Lista 3 – Lab. De Estrutura de Dados 1 e Estrutura de Dados 1

Aluno: Anderson Carlos da Silva Morais

Matrícula: 2024011327

Questões

1º Crie um programa que peça ao usuário para digitar o número de alunos em uma turma. O programa deve usar essa informação para criar um vetor dinâmico que armazene as notas finais desses alunos. Peça ao usuário para entrar com a nota de dois alunos e em seguida mostre essas notas usando cout.

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

int main()
{
    setlocale(LC_ALL, "Portuguese");
    int n = 0;

    printf("Digite o número de alunos (mínimo 2):\n");
    scanf("%d", &n);

    if (n < 2)
    {
        printf("Número de alunos deve ser no mínimo 2.\n");
        return 1;
    }

    float *vet = (float *)malloc(n * sizeof(float));

    if (vet == NULL)
    {
        printf("Erro ao alocar memória.\n");
        return 1;
    }

    for (int i = 0; i < n; i++)
    {
        printf("Digite a nota do aluno %d: ", i + 1);
        scanf("%f", (vet + i));
    }

    printf("\nNotas digitadas:\n");
```

```
    for (int i = 0; i < n; i++)
    {
        printf("Aluno %d: %.2f\n", i + 1, *(vet + i));
    }

    free(vet);
    return 0;
}
```

2º Defina o registro balao como mostrado abaixo. Construa um programa para alocar dinamicamente uma variável do tipo balao. Peça ao usuário para entrar com valores para cada um dos membros e em seguida exiba o conteúdo do registro.

Em seguida mostre:

a) Como criar uma variável de tipo **balão***. peixe

Primeiro devemos garantir que existe espaço no vetor para que se possa armazenar uma entrada do tipo balão no vetor dinâmico.

b) Como alocar dinamicamente um registro de tipo **balão*** peixe.

Preencha a instância do vetor, de acordo com o índice do vetor

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

struct balao
{
    float diametro;
    char marca[20];
    int modelo;
};

void preencher(int i, struct balao *vetor)
{
    printf("Informe o diametro do balão\n");
    scanf("%f", &(vetor + i)->diametro);

    printf("Informe a marca do balão\n");
    scanf("%s", &(vetor + i)->marca);

    printf("Informe o modelo do balão\n");
    scanf("%i", &(vetor + i)->modelo);
}
```

```c
void exibir(struct balao *ptr)
{
    printf("Diâmetro: %.2f\n", ptr->diametro);
    printf("marca: %s\n", ptr->marca);
    printf("modelo: %d\n", ptr->modelo);
}

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

    int n;
    printf("Informe a quantidade de balões:\n");
    scanf("%d", &n);

    // Aloca memória para os balões
    struct balao *vet = (struct balao *)malloc(n * sizeof(struct balao));
    if (vet == NULL)
    {
        printf("Erro ao alocar memória.\n");
        return 1;
    }

    for (int i = 0; i < n; i++)
    {
        preencher(i, vet);
    }

    for (int i = 0; i < n; i++)
    {
        exibir((vet + i));
    }

    free(vet);
    return 0;
}
```

3º Construa um registro para guardar informações sobre um carro. Um carro deve ter um modelo, ano de fabricação e preço. Em seguida construa um vetor estático de 10 carros inicializando os dois primeiros carros respectivamente para "Vectra", 2009, R$58.000,00 e "Polo", 2008, R$45.000,00. Use um ponteiro para apontar para o segundo carro e exibir seus dados.

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

```c
#include <locale.h>

struct carro
{
    char modelo[100];
    int ano;
    float preco;
};

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

    struct carro vet[10] = {
        {"Vectra", 2009, 58000},
        {"Polo", 2008, 45000}};

    struct carro *ptr = vet; // vet já é o endereço do primeiro elemento
do array.
    printf("%s\n", (ptr + 1)->modelo);
    printf("%d\n", (ptr + 1)->ano);
    printf("R$ %.2f\n", (ptr + 1)->preco);
    printf("\n");

    printf("%s\n", (vet + 1)->modelo);
    printf("%d\n", (vet + 1)->ano);
    printf("R$ %.2f\n", (vet + 1)->preco);
    return 0;
}
```

4º Repita o exercício anterior criando um vetor dinâmico de carros. Ao invés de inicializar o vetor com valores predefinidos, peça ao usuário para digitar os dados de dois carros. Use uma função para receber o vetor de carros e exibir o valor total dos carros.

