Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

Professora: Rúbia Eliza de Oliveira Schultz Ascari
Departamento Acadêmico de Informática (Dainf)
Tecnologia em Análise e Desenvolvimento de Sistemas
Estruturas de Dados 1

Exercícios sobre Alocação Dinâmica

1) Crie um vetor do tipo inteiro com 50 itens valorizados aleatoriamente usando a função malloc. Percorra o vetor imprimindo endereço e valor de cada item.

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

int main(void){
    int *vet;
    int num = 50;
    int i;
    vet = (int *) malloc(num * sizeof(int));

   if (vet) {

         for (i = 0; i < num; i++) {
               vet[i] = rand();
         }

         for (i = 0; i < num; i++) {
               printf("Valor da posicao %d no vetor eh %d e o endereco eh
%ld\n", i, vet[i], &vet[i]);
         }

         free(vet);
   }

    return 0;
}
```

2) Repetir o exercício anterior usando a função calloc em vez de malloc, e inserindo o seu RA como conteúdo dos itens.

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

int main(void){
    int *vet;
    int num = 50;
    int i;
    vet = (int *) calloc(num, sizeof(int));

    if (vet) {
         for (i = 0; i < num; i++) {
               vet[i] = 12345;
         }

         for (i = 0; i < num; i++) {
               printf("Valor da posicao %d no vetor eh %d e o endereco eh
%ld\n", i, vet[i], &vet[i]);
```

```
            }

            free(vet);
        }

        return 0;
    }
```

3) Crie uma função que recebe o limite inferior e o limite superior de um intervalo e tem como objetivo retornar um vetor com todos os inteiros pertencentes ao intervalo.

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

int * geraVetor(int *vet, int inf, int sup)
{
    for (int i = 0; i < (sup - inf) + 1; i++)
    {
        if (i == 0)
        {
            vet = (int *) malloc(sizeof(int));
        }
        else
        {
            vet = (int *) realloc(vet, (i+1) * sizeof(int));
        }

        if (vet)
        {
            vet[i] = inf + i;
        }

    }
    return vet;
}

int main(void)
{
    int *vet;
    int inf, sup;

    printf("Informe um limite inferior e superior para criacao de um vetor:
");
    scanf("%d%d", &inf, &sup);

    vet = geraVetor(vet, inf, sup);

    if (vet)
    {
        for (int i = 0; i < (sup - inf) + 1; i++)
        {
            printf("Valor da posicao %d no vetor eh %d\n", i, vet[i]);
        }

        free(vet);
    }

    return 0;
}
```

4) Crie um programa que armazena um número dinâmico de notas escolares e retorna a média simples das notas, maior e menor nota.

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

int main(void)
{
    float *notas;
    int continua = 1;
    int i = 0;
    float media, maior, menor;

    while (continua == 1)
    {
        printf("\nInforme uma nota para o aluno %d: ", i+1);
        if (i == 0)
            notas = (float *) malloc(sizeof(float));
        else
            notas = (float *) realloc(notas, (i+1) * sizeof(float));
        if (notas)
        {
            scanf("%f", &notas[i]);
            media = 0;
            maior = 0;
            menor = notas[0];
            for (int j = 0; j <= i; j++)
            {
                media = media + notas[j];
                if (maior < notas[j])
                    maior = notas[j];
                if (menor > notas[j])
                    menor = notas[j];
            }
            media = media / (i+1);
            printf("\nA media das notas digitadas eh %f.\nA maior
nota eh %f.\n A menor nota eh %f.", media, maior, menor);
            printf("\nDeseja informar uma nova nota? (0-Nao/1-Sim)");
            scanf("%d", &continua);
            i++;
        }
    }

    if (notas)
        free(notas);

    return 0;
}
```

5) Crie uma estrutura dinâmica utilizando struct, para armazenar dados de RA, nome, telefone e email de alunos. O número de alunos e dados de cada um devem ser informados pelo usuário. Imprimir posteriormente os dados informados.

