Lista de Exercícios – Alocação Dinâmica e TAD

Estrutura de Dados 1

Prof. Paulo Henrique Ribeiro Gabriel

Para resolver esta lista, você deve estudar alocação dinâmica de memória e registros (`struct`). Todos esses conceitos (e diversos outros!) estão disponíveis nas videoaulas presentes no seguinte link: <encurtador.com.br/cfhLV>.

1. Escreva uma função que aloque dinamicamente uma matriz de números inteiros. Essa função deve ter o seguinte protótipo:

   ```
   int **alocaMatriz(int nCol, int nRow);
   ```

   sendo que: `nCol` é o número de colunas da matriz e `nRow` é o número de linhas da matriz.

2. Escreva uma função que apague a matriz alocada pela função do exercício anterior. Utilize o seguinte protótipo:

   ```
   void apagaMatriz(int **M, int nCol, int nRow);
   ```

3. Escreva uma função que imprima os dados da matriz `M`. Utilize o seguinte protótipo:

   ```
   void imprimeMatriz(int **M, int nCol, int nRow);
   ```

4. Considere, agora, uma matriz representada por meio do seguinte registro:

   ```
   struct matriz {
       int lin;
       int col;
       float *v;
   };

   typedef struct matriz Matriz;
   ```

Considere, ainda, o conjunto de funções mostrado a seguir:

```
Matriz* cria(int m, int n)
{
  Matriz* mat = (Matriz *) malloc(sizeof(Matriz));
  if (mat == NULL) {
    printf("Memoria insuficiente!\n");
    exit(1);
  }
  mat->lin = m;
  mat->col = n;
  mat->v = (float *) malloc(m * n * sizeof(float));
  return mat;
}

void libera(Matriz* mat)
{
  free(mat->v);
  free(mat);
}

float acessa(Matriz* mat, int i, int j) {
  int k;

  if (i < 0 || i >= mat->lin || j < 0 || j >= mat->col) {
printf("Acesso invalido!\n");
exit(1);
  }
  k = i*mat->col + j;

  return mat->v[k];
}
```

(a) Explique o funcionamento de cada uma das funções (`cria`, `libera` e `acessa`) mostradas anteriormente. Faça testes de mesa (ou seja, invente alguns valores e teste, em papel, cada uma dessas funções a fim de entender seu funcionamento).

(b) Crie uma função `atribua` que insere um valor `v` qualquer na matriz. Essa função deve respeitar o seguinte cabeçalho:

```
void atribui(Matriz *mat, int i, int j, float v)
```

(c) Escreva um programa (função `main`) que execute as seguintes operações: leia as dimensões de uma matriz (número de linhas e de colunas) crie uma matriz com

essas dimensões, insira alguns valores na matriz, imprima essa matriz e, finalmente, libere a memória alocada. As dimensões da matriz e os valores a serem inseridos serão dados pelo usuário. Para manipular sua matriz, utilize **apenas** as funções criadas anteriormente (ou seja, `cria`, `libera`, `acessa` e `atribui`).

5. Escreva um programa que organize os dados pessoais dos alunos da UFU: nome, ano de ingresso e cidade natal.

   (a) Crie uma estrutura de dados do tipo `struct Aluno` para armazenar esses dados.

   (b) Aloque dinamicamente um vetor de `n` (valor definido pelo usuário) elementos do tipo `Aluno`.

   (c) Escreva uma função que leia, da entrada padrão, os dados referentes a `n` alunos.

   (d) Escreva uma função que imprima os dados dos `n` alunos na saída padrão.

6. Sobre a estrutura de dados do tipo `union`:

   (a) Explique o seu funcionamento.

   (b) Dadas as definições abaixo:

```
typedef enum{FLOAT, INT} tipoDado;

typedef union {
    int inteiro;
    float real;
} numero;

typedef struct {
    numero n;
    tipoDado tipo;
} meu_dado;

meu_dado dados[5];
```

   Escreva um programa que leia 5 números (o usuário escolhe o tipo do dado para cada elemento do vetor) e, a seguir, imprima esses 5 números.