Prova 1 - Data 10/10/2019

Conteúdo:

- **Programação Básica em C:**
  Estruturas de Controle;
  Funções;
  Ponteiros e Endereços;
- **Listas Lineares**
  Sequenciais
  Ligadas
  - Implementação Estática;
  - Implementação Dinâmica;
  - Técnicas Especiais: Cabeça, Sentinela, Circularidade, Encadeamento Duplo;

  Filas
  - Implementação Estática;
  - Implementação Dinâmica;

  Deques (Filas de Duas Pontas)
  - Implementação Dinâmica;

  Pilhas
  - Implementação Estática;
  - Implementação Dinâmica;
  - Implementação de Duas Pilhas em uma única estrutura

Referências Principais:

Notas de Aula

Apostila ACH2023 - ALGORITMOS E ESTRUTURAS DE DADOS I, *Willian Yukio Honda e Ivandré Paraboni*

Lista de Exercícios / Exemplo de Prova:

No seguinte link é apresentado um exemplo de prova que foi aplicada em ACH2023: **[pdf]**

Um ÓTIMO exercício é implementar e testar os algoritmos da apostila e das notas de aula (que estão disponíveis no site da disciplina), principalmente as funções que não foram implementadas em aula e entender a definição, pontos fracos e fortes de cada implementação de cada estrutura de dados.

Exemplo de questões para a prova

Pergunta 1. Dado o seguinte programa:

```
#include <stdio.h>

typedef int * pontInt;

int main(){
  pontInt *x;
  pontInt y;
  int z;
  x=&y;
  y=&z;
  z = 5;
  *y = 7;
  **x = 9;

  printf("x: %p\n",x);
```

```
  printf("*x: %p\n",*x);
  printf("**x: %i\n",**x);
  printf("&x: %p\n",&x);
  return 0;
}
```

Qual é o resultado impresso pelo programa, sabendo-se que: (i) a variável x foi criada no endereço 3608; (ii) a variável y foi criada no endereço 3604; e (iii) a variável z foi criada no endereço 3600.

Pergunta 2. Dado o seguinte programa:

```
#include <stdio.h>
#include <malloc.h>

typedef struct tempR{
  int valor;
  struct tempR * ant;
  struct tempR * prox;
} Registro;

Registro * criarRegistro(int val){
   Registro* novo = (Registro*) malloc(sizeof(Registro));
   novo->prox = NULL;
   novo->ant = NULL;
   novo->valor = val;
   return novo;
}

int main(){
  Registro* primeiro = NULL;
  primeiro = criarRegistro(10);
  primeiro->prox = criarRegistro(12);
  primeiro->prox->ant = primeiro;
  Registro* temp;
  temp = primeiro;
  primeiro = criarRegistro(15);
  primeiro->prox = temp;
  primeiro->prox->ant = primeiro;

  printf("(*primeiro).prox->valor: %i\n",(*primeiro).prox->valor);
  printf("primeiro->prox->valor: %i\n",primeiro->prox->valor);
  printf("primeiro->prox->prox->valor: %i\n",primeiro->prox->prox->valor);
  printf("primeiro->valor: %i\n",primeiro->valor);
  printf("primeiro: %p\n", primeiro);
  printf("primeiro->prox->ant: %p\n",primeiro->prox->ant);
  printf("primeiro->prox->prox: %p\n",primeiro->prox->prox);

  return 0;
}
```

Qual é o resultado impresso pelo programa, sabendo-se que: (i) o registro com valor=10 foi gravado no endereço 3528; (ii) o registro com valor=12 foi gravado no endereço 3552; e (iii) o registro com valor=15 foi gravado no endereço 3576;

Pergunta 3. Suponha que há um problema para se manter uma lista ordenada de alunos da USP. Duas soluções foram propostas uma utilizando uma lista ligada implementação dinâmica e a outra utilizando lista sequencial (estática). Diga uma vantagem e uma desvantagem de cada uma das propostas.

Pergunta 4. Definimos uma versão alternativa de um deque como uma estrutura duplamente ligada (sem nós cabeças), onde é possível inserir ou remover os elementos de qualquer uma de suas entradas (acessíveis pelos endereços inicio1 e inicio2, que terão valor NULL [ambos] caso o deque esteja vazio). Utilizamos as definições abaixo:

```
typedef int TIPOCHAVE;

typedef struct tempRegistro{
```

```
  TIPOCHAVE chave;
  struct tempRegistro *prox;
  struct tempRegistro *ant;
} REGISTRO;

typedef REGISTRO* PONT;

typedef struct {
  PONT inicio1;
  PONT inicio2;
} DEQUE;
```

Implemente os métodos *inserirDeque1* e *excluirElemDeque2* conforme as definições abaixo:

```
/* Inserção no deque, entrada 1 */
bool inserirDeque1(REGISTRO reg, DEQUE *l) e

/* Exclusão do primeiro elemento a partir da entrada 2 do deque
   e cópia do valor da chave do elemento na memória apontada por ch*/
bool excluirElemDeque2(DEQUE *l, TIPOCHAVE * ch){
```

Pergunta 5. Utilizando as definições da estrutura deque do exercício anterior, este exercício apresenta duas funções *inicializarDeque*, mas apenas uma funcionará corretamente (quando chamada por outro progama ou função). Qual? Justifique sua escolha.

```
// Primeira
void inicializarDeque(DEQUE *d){
  d->inicio1 = NULL;
  d->inicio2 = NULL;
} /* inicializarDeque */

// Segunda
void inicializarDeque(DEQUE d){
  d.inicio1 = NULL;
  d.inicio2 = NULL;
} /* inicializarDeque */
```

Respostas

Resposta 1:

```
x: 3604
*x: 3600
**x: 9
&x: 3608
```

Resposta 2:

```
(*primeiro).prox->valor: 10
primeiro->prox->valor: 10
primeiro->prox->prox->valor: 12
primeiro->valor: 15
primeiro: 3576
primeiro->prox->ant: 3576
primeiro->prox->prox: 3552
```

Resposta 3 (exemplo de resposta possível):
Solução Estática. Vantagem: permite a busca binária. Desvantagem: é necessário definir um número máximo de registros e a memória alocada para esses registros é alocada na criação da lista (mesmo que não exista nenhum registro utilizado).

Solução Dinâmica. Vantagem: só é alocado espaço para registros válidos (usados); não existe limite para o número de registros. Desvantagem: não é possível fazer a busca binária.

Resposta 4 modifique a implementação no site da disciplina.

Resposta 5 a primeira função inicializarDeque irá funcionar pois ele recebe o endereço de um deque e modifica os valores de inicio1 e inicio2 do deque presente nesse endereço. A segunda função não funciona porque recebe como parâmetro uma cópia de um DEQUE e altera apenas os valores das variáveis inicio1 e inicio2 da cópia, não afetando os dados do DEQUE original.