Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Estruturas de Dados

LISTA DE EXERCÍCIOS 5

Exercício 1. Adapte os códigos `sllist.h` e `sllist.c` para criar os códigos `dllist.h` e `dllist.c`, implementando uma lista *duplamente* encadeada (*Doubly Linked List*). Escreva também um programa `main.c` para testar as funções implementadas.

Exercício 2. Uma lista *circular* duplamente encadeada (*Circular Doubly Linked List*) é uma lista duplamente encadeada com a diferença de que sucede o último nó da lista o primeiro, assim como antecede o primeiro nó da lista o último. Adapte os códigos `dllist.h` e `dllist.c` para criar os códigos `cdllist.h` e `cdllist.c`, implementando uma lista circular duplamente encadeada. Escreva também um programa `main.c` para testar as funções implementadas.

Exercício 3. Adapte todas as suas bibliotecas implementadas até aqui para generalizar o tipo da chave armazenada nos nós das estruturas, de modo que um nó armazena nada mais que um ponteiro do tipo `void *` para o endereço de memória onde a chave está armazenada. Agora, perceba que, nas funções que imprimem a lista, por exemplo, não há como saber como as chaves devem ser impressas. A solução para este problema é deixar a cargo de quem usa a biblioteca que forneça para a função de impressão da lista uma função que trate da impressão de chave. Em C, isto é possível fazendo a função que imprime a lista receber como um de seus parâmetros um ponteiro para a função que imprime chave, como no exemplo a seguir do novo `sllist.h`. Outras funções das suas bibliotecas podem precisar receber funções como parâmetro.

```
#ifndef _SLLIST
#define _SLLIST
```

```
typedef struct _lnode {
  void *key;
  struct _lnode *next;
} lnode;

typedef struct {
  lnode *head;
} sllist;

void sllist_init(sllist* l);

int sllist_insert(sllist *l, void *key);

void sllist_erase(sllist *l, void *key);

void sllist_print(sllist *l, void (*print_key)(void *));

void sllist_free(sllist *l, void (*free_key)(void *));

#endif
```

Exercício 4. Como explicado em sala de aula, listas, filas, e pilhas podem ser implementadas também em *arrays* dinâmicos. Adapte todos os códigos desenvolvidos, criando versões em que as estruturas são implementadas em *arrays* alocados dinamicamente. Escreva um ou mais programas para testar as funções implementadas por todas as suas bibliotecas.