**Atividades do conteúdo: Unidade 1 (Conceitos Básicos)**

**1 - 1** Declare e inicialize uma instância para cada um dos seguintes tipos de dados:

1. Smartphone, com o nome do modelo, memória e dimensões de tela.
2. Jogo de futebol, com os nomes dos times e o placar (faça o placar ser também um tipo de dado);

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  typedef struct smartphone{  char nomeModelo[30];  float memoria;  float largura;  float altura;  }TPSmartphone;  typedef struct placar{  int placarTimeLadoEsquerdo;  int placarTimeLadoDireito;  }TPPlacar;  typedef struct jogoDeFutebol{  char timeLadoDireito[30];  char timeLadoEsquerdo[30];  TPPlacar \*placar;  }TPJogoDeFutebol;  int main(){  TPSmartphone \*ponteiroSmartphone;  ponteiroSmartphone = (TPSmartphone \*) malloc(sizeof(TPSmartphone));  TPJogoDeFutebol \*ponteiroJogo;  ponteiroJogo = (TPJogoDeFutebol \*) malloc(sizeof(TPJogoDeFutebol));  ponteiroJogo->placar = (TPPlacar \*) malloc(sizeof(TPPlacar));  printf("Digite o modelo: ");  gets(ponteiroSmartphone->nomeModelo);  printf("Digite a capacidade de memoria: ");  scanf("%f", &ponteiroSmartphone->memoria);  printf("Digite a largura: ");  scanf("%f", &ponteiroSmartphone->largura);  printf("Digite a altura: ");  scanf("%f", &ponteiroSmartphone->altura);  fflush(stdin);  printf("Time A: ");  gets(ponteiroJogo->timeLadoDireito);  printf("Time B: ");  gets(ponteiroJogo->timeLadoEsquerdo);  printf("Placar Time A: ");  scanf("%i", &ponteiroJogo->placar->placarTimeLadoDireito);  printf("Placar Time B: ");  scanf("%i", &ponteiroJogo->placar->placarTimeLadoEsquerdo);  } |

**1 - 2** Implemente o TAD TRegistroCorrida, conforme descrito na atividade 1-2.

|  |
| --- |
| #ifndef REGISTROCORRIDA\_H\_INCLUDED  #define REGISTROCORRIDA\_H\_INCLUDED  typedef struct RegistrosCorrida TRegistroCorrida;  TRegistroCorrida \*criar(int quantidade);  float calcularMedia(TRegistroCorrida \*registroCorrida, int quantidadeParaCalculo);  float maiorFrequencia(TRegistroCorrida \*registroCorrida, int quantidadeParaVetor);  void findByType(TRegistroCorrida \*registroCorrida, int quantidadeParaVetor, int defineRetorno);  #endif // REGISTROCORRIDA\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #includ "registroCorrida.h"  struct RegistrosCorrida {  float frequencia;  float temperatura;  float elevacao;  };  TRegistroCorrida \*criar(int quantidade){  TRegistroCorrida \*registroCorrida = (TRegistroCorrida \*) malloc(quantidade\*sizeof(TRegistroCorrida));  return registroCorrida;  };  float calcularMedia(TRegistroCorrida \*registroCorrida, int quantidadeParaCalculo){  int i; float soma = 0;  for(i = 0; i < quantidadeParaCalculo; i++){  soma = soma + registroCorrida[i].frequencia;  }  return soma / quantidadeParaCalculo;  }  float maiorFrequencia(TRegistroCorrida \*registroCorrida, int quantidadeParaVetor){  int i; float maiorValor = 0;  for(i = 0; i < quantidadeParaVetor; i++){  if(registroCorrida[i].frequencia > maiorValor){  maiorValor = registroCorrida[i].frequencia;  }  }  return maiorValor;  }  void findByType(TRegistroCorrida \*registroCorrida, int quantidadeParaVetor, int defineRetorno){  int i;  switch(defineRetorno){  case 1:  printf("Lista de FREQUENCIAS");  for(i = 0; i < quantidadeParaVetor; i++){  printf("\nFrequencia: %f", registroCorrida[i].frequencia);  }break;  case 2:  printf("Lista de TEMPERATURAS");  for(i = 0; i < quantidadeParaVetor; i++){  printf("\nTemperatura: %f", registroCorrida[i].temperatura);  }break;  case 3:  printf("Lista de ELECACOES");  for(i = 0; i < quantidadeParaVetor; i++){  printf("\nElevacao: %f", registroCorrida[i].elevacao);  }break;  }  } |

**1 - 3** Implemente o TAD Jogo de Futebol com as seguintes funcionalidades:

1. Criar um jogo, que recebe dos dados de um jogo, inicializa uma variável Jogo de Futebol com esses dados, e retorna o ponteiro para ela;
2. Vencedor, que recebe um jogo e mostra na tela o nome do time vencedor;
3. Goleada, que recebe um jogo e retorna 1, se houve uma goleada e 0 em caso contrário. Uma goleada é um placar com diferença maior ou igual a 3.

|  |
| --- |
|  |