

## Lista 2 – Lab. De Estrutura de Dados 1 e Estrutura de Dados 1

**Aluno:** Anderson Carlos da Silva Moraes

**Matrícula:** 2024011327

### Questões

**1º** Defina um registro que descreve um peixe. O registro deve incluir o tipo (string), o peso (ponto-flutuante) e o comprimento (inteiro) do peixe. Em seguida mostre:

- a) Como criar uma variável de tipo peixe
- b) Como criar um ponteiro para uma variável de tipo peixe.

**2º** Construa uma função que receba um peixe e exiba o seu conteúdo.

- a) Faça uma versão utilizando um parâmetro tipo peixe
- b) Faça uma versão utilizando um parâmetro tipo ponteiro para peixe

**Resposta 1 e 2:**

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

struct peixe
{
    char tipo[20];    // String -> array de char
    float peso;       // ponto flutuante
    int comprimento; // comprimento tipo inteiro
};

void exibirPeixe(struct peixe p);
void exibirPeixeP(struct peixe *ptr);

int main()
{
    setlocale(LC_ALL, "Portuguese");

    struct peixe peixe1 = {"Baiacu", 0.100, 15}; // criando uma variável

    struct peixe *ptr = &peixe1; // Criando um ponteiro chamado ptr do
    tipo struct peixe e passando o endereço da variável peixe1

    exibirPeixe(peixe1);
    exibirPeixeP(ptr);
    return 0;
}
```

```

}

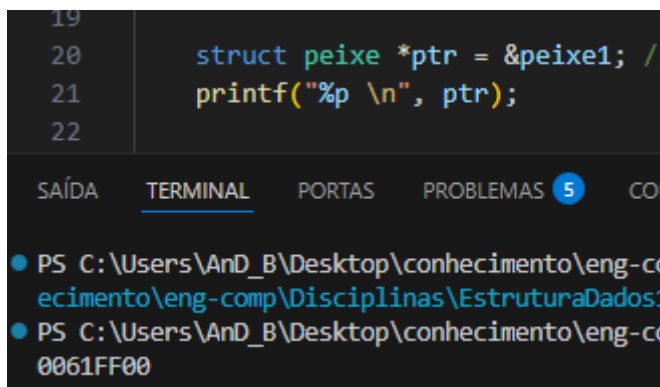
void exibirPeixe(struct peixe p)
{
    printf("Tipo: %s\n", p.tipo);
    printf("Peso: %f\n", p.peso);
    printf("Comprimento: %d\n", p.comprimento);
}

void exibirPeixeP(struct peixe *ptr)
{
    printf("\n");
    printf("Tipo: %s\n", ptr->tipo);
    printf("Peso: %f\n", ptr->peso);
    printf("Comprimento: %d\n", ptr->comprimento);
}

```

3º Descubra o que acontece ao tentarmos acessar um ponteiro que contém um endereço inválido. Para isso tente mostrar o conteúdo apontado por um ponteiro recém-criado:

Ao tentar acessar um ponteiro recém-criado, ele não irá apontar para nada em específico, exibindo um endereço de lixo de memória.



The screenshot shows a code editor with the following C code:

```

19
20     struct peixe *ptr = &peixe1; //
21     printf("%p \n", ptr);
22

```

Below the code, the terminal output is shown:

```

SAÍDA  TERMINAL  PORTAS  PROBLEMAS 5  CON
• PS C:\Users\AnD_B\Desktop\conhecimento\eng-co
ecimento\eng-comp\Disciplinas\EstruturaDados1
• PS C:\Users\AnD_B\Desktop\conhecimento\eng-co
0061FF00

```

4º Construa duas funções que realizem o incremento de um número em uma unidade. A função Mais deve receber um número inteiro através de um ponteiro. A função Incrementa deve receber um valor inteiro, sem usar ponteiros, e retornar o valor incrementado em uma unidade. Utilize as duas funções como no exemplo abaixo:

**Dica:** observe que a função Mais pode modificar diretamente o valor da variável recebida, enquanto a função Incrementa precisa retornar o valor porque ela tem acesso apenas a uma cópia da variável.

