Lista de Exercícios (Ponteiros e Alocação Dinâmica)

1) O que uma variável de ponteiro armazena?
2) Qual o tamanho (em bytes) de uma variável de ponteiro int e de uma variável de ponteiro char?
3) Por qual motivo uma variável de ponteiro, ao ser apresentado o seu conteúdo ao usuário, pode mostrar 0060FEF0 ou 000000000060FEF0?
4) Em qual área da memória serão alocadas as variáveis de ponteiro?
5) Em qual área da memória são alocados blocos de memória para uma alocação dinâmica?
6) Considerando o programa desenvolvido em linguagem C abaixo, e sabendo que o endereço de memória da variável x é 0060FF24 e o da variável ponteiro *ptr_x é 0060FF20, qual será o resultado exibido ao usuário após a execução do comando **printf("%d",*ptr_x);** na linha5?

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4   int x=22,*ptr_x=&x;
5   //printf aqui
6   getch();
7   return 0;
8 }
```

7) Sobre a questão anterior, o que seria apresentado ao usuário após a execução do comando **printf("%p",ptr_x);** na linha 5?
8) Ainda sobre a questão 6, o que seria apresentado ao usuário após a execução do comando **printf("%p",&ptr_x);** na linha 5?
9) Também sobre a questão 6, o que seria apresentado ao usuário após a execução do comando **printf("%p",&*ptr_x);** na linha 5?
10) Finalmente sobre a questão 6, o que seria apresentado ao usuário após a execução do comando **printf("%d", x);** na linha 5?
11) Qual o tipo que um ponteiro deve ser declarado? Explique.
12) Embora não recomendado, ao declararmos um ponteiro como *void*, o que será necessário para apresentar o valor ao qual ele está apontando?
13) Analisando o código abaixo desenvolvido em linguagem C, apresente o que será apresentado ao usuário após sua execução e justifique sua resposta.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4   int x=22,*ptr_x;
5   printf("%d\n",*ptr_x);
6   getch();
7   return 0;
8 }
```

14) Analisando o código abaixo desenvolvido em linguagem C, apresente o que será apresentado ao usuário após sua execução. Justifique sua resposta

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   int x=22;
   int *ptr_x=&x;
   *ptr_x=51;
   printf("%d\n",*ptr_x);
   getch();
   return 0;
}
```

15) Analisando o código abaixo desenvolvido em linguagem C, apresente o que será apresentado ao usuário após sua execução. Justifique sua resposta

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   int x=100, y=200;
   int *ptr=&x;
   x=100;
   printf("%d\n",*ptr);
   ptr=&y;
   printf("%d\n",*ptr);
   getch();
   return 0;
}
```

16) Qual o objetivo da função *sizeof* utilizando em linguagem C? Explique e dê um exemplo.

17) Considerando o código abaixo desenvolvido em linguagem C em um compilador de 64bits, apresente o que será apresentado ao usuário após sua execução. Justifique sua resposta

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   int idade,*ptr=&idade;
   printf("%d",sizeof(idade)+sizeof(*ptr));
   getch();
   return 0;
}
```

18) Construa um programa em linguagem C que solicite ao usuário o valor de uma variável do tipo *int* denominada **n1**. Crie um ponteiro identificado por **ptr** (também do tipo int) e faça a referência à variável n1. Posteriormente, apresente ao usuário (linha a linha):
   a) O valor da variável n1e memória da variável n1;
   b) O endereço de memória da variável n1;
   c) O valor da variável ptr;
   d) O endereço de memória da variável ptr;
   e) O valor ao qual a variável ptr está apontando;
   f) O endereço de memória da qual a variável ptr está apontando
   g) O tamanho em bytes da variável n1;
   h) O tamanho em bytes da variável ptr;
   i) O tamanho em bytes ao qual a variável ptr está apontando

19) Apresente que tipo de informações são alocadas nas seguintes áreas da memória:
   a) Stack
   b) Heap
   c) Executable Instructions
   d) Static Variable / Automatic Variable

20) Um código em linguagem C possui uma variável do tipo char identificada por **s** de tamanho 10 com o conteúdo "Moreno" e uma segunda variável de (ponteiro do tipo char), identificada por **ptr**, fazendo referência à variável **s.** Considerando que a variável ptr possua o valor 0060FF10, qual será o valor apresentado ao usuário ao executar o comando **printf("%p \n",&s[2]);**? Justifique sua resposta.

21) Analisando o código abaixo desenvolvido em linguagem C, apresente o resultado que será apresentado ao usuário após sua execução. Justifique sua resposta.

