FACULDADE INTEGRADA
DA GRANDE FORTALEZA

ESTRUTURA DE DADOS

elidiane@fgf.edu.br

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

Objetivo

ALOCAÇÃO DINÂMICA DE MEMÓRIA

- A melhor solução para problemas onde a **quantidade de dados pode variar drasticamente** entre várias execuções de um programa.
- Na alocação estática o espaço a ser alocado é conhecido em **tempo de programação**, na alocação dinâmica é conhecido em **tempo de execução**.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Alocação Dinâmica

- A alocação de memória em C é feita por meio de ponteiros e de quatro funções disponíveis na biblioteca stdlib.h.
  - **malloc(): Aloca um número especificado de bytes em memória e retorna um ponteiro para o início do bloco alocado;**
  - calloc(): Similar ao malloc, mas a função inicia todos os bytes alocados com zero. Também permite a alocação de memória para mais de um bloco numa mesma chamada;
  - realloc(): modifica o tamanho de um bloco previamente alocado dinamicamente;
  - free(): Libera o espaço de um bloco de memória previamente alocado com malloc(), calloc() ou realloc() .

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Alocação Dinâmica

- malloc() – Retorna um endereço de memória
  - malloc (tamanho em byte do espaço necessário)
  - **ponteiroparatipo = malloc (sizeof(tipo));**
  - A função malloc anterior aloca um espaço do tamanho suficiente para armazenar o “tipo” e retorna para o “ponteiroparatipo” o endereço da primeira posição de memória alocada;
  - Quando a função não consegue alocar espaço em memória para o bloco solicitado, ela retorna **NULL**;

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Alocação Dinâmica

- free()
  - Recebe como argumento um ponteiro que aponta para uma posição de memória:
    - void free (void *free);

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Alocação Dinâmica

- É comum encontrar a seguinte sintaxe:

- Ptrqualquer = malloc(sizeof(tipoqualquer));
- Ptrqualquer = (tipoqualquer *) malloc(sizeof(tipoqualquer));

- Ambas operações possuem o mesmo resultado. No entanto, a segunda situação especifica o tipo, enquanto a primeira trata um tipo genérico de retorno (void *);

```c
//este é o lista.h

struct no{
  int dado;
  struct no *prox;
};

typedef struct no Lista;

//cria lista
Lista* crialista();

//insere na lista
Lista * insere(Lista *l, int x);

//retira um elemento da lista
Lista* retira(Lista *l, int valor);

//imprime elementos da lista
void imprime(Lista *l);
```

```c
/* lista.c exemplo das funções:
      - cria lista;
      - insere;
      - imprime;
*/
#include "lista.h"
#include <stdlib.h>
#include <stdio.h>

Lista* crialista(){
  return NULL;
}

Lista * insere(Lista *l, int x){
  Lista * novono = (Lista *) malloc(sizeof(Lista));
  novono->dado = x;
  novono->prox = l;
  return novono;
}


void imprime(Lista *l){
  Lista *aux;
  for (aux=l; aux!=NULL; aux=aux->prox)
    printf("%d ",aux->dado);
  printf("/n");
}
```

```c
/* programa cliente.c
A saída será: 3 5 15 10
Observe que a lista insere no começo.
*/

#include "lista.h"
#include <stdio.h>

int main(){
  Lista *l;
  l = crialista();
  l = insere(l, 10);
  l = insere(l, 15);

  l = insere(l, 5);
  l = insere(l, 3);

  imprime(l);
}
```

```
MacBook-Air-de-Elidiane:Lista-Dinamica elidianemartins$ gcc -c lista.c
MacBook-Air-de-Elidiane:Lista-Dinamica elidianemartins$ gcc -c cliente.c
MacBook-Air-de-Elidiane:Lista-Dinamica elidianemartins$ gcc -o prog cliente.o lista.o
MacBook-Air-de-Elidiane:Lista-Dinamica elidianemartins$ ./prog
3 5 15 10
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Exercício

Implemente função remove emento:

- **Função remove: Recebe dois parâmetros, um endereço da lista e o valor a ser removido;**

Primeiro deve verificar se o elemento existe na lista (implementar uma função BUSCA que retorna NULL caso não exista ou o endereço do elemento, caso exista);

Caso exista, e se o elemento estiver no início da lista, o próximo elemento passa a ser o início da lista;

Se o elemento estiver no final da lista, basta liberar o espaço de memória ocupada pelo endereço l->prox=NULL;

Se o elemento estiver no meio da lista, deve-se refazer os apontamentos. Para tanto, no ato da busca do elemento, é importante armazenar o endereço do **elemento anterior**;

elementoAnterior->prox = l->prox;

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Exercício

Implemente função remove emento:

- **Função remove:**
- **Função insere no final da lista.**