```

#include <stdio.h>
#include <locale.h>

void Mais(int *ptr)
{
    (*ptr)++;
}

int Incrementa(int num)
{
    return num + 1;
}

int main()
{
    setlocale(LC_ALL, "Portuguese"); // Configura o locale para português
    no Windows

    int num = 0;

    printf("Informe um número:\n");
    scanf("%d", &num);

    printf("numero: %d\n", num);
    Mais(&num);

    printf("Resultado de Mais: %d\n", num);
    printf("Resultado de Incrementa: %d\n", Incrementa(num));
    printf("numero: %d\n", num);

    return 0;
}

```

5º Modifique o programa abaixo para que ele mostre o número 10 na tela usando o ponteiro q:

```

#include <stdio.h> int main()

{

    int x, *p, **q; p = &x;

    q = &p; x = 10;

    printf("%p\n", (void*)q);
}

```

```
return 0;
```

```
}
```

```
#include <stdio.h>
int main()
{
    int x, *p, **q; // **q é um ponteiro para um ponteiro de int
    p = &x;         // p recebe o endereço de x
    q = &p;         // q recebe o endereço do ponteiro p. Ou seja, q
                    // aponta para p, que aponta para x.
    x = 10;
    printf("%d\n", **q);
    return 0;
}
```

6 ° Crie uma variável do tipo char inicializada para o caractere 'A' e um ponteiro que aponte para esta variável. Modifique a variável criada usando o ponteiro, de forma que seu conteúdo agora seja 'B'. Por fim, mostre o conteúdo da variável e o conteúdo apontado pelo ponteiro.

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

int main()
{
    setlocale(LC_ALL, "Portuguese");

    char ltr = 'A';
    char *ptr = &ltr;

    *ptr = 'B';

    printf("%c\n", ltr);
    printf("%c\n", *ptr);

    return 0;
}
```

7º Uma cor pode ser representada pela combinação de 4 valores de intensidade para R (Red), G (Green), B (Blue) e A (Alpha). Esses valores podem ser guardados em um registro com 4 inteiros de 8 bits (0-255) ou por um valor inteiro de 32 bits codificado com os 4 valores. Construa uma união para armazenar uma cor. Em seguida construa uma

função para ler do usuário uma cor no formato RGBA e outra para ler uma cor no formato inteiro de 32 bits. Ambas as funções devem receber o endereço de uma variável do tipo cor e modificar a variável recebida, sem retornar valor.

```
#include <stdio.h>
#include <stdint.h>
#include <locale.h>

struct code_rgba
{
    uint8_t r, g, b, a;
};

union COR
{
    struct code_rgba rgba;
    uint32_t cod;
};

void cor_rgba(union COR *ptr)
{
    int r, g, b, a;
    printf("Informe os valores para: RGBA\n");
    scanf("%d %d %d %d", &r, &g, &b, &a);

    ptr->rgba.r = (uint8_t)r;
    ptr->rgba.g = (uint8_t)g;
    ptr->rgba.b = (uint8_t)b;
    ptr->rgba.a = (uint8_t)a;
}

void cor_cod(union COR *ptr)
{
    int codigo;
    printf("Informe o valor inteiro de 32 bits (RGBA)\n");
    scanf("%u", &codigo);

    ptr->cod = codigo;
}

void imprimir_cor(union COR *ptr)
{
    printf("Cor (R,G,B,A): (%u, %u, %u, %u)\n",
           ptr->rgba.r, ptr->rgba.g, ptr->rgba.b, ptr->rgba.a);
    printf("Valor inteiro: %u (0x%08X)\n", ptr->cod, ptr->cod);
}

int main()
```

```

{
    setlocale(LC_ALL, "Portuguese");

    union COR cor, *px;
    px = &cor;

    printf("Ler cor no formato RGBA\n");
    cor_rgba(px);
    imprimir_cor(px);

    printf("Ler cor no formato inteiro\n");
    cor_cod(px);
    imprimir_cor(px);

    return 0;
}