```c
#include <stdio.h>
#include <stdlib.h>
void troca(int a, int b){
int tmp;
  tmp=a;
  a=b;
  b=tmp;
}
//------------------------
int main() {
  int n1=10, n2=30;
  troca(n1,n2);
  printf("n1=%d\n",n1);
  printf("n2=%d\n",n2);
  getch();
  return 0;
}
```

22) Analisando o código abaixo desenvolvido em linguagem C, muito semelhante ao anterior, apresente o resultado que será apresentado ao usuário após sua execução. Justifique sua resposta.

```c
//Questao 22
#include <stdio.h>
#include <stdlib.h>
void troca(int *a, int *b){
int tmp;
  tmp=*a;
  *a=*b;
  *b=tmp;
}
//--------------------------
int main() {
  int n1=10, n2=30;
  troca(&n1,&n2);
  printf("n1=%d\n",n1);
  printf("n2=%d\n",n2);
  getch();
  return 0;
}
```

23) Considerando o programa abaixo, implemente um procedimento chamado altera que receba como parâmetros um vetor (por referência) e seu tamanho (por valor). Este procedimento deverá atualizar cada elemento do vetor atribuindo o dobro do seu valor.

```c
#include <stdlib.h>
#include <stdio.h>
void mostra(int k[], int tam){
int i;
for (i=0; i<tam; i++)
    printf("%d ",k[i]);
  printf("\n");
}
//-------------------------------------
//aqui será desenvolvido o procedimento
//-------------------------------------
int main() {
  int vet[]={12,15,14,45};
  int tam=sizeof(vet)/sizeof(int);
  mostra(vet,tam);
  altera(vet,tam);
  mostra(vet,tam);
  printf("\n\n");
  system("pause");
  return 0;
}
```

24) Construa um programa em linguagem C que solicite do usuário 5 valores inteiros e os armazene em um vetor. Posteriormente, apresente ao usuário utilizando a técnica de aritmética de ponteiro.

25) Construa um programa em linguagem C que solicite 4 palavras com no máximo 10 caracteres (armazenando-as em um vetor) e posteriormente apresente:
   a) O tamanho do vetor (em bytes);
   b) A quantidade de caracteres de cada elemento do vetor;
   c) O endereço de memória do vetor;
   d) O endereço de memória de cada elemento do vetor
26) Com base no exercício anterior, apresente os itens **c** e **d** utilizando aritmética de ponteiro
27) É possível utilizar um ponteiro para fazer referência a uma variável de *struct*? Se sim, apresente um exemplo.
28) Defina o termo "ponteiro de ponteiro"
29) Considerando o programa abaixo, descreva o que será apresentado ao usuário após a execução completa do mesmo.

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
   int x = 10;
   int *ptr1;
   int **ptr2;
   ptr1 = &x;
   ptr2 = &ptr1;
   printf("%p\n", &x);
   printf("%p\n", &ptr1);
   printf("%p\n", &ptr2);
   printf("%d\n", x);
   printf("%p\n", *ptr1, &*ptr1);
   printf("%d %p\n", **ptr2, &*ptr2);
   system("pause");
   return 0;
}
```

30) Explique uma situação prática em que o uso de ponteiros em linguagem C é essencial ou traz vantagens em relação ao uso de variáveis comuns. Descreva o motivo da utilização de ponteiros nesse contexto.
31) Considerando que uma função retorna apenas um único resultado, é possível retornar mais de um valor em uma função através dos parâmetros? Explique.
32) Conclua o programa abaixo desenvolvido em linguagem C que apresente o quádruplo da variável **a**, utilizando o conceito de "ponteiro de ponteiro de ponteiro".

```c
#include <stdlib.h>
#include <stdio.h>
int main() {
   int a, *b, **c, ***d;
   printf("informe um valor ");
   scanf("%d",&a);
   //concluir aqui
   system("pause");
   return 0;
}
```

