Curso de programação em C - FK

1 – Estrutura, variáveis e operadores.

2 – Estrutura condicional (If, If-Else e Switch-Case).

3 – Estrutura de repetição (For, While e Do-While).

4 – Vetores, Strings, Matrizes e Structs.

5 – Funções, Ponteiros, Alocação e Arquivos.

Programação em C

Aula 8 – Ponteiros

- Declarações e Inicialização de Variáveis Ponteiros
- Operadores de Ponteiros (%p, &, &*, *&).
- Aritmética de ponteiros.
- Chamando Funções por Referência: Variável, Vetor, Matriz e Struct.

Ponteiros

- Os ponteiros permitem programar, simular chamadas por referência e criar e manipular estruturas dinâmicas de dados, isto é, estruturas de dados que podem crescer e diminuir, como as listas encadeadas, filas (queues), pilhas (stacks) e árvores (trees).

- Os ponteiros são variáveis que contêm endereços de memória como valores. Normalmente, uma variável faz uma referência direta a um valor específico. Um ponteiro, por outro lado, contém um endereço de uma variável que contém um valor específico.

- Um nome de variável faz uma referência direta a um valor, e um ponteiro faz referência indireta a um valor.

Sintaxe

- Para declarar um ponteiro, você precisa especificar o tipo do valor ao qual o ponteiro aponta(int, float, char, etc.) e um asterisco (*) antes do nome da variável.

```
tipo *nome;
```

- Caso você crie um ponteiro para apontar outro ponteiro deverá preencher dois asterisco (**):

```
tipo **nome;
```

Exemplo 1

- Dada uma variável num1 do tipo inteiro com valor 7. Faça um ponteiro que imprima o endereço da variável e também imprima o endereço do ponteiro e seu valor apontado. E usando o ponteiro altere o valor de 7 para 22. Imprima a variável num1 novamente.

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

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

    int *ptr;
    int num1 = 7;
    ptr = &num1;

    printf("O valor da variável num1 %d", num1);
    printf("\nO endereço da varíavel num1 é %p",&num1);
    printf("\nO valor apontado do ponteiro ptr é %d",*ptr);
    printf("\nO endereço apontado pelo ponteiro ptr é %p", ptr);
    printf("\nO endereço do ponteiro ptr é %p", &ptr);
    printf("\nO valor do ponteiro ptr é %d\n", ptr);
    printf("\nSabendo que * e & complementam-se mutuamente."
    "\n&*ptr = %p\n*&ptr = %p\n\n",&*ptr, *&ptr);
    /* Os operadores & e * complementam-se mutuamente – quando os dois forem
    aplicados consecutivamente a Ptr em qualquer ordem, os mesmos resultados
    são impressos.*/
    *ptr = 22;
    printf("\nO valor de num1 mudou para %d",num1);
    printf("\nO valor apontado por *ptr mudou para %d",*ptr);

    return 0;
}
```

```
O valor da variável num1 7
O endereço da varíavel num1 é 000000000062FE44
O valor apontado do ponteiro ptr é 7
O endereço apontado pelo ponteiro ptr é 000000000062FE44
O endereço do ponteiro ptr é 000000000062FE48
O valor do ponteiro ptr é 6487620

Sabendo que * e & complementam-se mutuamente.
&*ptr = 000000000062FE44
*&ptr = 000000000062FE44



O valor de num1 mudou para 22
O valor apontado por *ptr mudou para 22
--------------------------------
Process exited after 0.01722 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Aritmética de ponteiros

- A aritmética de ponteiros permite manipular os endereços armazenados em ponteiros. As operações comuns incluem:

- Incremento e Decremento (++, --): Movem o ponteiro para o próximo ou anterior elemento do tipo que ele aponta. Lembre-se que é o endereço armazenado no ponteiro que desejamos incrementar. Ao realizar a operação, o ponteiro passará a apontar para o próximo dado do seu tipo no caso de incremento e o oposto no caso de decremento.

- Adição e Subtração (+, -): Permitem mover o ponteiro em múltiplas posições.

- Diferença entre ponteiros (ptr1 - ptr2): Retorna o número de elementos entre dois ponteiros do mesmo tipo.