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

struct carro
{
    char modelo[100];
    int ano;
    float preco;
};

void preencher(int i, struct carro *vetor)
```

```c
{
    printf("Informe o modelo do carro %d\n", i + 1);
    scanf("%s", &(vetor + i)->modelo);

    printf("Informe o ano do carro %d\n", i + 1);
    scanf("%d", &(vetor + i)->ano);

    printf("Informe o preco %d\n", i + 1);
    scanf("%f", &(vetor + i)->preco);
}

void exibir(int n, struct carro *vetor)
{
    float sum = 0;
    for (int i = 0; i < n; i++)
    {
        sum += (vetor + i)->preco;
    }
    printf("O valor  e: R$ %.2f\n", sum);
}

int main()
{
    setlocale(LC_ALL, "Portuguese");
    int n = 0;
    printf("Informe a quantidade de carros: ");
    scanf("%d", &n);

    struct carro *vet = (struct carro *)malloc(n * sizeof(struct carro));
    if (vet == NULL)
    {
        printf("Erro ao alocar memória.\n");
        return 1;
    }

    for (int i = 0; i < n; i++)
    {
        preencher(i, vet);
        printf("\n");
    }

    exibir(n, vet);

    free(vet);
    return 0;
}
```

5º Construa um vetor dinâmico de alunos. O registro aluno deve ser composto por nome (ou matrícula), código da disciplina (número inteiro sem sinal), e situação da disciplina. A situação da disciplina deve ser uma enumeração com os valores: Aprovado, Trancado, Reprovado. Peça ao usuário para digitar o número de alunos do vetor e em seguida leia os dados do primeiro aluno. Para finalizar mostre os dados do primeiro aluno usando uma função que recebe um ponteiro para aluno.

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <string.h>

typedef enum
{
    APROVADO,
    TRANCADO,
    REPROVADO
} situacao;

typedef struct
{
    char nome[100];
    unsigned cod_disciplina;
    situacao sit_aluno;
} aluno;

const char *situacao_str(situacao s)
{
    switch (s)
    {
    case APROVADO:
        return "Aprovado";
    case TRANCADO:
        return "Trancado";
    case REPROVADO:
        return "Reprovado";
    default:
        return "Desconhecido";
    }
}

void exibir(int n, aluno *vetor)
{
    printf("Nome: %s\n", (vetor + n)->nome);
    printf("Código da disciplina: %d\n", (vetor + n)->cod_disciplina);
```

```
    printf("Situação do aluno: %s\n", situacao_str((vetor + n)-
>sit_aluno));
}

int main()
{
    setlocale(LC_ALL, "Portuguese");
    int n = 0;
    char sit_str[20];

    printf("Informe a quantidade de alunos: ");
    scanf("%d", &n);

    aluno *vet_aluno = (aluno *)malloc(n * sizeof(aluno));
    if (vet_aluno == NULL)
    {
        printf("Erro ao alocar memória.\n");
        return 1;
    }

    printf("Informe os valores do primeiro aluno (NOME |
CÓDIGO_DISCIPLINA | SITUAÇÃO)\n");
    scanf("%s %d %s", vet_aluno->nome, &vet_aluno->cod_disciplina,
sit_str);

    // Conversão da string para enum
    if (strcmp(sit_str, "APROVADO") == 0)
        vet_aluno->sit_aluno = APROVADO;
    else if (strcmp(sit_str, "TRANCADO") == 0)
        vet_aluno->sit_aluno = TRANCADO;
    else
        vet_aluno->sit_aluno = REPROVADO;

    exibir(0, vet_aluno);

    free(vet_aluno);
    return 0;
}
```

Teste de terminal:

Informe a quantidade de alunos: 1

Informe os valores do primeiro aluno (NOME | CÓDIGO_DISCIPLINA | SITUAÇÃO)

Anderson 01 APROVADO

Nome: Anderson

Código da disciplina: 1

Situação do aluno: Aprovado

6º As instruções abaixo resultam em um código válido? Explique o porquê.

float peso;

peso = 30;

cout << peso;

delete peso;

Peso não é uma estrutura dinâmica em que devemos realizar o gerenciamento de memória, é uma variável primitiva que o compilador consegue tratar adequadamente.

7º Declare um ponteiro para inteiro, aloque memória dinamicamente para ele e armazene o número 100 nessa memória. Mostre o conteúdo apontado. Peça que o usuário digite um novo número inteiro e armazene-o na memória previamente alocada. Libere o espaço alocado dinamicamente ao final do programa.