33) Implemente um programa em linguagem C que solicite uma *string* e passe como parâmetro para uma função denominada **inverte,** cujo objetivo é inverter a *string* e retornar o resultado.
34) Implemente um programa em linguagem C que leia uma *string* (s1) informada pelo usuário. Em seguida, desenvolva um procedimento denominado ***COPIA***, que receba duas *strings* como parâmetros. Esse procedimento deverá copiar o conteúdo de *s1* para *s2*, caractere por caractere, simulando o comportamento da função *strcpy*, porém sem utilizá-la.
35) O que significam as funções CALLOC, MALLOC e REALLOC?
36) Qual é o objetivo da função FREE() e como devemos (sintaxe) utilizá-lo?
37) Qual o objetivo de utilizarmos alocação dinâmica de memória? Explique.
38) Ao tentarmos alocar dinamicamente memória poderá não haver disponibilidade. Como resolver isso sem causar uma interrupção inesperada no programa?
39) Como funciona e quais parâmetros são necessários para uma alocação dinâmica de memória utilizando o comando CALLOC? Explique.
40) Como funciona e quais parâmetros são necessários para uma alocação dinâmica de memória utilizando o comando MALLOC? Explique.
41) Qual situação devemos utilizar a alocação CALLOC ou MALLOC? Explique
42) O que exatamente uma alocação dinâmica retorna, seja com CALLOC ou como MALLOC?
43) Quantos bytes de memória o retorno das funções CALLOC e MALLOC consumirão?
44) O bloco de memória alocado pela função MALLOC poderá conter lixo. Explique como isso pode acontecer.
45) Construa um programa em linguagem C que, através de alocação dinâmica de memória, solicite 3 valores do tipo *float*;
46) Construa um programa em linguagem C que, através de alocação dinâmica de memória, randomize 5 valores inteiros (entre 0 e 99). Posteriormente apresente os valores randomizados e armazenados na memória dinâmica. Obs. Não esqueça de liberar a memória posteriormente.
47) Construa um programa em linguagem C que aloque dinamicamente espaço para um vetor de 5 elementos do tipo inteiro (denominado vet1 alocado com MALLOC) e também espaço para armazenar 5 elementos do tipo inteiro (denominado vet2 alocado com CALLOC). Posteriormente apresente os valores de cada um dos vetores logo após as suas alocações.
48) Implemente um programa em linguagem C que solicite ao usuário a quantidade de elementos do tipo *float* que deseja criar dinamicamente. Em seguida, efetue a alocação através da função MALLOC e apresente a quantidade de bytes alocados caso o processo tenha sido realizado com êxito.
49) Implemente um programa em linguagem C que aloque dinamicamente espaço para armazenar 10 variáveis do tipo inteiro. Posteriormente, alimente os elementos com valores randômicos entre 20 e 50. Finalmente, redimensione o vetor para 15 elementos e apresente-os ao usuário (poderão ser apresentados lixos de memória nos últimos 5 elementos).
50) Implemente um programa em linguagem C que aloque dinamicamente espaço para armazenar 7 variáveis do tipo inteiro. Posteriormente, alimente os elementos com valores randômicos entre 10 e 30. Realoque o vetor para 5 elementos e apresente ao usuário, bem como seus respectivos endereços de memória.
51) Com base ao exercício anterior, apresente as informações ao usuário utilizando aritmética de ponteiro. Caso tenha feito anteriormente assim, utilize o modelo tradicional.
52) Ao realocar para mais um bloco de memória, o que pode acontecer com seus endereços caso não haja espaço contiguo no endereço atual do bloco?

53) Construa um programa em linguagem C que aloque dinamicamente um vetor com 10 elementos inteiros com valores randomizados a seu critério. Posteriormente, efetue uma realocação para 20 elementos e apresente ao usuário uma mensagem dizendo se houve ou não mudança de endereço do vetor após o *realloc*.

54) Implemente um programa em linguagem C que preencha de forma automática através de randomização, uma matriz de tamanho 5x4 com valores inteiros entre 100 e 200. Posteriormente, apresente a matriz ao usuário. Utilize alocação dinâmica.

55) Com base no exercício anterior, faça a apresentação ao usuário utilizando aritmética de ponteiro.

56) Implemente um programa em linguagem C que preencha um vetor com 5 elementos contendo nomes. A alocação dos nomes deverá ser feita de acordo com seu tamanho, ou seja, se o nome for “JOSE” a alocação deverá ser de tamanho 5 (4+1). Se o nome for “MARIANA”, a alocação naquela posição do vetor deverá ser de tamanho 8 (7+1). Apresente no final o endereço do vetor, o conteúdo de cada elemento do vetor, bem como seu respectivo endereço de memória.

57) Construa um programa em linguagem C que preencha uma matriz 3x4 de *strings* informada pelo usuário. O tamanho de cada *string* informada não poderá ultrapassar 10 caracteres. Apresente no final a matriz ao usuário.