```

**8º** Declare um registro "Tigela" com os campos estado (cheia ou vazia) e tipo de alimento (sopa ou canja). Crie uma função "Fome" que recebe um ponteiro para uma Tigela e altera o seu estado para "vazia". Na função principal crie uma tigela cheia, crie um ponteiro que aponta para essa tigela e então mostre como a tigela estava antes da janta. Depois chame a função Fome com o ponteiro que aponta para a tigela e ao fim mostre a tigela depois da janta.

```

#include <stdio.h>
#include <locale.h>

enum estado
{
    CHEIO,
    VAZIA
};
enum tipo_food
{
    sopa,
    canja
};

struct Tigela
{
    enum estado stage;
    enum tipo_food comida;
};

```

```

void Fome(struct Tigela *ptr)
{
    ptr->stage = VAZIA;
}

int main()
{
    setlocale(LC_ALL, "Portuguese");

    struct Tigela t1 = {CHEIO, sopa};
    struct Tigela *ptr = &t1;

    const char *estados[] = {"cheia", "vazia"}; // ponteiro de vetor de
array de caracteres
    const char *comidas[] = {"sopa", "canja"}; // ponteiro de vetor
array de caracteres

    printf("Antes do RU: %s\n%s\n", estados[ptr->stage], comidas[ptr-
>comida]);
    // acessando o elemento estado[CHEIO], o enum possui um índice
integrado, logo CHEIO -> 0, VAZIO -> 1, estado -> n-1
    // acessando o elemento comida[sopa], o enum possui um índice
integrado, logo sopa -> 0, canja -> 1, comida_n -> n-1

    Fome(ptr);

    printf("Pos RU: %s\n%s\n", estados[ptr->stage], comidas[ptr-
>comida]);

    return 0;
}

```

9º Declare um registro Horário com os campos horas e minutos. Crie uma função MostrarHorario que deve receber um ponteiro para um Horário e mostrá-lo no formato HH:MM. Na função principal, declare uma variável do tipo Horário e um ponteiro que aponta para ela. Peça que o usuário digite o horário atual e guarde-o na variável. Usando o ponteiro, incremente o horário recebido em uma hora e em seguida mostre o horário corrigido com MostrarHorario.

```

#include <stdio.h>
#include <locale.h>
struct Horario
{
    int h;

```

```

    int min;
};

void MostrarHorario(struct Horario *ptr)
{
    printf("%02d:%02d\n", ptr->h, ptr->min);
}

// Função para normalizar o horário: converte minutos extras em horas
void NormalizarHorario(struct Horario *ptr)
{
    if (ptr->min >= 60)
    {
        ptr->h += ptr->min / 60;
        ptr->min = ptr->min % 60;
    }
    if (ptr->h >= 24)
    {
        ptr->h = ptr->h % 24;
    }
}

int main()
{
    setlocale(LC_ALL, "Portuguese");

    struct Horario h1, *px;
    px = &h1;

    printf("Informe o horário atual [hh:mm]\n");
    scanf("%d %d", &px->h, &px->min);

    px->h++;

    NormalizarHorario(px);
    MostrarHorario(px);
    return 0;
}

```

**10º** Descubra qual é a saída do seguinte trecho de código, sem auxílio do computador. Depois rode o programa passo a passo com o depurador para verificar se conseguiu chegar na resposta certa.

```

#include <stdio.h> int main() {

    int valor = 10, *temp, soma = 0;

```



```
temp = &valor; *temp = 20;

temp = &soma; *temp = valor;

printf("valor: %d\nsoma: %d\n", valor, soma);

return 0;

}
```

**Sugestão:** observe como as variáveis se alteram com a execução do programa.

A saída será o valor 20 e 20. O código consiste em usar o ponteiro temp para alterar o valor das variáveis.