos...
} REGISTRO;
typedef struct aux {
  REGISTRO reg;
  struct aux* prox;
} ELEMENTO;

typedef ELEMENTO* PONT;

typedef struct {
  PONT inicio;
  PONT fim;
} FILA;

void inicializarFila(FILA* f) {
  f->inicio = NULL;
  f->fim = NULL;
}

int tamanho(FILA* f) {
  PONT end = f->inicio;
  int tam = 0;
  while (end != NULL) {
    tam++;
    end = end->prox; }
return tam; }
```

```c
void exibirFila(FILA* f) {
  PONT end = f->inicio;
  printf("Fila: \" ");
  while (end != NULL) {
    printf("%i ", end->reg.chave);
    end = end->prox; }
  printf("\"\n"); }

bool inserirNaFila(FILA* f,REGISTRO reg) {
  PONT novo = (PONT) malloc(sizeof(ELEMENTO));
  novo->reg = reg;
  novo->prox = NULL;
  if (f->inicio==NULL) f->inicio = novo;
  else f->fim->prox = novo;
  f->fim = novo;
  return true;
}

bool excluirDaFila(FILA* f, REGISTRO* reg) {
  if (f->inicio==NULL) return false;
  *reg = f->inicio->reg;
  PONT apagar = f->inicio;
  f->inicio = f->inicio->prox;
  free(apagar);
  if (f->inicio == NULL) f->fim = NULL;
  return true; }

void reinicializarFila(FILA* f) {
  PONT end = f->inicio;
  while (end != NULL) {
    PONT apagar = end;
    end = end->prox;
    free(apagar);
  }
  f->inicio = NULL;
  f->fim = NULL;
}
```

Instruções para resolução dos exercícios envolvendo Lista Ligada

- Sempre que possível, faça uso das funções já definidas referentes às operações básicas sobre a lista ligada, vistas em aula, para resolução dos exercícios a seguir.
- Caso seja necessário, pode-se realizar mudanças para a lista ligada armazenar outros tipos de dados.

Implementação da Lista Ligada

```c
#include <stdio.h>
#include <stdlib.h>
#define true 1
#define false 0

typedef int bool;

typedef int TIPOCHAVE;

typedef struct {
  TIPOCHAVE chave;
  // outros campos...
} REGISTRO;

typedef struct aux {
  REGISTRO reg;
  struct aux* prox;
} ELEMENTO;

typedef ELEMENTO* PONT;

typedef struct {
  PONT inicio;
} LISTA;

void inicializarLista(LISTA* l){
  l->inicio = NULL;
}

int tamanho(LISTA* l) {
  PONT end = l->inicio;
  int tam = 0;
  while (end != NULL) {
    tam++;
    end = end->prox;
  }
return tam; }

void exibirLista(LISTA* l) {
  PONT end = l->inicio;
  printf("Lista: \" ");
  while (end != NULL) {
    printf("%i ", end->reg.chave);
    end = end->prox; }
  printf("\"\n"); }

bool insere(LISTA* l, REGISTRO reg, int pos) {
if (pos<0 || pos>tamanho(l)) return false;
ELEMENTO* novo = (ELEMENTO*) malloc(sizeof(ELEMENTO));
novo->reg = reg;
int i;
ELEMENTO* p;
if (pos == 0){
  novo->prox = l->inicio;
  l->inicio = novo;
}else{
  p = l->inicio;
  for (i=0;i<pos-1;i++) p = p->prox;
  novo->prox = p->prox;
  p->prox = novo;}
return true; }

bool exclui(LISTA* l, int pos) {
  if (pos<0 || pos>tamanho(l)-1) return false;
  int i;
  ELEMENTO* p;
  ELEMENTO* apagar;
  if (pos == 0) {
    apagar = l->inicio;
    l->inicio = apagar->prox;
  }else {
    p = l->inicio;
    for (i=0;i<pos-1;i++) p = p->prox;
    apagar = p->prox;
    p->prox = apagar->prox; }
  free(apagar);
  return true; }


void reinicializarFila(LISTA* l) {
  PONT end = l->inicio;
  while (end != NULL) {
    PONT apagar = end;
    end = end->prox;
    free(apagar); }
  l->inicio = NULL;
}
```

Exercício 1

- Escreva um algoritmo que leia um número indeterminado de valores inteiros. O valor 0 (zero) finaliza a entrada de dados. Para cada valor lido, determinar se ele e um número par ou ímpar. Se o numero for par, então incluí-lo na FILA PAR; caso contrário, incluí-lo na FILA ÍMPAR. Após o término da entrada de dados, retirar um elemento de cada fila alternadamente (iniciando-se pela FILA ÍMPAR) até que ambas as filas estejam vazias. Seu programa deve exibir os elementos na ordem em que estão sendo retirados das filas.

Exercício 2

- Um mercado possui 1 caixa de pagamento em que os clientes são atendidos por ordem de chegada na fila. No início, a fila está vazia. **`C`** clientes estão a efetuar compras neste supermercado. Cada cliente é identificado por um nome, um tempo de chegada (em segundos) ao caixa e número de produtos para pagar. Um cliente com **`P`** produtos demora **`10+P*K`** segundos a ser atendido na caixa, onde **`K`** é uma constante que define quantos segundos o operador da caixa demora a registrar um único produto, e **`10`** é o tempo constante necessário para o cliente efetuar o pagamento.
- Considera-se que os clientes nunca chegam exatamente no mesmo segundo ao caixa.

Exercício 2

- **Entrada:** A primeira linha contém um inteiro que representa **`K`**, a rapidez de atendimento da caixa. A segunda linha contém um inteiro **`C`**, o número de clientes. Seguem-se **`C`** linhas, cada uma contendo a descrição de um cliente no formato **`"NOME SEGUNDO_CHEGADA NUMERO_PRODUTOS"`**, onde **`NOME`**, é uma sequência contíguas de caracteres (sem espaços) e **`SEGUNDO_CHEGADA`** e **`NUMERO_PRODUTOS`** números inteiros indicando, respectivamente, o segundo de chegada e o número de produtos do cliente. É garantido que os clientes são informados por ordem estritamente crescente do segundo de chegada. As informações sobre cada cliente devem ser armazenadas em uma estrutura de fila.
- **Saída:** Devem ser escritas **`C`** linhas, uma por cada cliente, no formato **`"NOME TEMPO_CHEGADA TEMPO_SAIDA"`**, onde o **`TEMPO_SAIDA`** é o segundo onde o cliente termina o seu atendimento. Após todos os dados de entrada terem sido informados pelo usuário, a saída deve ir sendo escrita conforme os elementos forem sendo retirados da fila.

Exercício 2

- **Exemplo:**

Entrada	Saída
2 3 Renata 12 4 Ana 25 1 Matilde 45 4	Renata 12 30 Ana 25 42 Matilde 45 63

Exercício 3

- Implemente as seguintes funções, sobre Lista Ligadas:
  A. **`void concatenaListas(LISTA* l1, LISTA* l2)`** - concatena duas listas (isto é, "amarra" a segunda no fim da primeira).
  B. **`bool iguaisListas(LISTA* l1, LISTA* l2)`** - verifica se duas listas dadas são iguais, ou seja, se têm o mesmo conteúdo.
  C. **`bool modificaElem(LISTA* l, REGISTRO reg, int pos)`** - modifica o registro da lista na posição **`pos`** (seus valores).
- Escreva um programa principal para testar as funções criadas.