Lista de exercício Estruturas

1.) Escreva um programa fazendo o uso de estruturas. Você deverá criar uma estrutura chamada Ponto, contendo apenas a posição x e y (inteiros) do ponto. Declare 2 pontos, leia a posição (coordenadas x e y) de cada um e calcule a distância entre eles. Apresente no final a distância entre os dois pontos.

```
typedef struct _PONTO {
f loat x , y;
} PONTO;
typedef struct _CIRCUNFERENCIA {
f loat raio ;
PONTO centro ;
} CIRCUNFERENCIA ;
```

```
#include < stdio .h>
typedef struct _PONTO {
f loat x , y;
} PONTO;
int main (void) {
PONTO p;
printf (" Entre com as coordenadas do ponto p\n");
scanf ("%f %f", &p.x, &p.y);
printf (" Dados lidos\n");
printf (" Ponto p: x = %f , y = %f\n" , p.x , p.y);
return 0;
}
```

==========

```
#include < stdio .h>
typedef struct _PONTO {
f loat x , y;
} PONTO;
typedef struct _CIRCUNFERENCIA {
f loat raio ;
PONTO centro ;
} CIRCUNFERENCIA ;

int main (void) {
CIRCUNFERENCIA c1 ;
printf (" Entre com o raio do circulo c1 \n");
scanf ("%f", & c1 . raio );
printf (" Entre com as coordenadas do centro do circulo c1 \n");
scanf ("%f %f", &c1. centro .x, &c1 . centro .y);
printf (" Dados lidos\n");
printf (" Circulo c1: raio = %f, x = %f , y = %f\n" , c1. raio , c1 .centro .x ,c1 .centro .y);
return 0;
}
```

2.) Fazer um programa para simular uma agenda de telefones. Para cada pessoa devem-se

ter os seguintes dados:

• Nome

• E-mail

• Endereço (contendo campos para Rua, numero, complemento, bairro, cep, cidade, estado, país).

• Telefone (contendo campo para DDD e numero)

• Data de aniversario (contendo campo para dia, m es, ano).

• Observações: Uma linha (string) para alguma observação especial.

(a) Definir a estrutura acima.

(b) Declarar a variavel agenda (vetor) com capacidade de agendar até 100 nomes.

(c) Definir um bloco de instruções busca por primeiro nome: Imprime os dados da pessoa com esse nome (se tiver mais de uma pessoa, imprime para todas).

(d) Definir um bloco de instruções busca por m es de aniversário: Imprime os dados de todas as pessoas que fazem aniversario nesse m ´ es. ˆ

(e) Definir um bloco de instruções busca por dia e mes de aniversário: Imprime os dados de todas as pessoas que fazem aniversario nesse dia e mes.

(f) Definir um bloco de instruções insere pessoa: Insere por ordem alfabética de nome.

(g) Definir um bloco de instruções retira pessoa: retira todos os dados dessa pessoa e desloca todos os elementos seguintes do vetor para a posição anterior.

(h) Definir um bloco de instruções imprime agenda com as opções:

• imprime nome, telefone e e-mail.

• imprime todos os dados.

(i) O programa deve ter um menu principal oferecendo as opções acima.

3.) Construa uma estrutura aluno com nome, numero de matrícula e curso. Leia do usuario a informação de 5 alunos, armazene em vetor dessa estrutura e imprima os dados na tela.

4.) Crie uma estrutura representando os alunos do curso de Introdução a Programação de Computadores. A estrutura deve conter a matrícula do aluno, nome, nota da primeira prova, nota da segunda prova e nota da terceira prova.

(a) Permita ao usuario entrar com os dados de 5 alunos. ´

(b) Encontre o aluno com maior nota da primeira prova.

(c) Encontre o aluno com maior media geral.

(d) Encontre o aluno com menor media geral

(e) Para cada aluno diga se ele foi aprovado ou reprovado, considerando o valor 6 para aprovação.

4.) Escreva um trecho de codigo em ”C” para fazer a criação dos novos tipos de dados conforme solicitado abaixo:

• Horario: composto de hora, minutos e segundos. ´

• Data: composto de dia, mes e ano. ˆ

• Compromisso: composto de uma data, horario e texto que descreve o compromisso.