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

typedef struct aluno Aluno;
```

```
struct aluno {
    int RA;
    char nome[30];
    char telefone[20];
    char email[20];
};

int main(void){
    Aluno *alunos;
    int continua = 1;
    int i = 0;

    while (continua == 1) {
        printf("\nInforme os dados do aluno %d: ", i);
        if (i == 0)
            alunos = (Aluno *) malloc(sizeof(Aluno));
        else
            alunos = (Aluno *) realloc(alunos, (i+1) * sizeof(Aluno));
       if (alunos) {
            printf("\nRA: ");
            scanf("%d", &alunos[i].RA);
            printf("Nome: ");
            scanf(" %[^\n]s", alunos[i].nome);
            printf("Telefone: ");
            scanf(" %[^\n]s", alunos[i].telefone);
              printf("E-mail: ");
            scanf(" %[^\n]s", alunos[i].email);
            printf("\nDeseja informar dados de um novo aluno? (0-Nao/1-
Sim): ");
            scanf("%d", &continua);
          }
        i++;
    }

   if (alunos) {
         printf("\n\nDados dos alunos: ");
         for(int j = 0; j < i; j++) {
           printf("\n\nAluno %d: ", j);
           printf("\nRA: %d", alunos[j].RA);
           printf("\nNome: %s", alunos[j].nome);
           printf("\nTelefone: %s", alunos[j].telefone);
           printf("\nE-mail: %s", alunos[j].email);
         }
         free(alunos);
   }

    return 0;
}
```

6) O exercício tem como objetivo fazer a soma de duas matrizes. Para isso:
   - Criar uma função que recebe um número de linhas e um número de colunas, aloca a matriz dinamicamente e retorna o ponteiro.
   - Criar uma função que recebe uma matriz e imprime seus elementos.
   - Criar uma função que recebe uma matriz e a libera.
   - Criar uma função que recebe uma matriz e a valoriza com números aleatórios.
   - Criar uma função que recebe duas matrizes e gera uma terceira matriz que corresponde a soma das duas matrizes. Retorne o ponteiro.

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

```c
#include <stdlib.h>

int **alocaMatriz(int nLinhas, int nColunas);
void imprime(int **m, int nLinhas, int nColunas);
void liberaMatriz(int **m, int nLinhas);
void valorizaMatriz(int **m, int nLinhas, int nColunas);
int **somaMatrizes(int **mA, int **mB, int nLinhas, int nColunas);

int **alocaMatriz(int nLinhas, int nColunas) {
    int **m, l;
    m = malloc(nLinhas * sizeof(int*));
    for (l = 0; l < nLinhas; l++) {
        m[l] = calloc(nColunas, sizeof(int));
    }
    return m;
}

void imprime(int **m, int nLinhas, int nColunas) {
    int l, c;
    for (l = 0; l < nLinhas; l++) {
        for (c = 0; c < nColunas; c++) {
            printf("%d\t", m[l][c]);
        }
        printf("\n");
    }
}

void liberaMatriz(int **m, int nLinhas) {
    int l;
    for (l = 0; l < nLinhas; l++) {
        free(m[l]);
    }
    free(m);
}

void valorizaMatriz(int **m, int nLinhas, int nColunas) {
    int l, c;
    for (l = 0; l < nLinhas; l++) {
        for (c = 0; c < nColunas; c++) {
            m[l][c] = rand() % 10;
        }
    }
}

int **somaMatrizes(int **mA, int **mB, int nLinhas, int nColunas) {
    int l, c;
    for (l = 0; l < nLinhas; l++) {
        for (c = 0; c < nColunas; c++) {
            mA[l][c] = mA[l][c] + mB[l][c];
        }
    }
    return mA;
}

int main(void) {

    int nLinhas, nColunas, **m, **mNova;
    nLinhas = 4;
    nColunas = 5;
    m = alocaMatriz(nLinhas, nColunas);
    printf("Matriz inicial: \n");
```

```
    imprime(m, nLinhas, nColunas);
    valorizaMatriz(m, nLinhas, nColunas);
    printf("Matriz inicial valorizada: \n");
    imprime(m, nLinhas, nColunas);
    mNova = alocaMatriz(nLinhas, nColunas);
    valorizaMatriz(mNova, nLinhas, nColunas);
    printf("Matriz nova valorizada: \n");
    imprime(mNova, nLinhas, nColunas);
    mNova = somaMatrizes(m, mNova, nLinhas, nColunas);
    printf("Soma de matrizes inicial e nova: \n");
    imprime(mNova, nLinhas, nColunas);
    liberaMatriz(m, nLinhas);
    liberaMatriz(mNova, nLinhas);
    return 0;
}
```