```c
/*****************************************************************************
//     filaDinamica.c
// Este programa gerencia filas lineares ligadas (implementacao dinamica).
// As filas gerenciadas podem ter um numero arbitrario de elementos.
// Não usaremos sentinela ou cabeça nesta estrutura.
*****************************************************************************/
#include <stdio.h>
#include <malloc.h>
#define ERRO -1
#define true 1
#define false 0
typedef int bool;
typedef int TIPOCHAVE;

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

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

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

/* Inicialização da fila ligada (a fila jah esta criada e eh apontada
pelo endereco em f) */
void inicializarFila(FILA* f){
  f->inicio = NULL;
  f->fim = NULL;
} /* inicializarFila */

/* Retornar o tamanho da fila (numero de elementos) */
int tamanho(FILA* f) {
  PONT end = f->inicio;
  int tam = 0;
  while (end != NULL){
    tam++;
    end = end->prox;
  }
  return tam;
} /* tamanho */

/* Retornar o tamanho em bytes da fila. Neste caso, isto depende do numero
   de elementos que estao sendo usados.   */
int tamanhoEmBytes(FILA* f) {
  return (tamanho(f)*sizeof(ELEMENTO)) + sizeof(FILA);
} /* tamanhoEmBytes */

/* Destruição da fila
   libera a memoria de todos os elementos da fila*/
void destruirFila(FILA* f) {
  PONT end = f->inicio;
  while (end != NULL){
    PONT apagar = end;
    end = end->prox;
    free(apagar);
  }
  f->inicio = NULL;
  f->fim = NULL;
} /* destruirFila */


/* retornarPrimeiro - retorna o endereco do primeiro elemento da fila e (caso
   a fila nao esteja vazia) retorna a chave desse elemento na memoria
   apontada pelo ponteiro ch */
PONT retornarPrimeiro(FILA* f, REGISTRO *ch){
  if (f->inicio != NULL) *ch = f->inicio->reg;
  return f->inicio;
} /* retornarPrimeiro */

/* retornarUltimo - retorna o endereco do ultimo elemento da fila e (caso
   a fila nao esteja vazia) retorna a chave desse elemento na memoria
   apontada pelo ponteiro ch */
PONT retornarUltimo(FILA* f, TIPOCHAVE* ch){
  if (f->inicio == NULL) return NULL;
  *ch = f->fim->reg.chave;
  return f->fim;
} /* retornarUltimo */
```

```
85
86
87  /* Inserção no fim da fila */
88  bool inserirNaFila(FILA* f,REGISTRO reg) {
89    PONT novo = (PONT) malloc(sizeof(ELEMENTO));
90    novo->reg = reg;
91    novo->prox = NULL;
92    if (f->inicio==NULL){
93       f->inicio = novo;
94    }else{
95       f->fim->prox = novo;
96    }
97    f->fim = novo;
98    return true;
99  } /* inserir */
100
101 REGISTRO inserirNaFila2(FILA* f, char tipo) {
102   PONT novo = (PONT) malloc(sizeof(ELEMENTO));
103   if(f->fim == NULL) {
104     novo->reg.chave = 1;
105   } else {
106      novo->reg.chave = f->fim->reg.chave + 1;
107   }
108   novo->reg.tipo = tipo;
109   novo->prox = NULL;
110   if (f->inicio==NULL){
111      f->inicio = novo;
112   }else{
113      f->fim->prox = novo;
114   }
115   f->fim = novo;
116   return novo->reg;
117 } /* inserir */
118
119 /* Excluir  */
120 bool excluirDaFila(FILA* f, REGISTRO* reg) {
121   if (f->inicio==NULL){
122     return false;
123   }
124   *reg = f->inicio->reg;
125   PONT apagar = f->inicio;
126   f->inicio = f->inicio->prox;
127   free(apagar);
128   if (f->inicio == NULL){
129     f->fim = NULL;
130   }
131   return true;
132 } /* excluirDaFila */
133
134
135 /* Exibição da fila sequencial */
136 void exibirFila(FILA* f){
137   PONT end = f->inicio;
138   printf("Fila: \" ");
139   while (end != NULL){
140     printf("%c%d ", end->reg.tipo ,end->reg.chave); // soh lembrando TIPOCHAVE = int
141     end = end->prox;
142   }
143   printf("\"\n");
144 } /* exibirFila */
145
146 /* Busca sequencial */
147 PONT buscaSeq(FILA* f,TIPOCHAVE ch){
148   PONT pos = f->inicio;
149   while (pos != NULL){
150     if (pos->reg.chave == ch) return pos;
151     pos = pos->prox;
152   }
153   return NULL;
154 } /* buscaSeq */
155
156 /* Busca sequencial com sentinela alocado dinamicamente */
157 PONT buscaSeqSent1(FILA* f,TIPOCHAVE ch){
158   if (!f->inicio) return NULL;
159   PONT sentinela = malloc(sizeof(ELEMENTO));
160   sentinela->reg.chave = ch;
161   f->fim->prox = sentinela;
162   PONT pos = f->inicio;
163   while (pos->reg.chave != ch) pos = pos->prox;
164   free(sentinela);
165   f->fim->prox = NULL;
166   if (pos!=sentinela) return pos;
167   return NULL;
168 } /* buscaSeqSent1 */
```

```
169
170   /* Busca sequencial com sentinela como variavel local */
171   PONT buscaSeqSent2(FILA* f,TIPOCHAVE ch){
172     if (!f->inicio) return NULL;
173     ELEMENTO sentinela;
174     sentinela.reg.chave = ch;
175     f->fim->prox = &sentinela;
176     PONT pos = f->inicio;
177     while (pos->reg.chave != ch) pos = pos->prox;
178     f->fim->prox = NULL;
179     if (pos!=&sentinela) return pos;
180     return NULL;
181   } /* buscaSeqSent1 */
```