• Byte: usado para armazenar 8 bits (definido em função dos tipos básicos do ”C”) ´

5.) Faça um programa que armazene em um registro de dados (estrutura composta) os dados de um funcionário de uma empresa, compostos de: Nome, Idade, Sexo (M/F), CPF, Data de Nascimento, Codigo do Setor onde trabalha (0-99), Cargo que ocupa (string de ate 30 caracteres) e Sal ´ ario. Os dados devem ser digitados pelo usuário, armazenados na estrutura e exibidos na tela.

6.) Faça um programa que leia os dados de 10 alunos (Nome, matricula, Media Final), armazenando em um vetor. Uma vez lidos os dados, divida estes dados em 2 novos vetores, o vetor dos aprovados e o vetor dos reprovados, considerando a media mínima para a aprovação como sendo 5.0. Exibir na tela os dados do vetor de aprovados, seguido dos dados do vetor de reprovados.

7.) Faça um programa com N=5 e:

• Crie e leia um vetor de alunos, sendo que cada aluno contem os dados: nome (maximo 15 letras), notas de 3 provas, média final e nível (inteiro). Este ultimo campo nao deve ser lido agora.

• Preencha o campo nível. Seu valor deve ser igual a parte inteira de (5*média final/media da sala). Na sua função main(), mostre o nome e o nível de cada aluno.

8.) Faça um programa que seja uma agenda de compromissos e:

• Crie e leia um vetor de 5 estruturas de dados com: compromisso (maximo 60 letras) e data. A data deve ser outra estrutura de dados contendo dia, mes e ano. ˆ

• Leia dois inteiros m e a e mostre todos os compromissos do mes m do ano a. Repita o procedimento ate ler m = 0. ´

Dica: use fgets(string, tamanho, stdin) para ler uma string, precedido imediatamente por fflush(stdin).

9.) Faça um programa que gerencie o estoque de um mercado e:

• Crie e leia um vetor de 5 produtos, com os dados: codigo (inteiro), nome (máximo 15 letras), preço e quantidade.

• Leia um pedido, composto por um codigo de produto e a quantidade. Localize este codigo no vetor e, se houver quantidade suficiente para atender ao pedido integralmente, atualize o estoque e informe o usuario. Repita este processo até ler um codigo igual a zero.

10.) Faça um programa que controle o fluxo de voos nos aeroportos de um país. Com v=5(voos) e a=5 (aeroportos) e:

• Crie e leia um vetor de voos, sendo que cada voo contem um código de aeroporto de origem e um de destino.

• Crie um vetor de aeroportos, sendo que cada aeroporto contem seu codigo, quantidade de voos que saem e quantidade de voos que chegam.

Nota: Cada aeroporto e identificado por um codigo inteiro entre 0 e (a-1). Não aceite aeroportos de codigo inexistente.

11.) Crie uma estrutura chamada Retângulo. Essa estrutura deverá conter o ponto superior esquerdo e o ponto inferior direito do retângulo. Cada ponto é definido por uma estrutura Ponto, criada no exercício anterior. Faça um programa que declare e leia uma estrutura Retângulo e exiba a área, o comprimento da diagonal e o perímetro desse retângulo.

12.) Usando as estruturas Retângulo e Ponto dos exercícios anteriores, faça um programa que declare e leia uma estrutura Retângulo e um Ponto e informe se esse ponto está ou não dentro do retângulo.

13.) Crie uma estrutura Atleta representando um atleta. Essa estrutura deve conter o nome do atleta, seu esporte, idade e altura. Agora, escreva um programa que leia os dados de cinco atletas. Calcule e exiba os nomes do atleta mais alto e do mais velho.

14.) Usando a estrutura Atleta do exercício anterior, escreva um programa que leia os dados de cinco atletas e os exiba por ordem de idade, do mais velho para o mais novo

15.) Considere a seguinte estrutura:

typedef struct _TEMPO {

int hora , minuto , segundo ;

} TEMPO;

Escreva um programa que leia dois tempos (TEMPO t1, t2;) gastos em uma tarefa qualquer. O programa deve imprimir o maior tempo.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
typedef struct _TEMPO {
   int h, m, s;
} TEMPO;
```

```
void leTempo(TEMPO *);
void imprimeTempo(TEMPO );
int comparaTempo( const void *, const void *);

int main(int argc, char **argv)
{
   TEMPO t[MAX];
   int i;

   for (i = 0; i < MAX; i++) {
      printf("%d? ", i);
      leTempo(&t[i]);
   }
```

//qsort: Essa função recebe o vetor de elementos a serem ordenados, o número de elementos desse vetor, o tamanho de cada elemento e uma função que é o critério de ordenação