58) Analise o código em linguagem C apresentado abaixo, descrevendo detalhadamente sua funcionalidade. Caso identifique algum erro, indique qual é o problema e apresente a devida correção.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    int **mat, i, j;
    srand(time(NULL));
    mat = malloc(3 * sizeof(int *));
    for (i = 0; i < 3; i++)
        mat[i] = malloc(4 * sizeof(int));
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            mat[i][j] = rand() % 100;
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 4; j++)
            printf("%4d ", mat[i][j]);
        printf("\n");
    }
    free(mat);
    getch();
    return 0;
}
```

59) Considerando uma matriz alocada dinamicamente de tamanho 6x4, composta por *strings* de até 51 caracteres, e representada por ***mat**, descreva as etapas necessárias para a correta liberação da memória ocupada por essa estrutura.

60) Explique por que, em linguagem C, o tamanho de um ponteiro não depende do tipo de dado que ele aponta, mas sim da arquitetura do sistema. Justifique sua resposta.

61) Analise o código em linguagem C apresentado a seguir e descreva, de forma detalhada, qual é a sua funcionalidade. Caso identifique algum erro no código, seja sintático ou lógico, aponte-o e explique qual seria a correção adequada a ser realizada.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int i,n;
char continua;

void entrada(int *dados, int n){
  for (i=0; i<n; i++){
    printf("Numero %d: ",i);
    scanf("%d",&dados[i]);}
}

void imprime(int *dados, int n){
  printf("numeros lidos: ");
  for (i=0; i<n; i++)
    printf("%d ",dados[i]);
  printf("\n");
}

int main() {
  int *numeros;
  do{
    system("cls");
    printf("quantos elementos? ");
    scanf("%d",&n);
    numeros=malloc(n*sizeof(int));
    entrada(numeros,n);
    imprime(numeros,n);
    printf("deseja continuar? (S/N)");
    continua=getch();
  }
  while (continua!='N'||continua!='n');
  free(numeros);
  getch();
  return 0;
}
```

62) Analisando o código (parcial) em linguagem C abaixo, descreva detalhadamente o que ele faz

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 50

void entrada(char **dados, int qt){
char tmp[SIZE];
int i;
  for (i=0; i<qt; i++){
    printf("Nome %d: ",i);
    gets(tmp);
    dados[i]=malloc((strlen(tmp)+1)*sizeof(char));
    strcpy(dados[i],tmp);
  }
}
```

63) Implemente um programa em linguagem C que permita o cadastro de N produtos, onde N deve ser informado pelo usuário. Para cada produto, devem ser armazenadas as seguintes informações:

- Código (int)
- Nome (string de tamanho variável)
- Preço (float)

Regras:

a) O cadastro dos produtos deve ser feito utilizando struct e alocação dinâmica.
b) O nome do produto deve ser armazenado de forma dinâmica, utilizando o tamanho exato informado pelo usuário.
c) Após o cadastro, o programa deve exibir todos os produtos cadastrados.
d) Não se esqueça de liberar toda memória alocada.

64) Crie um programa em linguagem C que permita cadastrar informações de alunos. Para cada aluno devem ser armazenados:

- Nome (*string* de tamanho variável)
- Quantidade de notas (*int*)
- Notas (vetor de *floats*, tamanho informado pelo usuário)

Regras:

a) O número de alunos deve ser informado pelo usuário.
b) Para cada aluno, o usuário deve informar quantas notas deseja armazenar.
c) O programa deve calcular e exibir a média de cada aluno.
d) Utilize alocação dinâmica para o cadastro dos alunos, para o nome de cada aluno e para o vetor de notas.
e) Libere toda memória alocada ao final do programa

65) Com relação ao exercício 64, apresente ao usuário também, a quantidade de memória consumida no *stack* e no *heap* após a alocação das variáveis.

66) Desenvolva um programa em linguagem C que realize o cadastro de pessoas, com as seguintes funcionalidades: inclusão, alteração, exclusão e consulta de registros.

As informações de cada pessoa devem ser armazenadas em uma estrutura (struct) contendo os seguintes campos:

- Nome
- Endereço
- Cidade
- Sexo
- CPF
- RG
- E-mail

Todos os campos devem ser alocados dinamicamente (uso de malloc, realloc, etc.) de acordo com a necessidade de armazenamento de *strings* e structs.

As estruturas deverão ser organizadas em um vetor dinâmico de ponteiros, permitindo a manipulação flexível da quantidade de registros conforme o programa é executado.

Implemente um menu interativo que permita ao usuário escolher entre as operações de cadastro, edição, exclusão e consulta de registros. Garanta a liberação adequada da memória ao final do programa.

67) Implemente um programa em linguagem C que utilize alocação dinâmica de memória para criar e preencher uma matriz de inteiros com dimensões 3x5, armazenando valores sequenciais de 0 a 15.

Após isso, o programa deve realocar dinamicamente a matriz para as novas dimensões 5x5, preservando os dados já inseridos e preenchendo os novos elementos com os valores sequenciais de 15 a 24.

O programa deve exibir a matriz ao usuário duas vezes:

1. Após o preenchimento da matriz 3x5.
2. Após a realocação e o preenchimento dos novos elementos na matriz 5x5.

Obs. Como a apresentação da matriz será efetuada duas vezes, utilize um procedimento.

Entrega:
https://forms.gle/S4ippSN4xaL2kJcQ9

Anexar um único PDF com as respostas