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

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

    int *ptr = (int *)malloc(sizeof(int));
    if (ptr == NULL)
    {
        printf("Erro de alocação de memória!\n");
        return 1;
    }
    *ptr = 100;

    printf("Conteúdo do ponteiro: %d\n", *ptr);

    printf("Informe um novo valor inteiro para o ponteiro: \n");
    scanf("%d", ptr);
```

```
    printf("Conteúdo do ponteiro: %d\n", *ptr);

    free(ptr);
    return 0;
}
```

8º Inicie o programa perguntando ao usuário quantos inteiros ele deseja armazenar em um vetor. Use a informação digitada para criar um vetor dinâmico com o espaço necessário para armazenar a quantidade de inteiros desejada. Depois disso, deixe que o próprio usuário preencha o vetor, utilizando o tamanho do vetor como condição de parada de um laço for. Mostre o vetor que foi preenchido através de outro laço e libere o espaço alocado dinamicamente ao final do programa.

Sugestão: utilize um laço for para percorrer um vetor

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

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

    int n = 0;
    printf("Quantos inteiros você deseja armazenar?\n");
    scanf("%d", &n);

    int *vet = (int *)malloc(n * sizeof(int));
    if (vet == NULL)
    {
        printf("Erro ao alocar memória!\n");
        return 1;
    }

    printf("Digite os %d inteiros:\n", n);
    for (int i = 0; i < n; i++)
    {
        printf("Elemento %d:\n", i + 1);
        scanf("%d", &vet[i]);
    }

    printf("\nvetor preenchido:\n");
    for (int i = 0; i < n; i++)
    {
```

```
        printf("%d\n", vet[i]);
    }
    printf("\n");

    free(vet);
    return 0;
}
```

9° Crie um registro "Local" com os campos nome, país e continente. Pergunte ao usuário quantos locais ele quer visitar nas próximas férias e crie um vetor de locais alocando dinamicamente o espaço de acordo com quantos locais ele quer visitar. Use um laço for para armazenar as informações dos locais que o usuário deseja visitar, e depois do armazenamento mostre os locais que ele escolheu. Libere o espaço alocado dinamicamente ao final do programa.

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

typedef struct
{
    char nome[50];
    char pais[50];
    char continente[50];
} Local;

void preencher(int n, Local *vet)
{
    for (int i = 0; i < n; i++)
    {
        printf("\nLocal %d:\n", i + 1);
        printf("Nome: ");
        fgets((vet + i)->nome, 50, stdin);
        (vet + i)->nome[strcspn((vet + i)->nome, "\n")] = 0;

        printf("País: ");
        fgets((vet + i)->pais, 50, stdin);
        (vet + i)->pais[strcspn((vet + i)->pais, "\n")] = 0;

        printf("Continente: ");
        fgets((vet + i)->continente, 50, stdin);
        (vet + i)->continente[strcspn((vet + i)->continente, "\n")] = 0;
    }
}
```

```c
void exibir(int n, Local *vet)
{
    for (int i = 0; i < n; i++)
    {
        printf("\nLocal %d:\n", i + 1);
        printf("  Nome: %s\n", (vet + i)->nome);
        printf("  País: %s\n", (vet + i)->pais);
        printf("  Continente: %s\n", (vet + i)->continente);
    }
}

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

    int n;
    printf("Quantos locais você quer visitar nas próximas férias?\n");
    scanf("%d", &n);
    getchar();

    Local *locais = (Local *)malloc(n * sizeof(Local));
    if (locais == NULL)
    {
        printf("Erro ao alocar memória!\n");
        return 1;
    }

    preencher(n, locais);
    printf("\nLocais que você escolheu:\n");
    exibir(n, locais);

    free(locais);
    return 0;
}
```

10º Defina um registro ASCII que armazena um caractere e um valor inteiro associado. Crie uma função que recebe um valor inteiro e um caractere, e retorna o endereço de um elemento do tipo ASCII, alocado dinamicamente na memória. O programa principal deve chamar a função passando valores lidos do usuário, receber o retorno em um ponteiro, exibir os valores de retorno e deletar a memória que foi alocada dentro da função.

Dica: funções que retornam memória alocada são perigosas. É fácil esquecer de guardar o endereço de retorno para dar o delete

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

typedef struct
{
    char caractere;
    int valor;
} ASCII;

ASCII *criarElementoASCII(int val, char ch)
{
    ASCII *novoElemento = (ASCII *)malloc(sizeof(ASCII));
    if (novoElemento == NULL)
    {
        printf("Erro ao alocar memória na função!\n");
        return NULL;
    }
    novoElemento->valor = val;
    novoElemento->caractere = ch;
    return novoElemento;
}

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

    int valor;
    char carac;

    printf("Digite um valor inteiro: ");
    scanf("%d", &valor);
    getchar();

    printf("Digite um caractere: ");
    scanf(" %c", &carac); // espaço antes de %c para consumir
newlines/espaços pendentes

    ASCII *ptr = criarElementoASCII(valor, carac);

    if (ptr != NULL)
    {
        // Exibe os valores de retorno
        printf("\nElemento ASCII criado:\n");
        printf("Caractere: %c\n", ptr->caractere);
        printf("Valor: %d\n", ptr->valor);

        // Deleta a memória que foi alocada dentro da função
        free(ptr);
```

```
    }
    else
    {
        printf("Não foi possível criar o elemento ASCII.\n");
    }

    return 0;
}
```