```
   qsort(t, MAX, sizeof(TEMPO), comparaTempo);

   for (i = 0; i < MAX; i++) {
      imprimeTempo(t[i]);
   }
   return 0;
}
int comparaTempo( const void *a, const void *b) {
   if ( ((TEMPO *)a)->h > ((TEMPO *)b)->h) return 1;
   else if ( ((TEMPO *)a)->h == ((TEMPO *)b)->h) return 0;
   else return -1;
}
void leTempo(TEMPO *x) {
   scanf("%d %d %d", &(x->h), &(x->m), &(x->s));
}
void imprimeTempo(TEMPO x) {
   printf("%d:%d:%d\n", x.h, x.m, x.s);
}
```

16.)Analise com detalhes o que este programa em C faz.

```
#include < stdio .h>
#include < stdlib .h>
typedef struct _fun
{
char nome [40];
f loat salario ;
} Tfunc;
void le ( Tfunc * cadastro , int funcionarios ) {
int i;
char linha [40];
for (i =0; i< funcionarios ; i ++) {
puts (" Nome ?");
fgets ((cadastro +i) ->nome , 39, stdin );
puts (" Salario ?"); fgets ( linha , 39, stdin );
scanf ( linha , "%f", &(( cadastro +i) -> salario ));
}
}
f loat media ( Tfunc * cadastro , int funcionarios ) {
f loat media =0.0;
int i;
for (i =0; i< funcionarios ; i ++) {
media += ( cadastro +i) -> salario ;
}
return media /= funcionarios ;
}
int main (void) {
Tfunc* cadastro ;
int funcionarios ;
char linha [40];
puts (" Quantos funcionarios ?"); fgets( linha , 39, stdin );
scanf (linha , "%d" , & funcionarios );
i f (!( cadastro = ( Tfunc*) malloc( funcionarios * s i zeof ( Tfunc )))) {
exit (1) ;
}
le(cadastro , funcionarios );
printf (" Salario medio = %.2 f\n",
media (cadastro , funcionarios ));
return 0;
}
```

17.) Considere que uma empresa precisa armazenar os seguintes dados de um cliente:

• Nome completo com no máximo 50 caracteres;
• renda mensal do cliente;
• ano de nascimento;
• possui ou não carro.

Defina um tipo e uma estrutura para armazenarem estes dados e escreva um programa que leia estes dados armazene-os em uma variável e em seguida os imprima.

18.) Considerando a mesma estrutura do exercício anterior, escreva um programa que leia os dados de 10 clientes e imprima:

- quantos clientes têm renda mensal acima da média;
- quantos clientes têm carro;
- quantos clientes nasceram entre 1960 (inclusive) e 1980 (exclusive).

19.) Crie uma estrutura chamada retângulo, que possua duas estruturas ponto (o ponto superior esquerdo e o ponto inferior direito). Faça um programa que receba as informações acerca de um retângulo (as coordenadas dos dois pontos), e informe a área, o comprimento da diagonal e o comprimento de cada aresta.

20.) Escreva um programa que use as mesmas estruturas do exercício anterior para descobrir se um ponto está dentro de um retângulo.

21.) Considere a seguinte estrutura:

```
typedef  struc t _ J O G A D O R {

in t p o n t o s ;

char nome [ 4 2 ] ;

} J O G A D O R ;
```

Escreva um programa que crie um vetor com os dados de 5 jogadores; leia estes do teclado e imprima na ordem em que foram lidos.

22.) Dada a seguinte estrutura:

typedef struct _FRACAO {
int numerador , denominador ;
} FRACAO ;

Escreva um programa que leia duas frações e calcule e imprima sua:
1. soma;
2. subtração;
3. produto;
4. divisão
Caso uma fração com denominador igual a zero seja lido o programa deve emitir um aviso e parar

23.) Escreva um programa que leia os dados de duas circunferências e verifique se elas estão uma interna a outra.

24.) Manipulação de bits. Identifique o que será impresso pelo programa abaixo.

```
2     Usando os operadores de deslocamento sobre bits */
3  #include <stdio.h>
4
5  void displayBits( unsigned value ); /* protótipo */
6
7  int main( void )
8  {
9     unsigned number1 = 960; /* inicializa number1 */
10
11    /* demonstra deslocamento à esquerda sobre bits */
12    printf( "\nO resultado do deslocamento à esquerda de\n" );
13    displayBits( number1 );
14    printf( "por 8 posições de bit usando o " );
15    printf( "operador de deslocamento à esquerda << é\n" );
16    displayBits( number1 << 8 );
17
18    /* demonstra deslocamento à direita sobre bits */
19    printf( "\nO resultado do deslocamento à direita de\n" );
20    displayBits( number1 );
21    printf( "por 8 posições de bit usando o " );
22    printf( "operador de deslocamento à direita >> é\n" );
23    displayBits( number1 >> 8 );
24    return 0; /* indica conclusão bem-sucedida */
25 } /* fim do main */
26
27 /* mostra bits de um valor inteiro sem sinal */
28 void displayBits( unsigned value )
29 {
30    unsigned c; /* contador */
```

```
32     /* declara displayMask e desloca 31 bits à esquerda */
33     unsigned displayMask = 1 << 31;
34
35     printf( "%7u = ", value );
36
37     /* loop pelos bits */
38     for ( c = 1; c <= 32; c++ ) {
39        putchar( value & displayMask ? '1' : '0' );
40        value <<= 1; /* desloca valor 1 bit à esquerda */
41
42        if ( c % 8 == 0 ) { /* mostra um espaço após 8 bits */
43           putchar( ' ' );
44        } /* fim do if */
45     } /* fim do for */
46
47     putchar( '\n' );
48  } /* fim da função displayBits */
```

```
O resultado do deslocamento à esquerda
    960 = 00000000 00000000 00000011 11000000
por 8 posições de bit usando o operador de deslocamento à esquerda << é
 245760 = 00000000 00000011 11000000 00000000

O resultado do deslocamento à direita
    960 = 00000000 00000000 00000011 11000000
por 8 posições de bit usando o operador de deslocamento à direita >> é
      3 = 00000000 00000000 00000000 00000011
```

25.) Neste exercício, você deve criar um protótipo de um sistema de batalha entre guerreiros de um jogo. Para isso, implemente os itens a seguir em um módulo separado chamado jogo.

1 Defina um novo tipo de dados chamado Guerreiro com os seguintes campos: ataque (inteiro), defesa (inteiro), pontos_vida (inteiro) e id_jogador (inteiro).

2 Escreva uma função de nome rolaDados que simula a rolagem de três dados de seis faces tradicionais (1 a 6) e retorna a soma dessas rolagens. Note que somar os valores resultantes da rolagem de três dados de seis faces é diferente de rolar um dado que retorna um número entre 3 e 18.

3 Escreva um procedimento de nome criaGuerreiro que recebe um Guerreiro por passagem de parâmetro por referência e que atribui valores aos seus campos de batalha. Os seus campos de batalha (ataque e defesa devem receber um valor inteiro da função rolaDados. O campo pontos_vida deve receber a soma dos valores retornados por três execuções da função rolaDados.

4 Escreva um procedimento de nome ataca que recebe dois Guerreiros por passagem de parâmetro por referência e simula um ataque do primeiro guerreiro no segundo. O ataque é dado da seguinte maneira:

a.) O primeiro guerreiro rola três dados e soma os seus valores com o seu campo ataque. Essa soma é o valor do golpe do primeiro guerreiro.

b.) O segundo guerreiro rola três dados e soma os seus valores com o seu campo defesa. Essa soma é o valor do escudo do segundo guerreiro.

c.) Faça dano = golpe − escudo. Se o dano for maior que zero, reduza dano dos pontos_vida do segundo guerreiro.

Geração de números randômicos em C:

Para gerar um número aleatório (randômico) em linguagem C podemos usar a **função rand** pertencente à biblioteca stdlib.h. Gerar sequências de números aleatórios é um problema bastante comum em programação. Quando esta função é chamada ela produz um valor aleatório na faixa entre 0 e a constante RAND_MAX. O valor desta constante encontra-se definida no arquivo stdlib.h.

Muitas vezes necessitamos gerar valores dentro de determinada faixa. Para exemplificar, vamos supor que a faixa de valores desejada esteja entre o valor mínimo zero e o valor máximo 100. Veja um exemplo prático que gera uma sequência com 10 números aleatórios.

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

int main(void)
{
   int i;

   printf("Gerando 10 valores aleatorios:\n\n");
    for (i = 0; i < 10; i++)     {
       /* gerando valores aleatórios entre zero e 100 */
       printf("%d ", rand() % 100);
   }
    getch();
   return 0; }
```