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

int main()
{
    setlocale(LC_ALL,"");
    int arr[5] = {10, 20, 30, 40, 50};
    int *ptr = arr; // Ponteiro para o primeiro elemento do array

    printf("Endereço inicial: %p, Valor: %d\n", ptr, *ptr);

    ptr++; // Move para o próximo elemento
    printf("Endereço após incremento: %p, Valor: %d\n", ptr, *ptr);

    ptr += 2; // Avança mais dois elementos
    printf("Endereço após adição: %p, Valor: %d\n", ptr, *ptr);

    ptr--; // Retorna um elemento
    printf("Endereço após decremento: %p, Valor: %d\n", ptr, *ptr);

    arr[1] = *(arr + 1) +=2;
    printf("\n%d", arr[1]);
    arr[4] = *(arr + 4) * 3;
    printf("\n%d", arr[4]);
    arr[2] = *(arr + 2) - 20;
    printf("\n%d", arr[2]);

    return 0;
}
```

```
Endereço inicial: 000000000062FE30, Valor: 10
Endereço após incremento: 000000000062FE34, Valor: 20
Endereço após adição: 000000000062FE3C, Valor: 40
Endereço após decremento: 000000000062FE38, Valor: 30

22
150
10
--------------------------------
Process exited after 0.01522 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Funções por referência - Variável

- Chamamos a função secundária e passamos os endereços de uma ou mais variáveis.

Ex: Na função main chamamos a função

```
cadastro(&num1, &vogal);
```

A função cadastro recebe os valores com 2 ponteiros de mesmo tipo que num1 e vogal.

```
cadastro( int *n1, char *letra){

…

}
```

Exemplo 2

- Dados 2 números do tipo float representando 2 notas informadas pelo usuário. Gere 2 variáveis a e b para receber estes dados e faça uma função chamada troca usando ponteiros para trocar os valores de cada variável. Imprima o resultado da alteração na função main.

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

float troca();

int main()
{
    float a, b;
    printf("Digite a primeira nota (a): ");
    scanf("%f",&a);
    printf("Digite a segunda nota (b): ");
    scanf("%f",&b);

    troca(&a, &b);

    printf("\n\nO valor da variavel a = %.1f e de b = %.1f\n\n", a, b);
    return 0;
}
float troca(float *n1, float *n2)
{
    float n3;
    n3 = *n1;
    *n1 = *n2;
    *n2 = n3;
}
```

```
Digite a primeira nota (a): 7.5
Digite a segunda nota (b): 8.9


O valor da variavel a = 8.9 e de b = 7.5


--------------------------------
Process exited after 12.41 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Funções por referência - Vetor

- Na passagem de um vetor para outra função não precisamos utilizar o &. Informaremos na chamada da função secundária o nome do vetor e seu respectivo tamanho. Ex:

```
int numeros [3] = {1, 5, 9};

contagem (numeros, 3);

void contagem (int *vetor, int tam){

}
```

Exemplo 3

- Dado um vetor chamado matemática do tipo inteiro de tamanho 5 com valores 1, 2, 3, 4 e 5. Use uma função que usará um ponteiro para receber os valores do vetor e vai somar para cada índice recebido mais 2. Imprima na função principal o vetor matemática com os dados alterados.

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

void alteraVetor(int *vetor, int tamanho)
{
    int j;
    for (j = 0; j < tamanho; j++)
    {
        vetor[j] = *(vetor + j) + 2; // Podemos utilizar *(vetor + j) += 2;
        printf("\n%d",vetor[j]);
    }
}

int main()
{
    int matematica [5] = {1, 2, 3, 4, 5};
    int i;

    alteraVetor(matematica,5);
    printf("\n\n");

    for(i = 0; i < 5; i++)
    {
        printf("%d ", matematica[i]);
    }

    return 0;
}
```

```
3
4
5
6
7

3 4 5 6 7
--------------------------------
Process exited after 0.05681 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Funções por referência – Matriz string

- Um array bidimensional como char matriz[LINHAS][COLUNAS] é tratado como um ponteiro para arrays de char. Usando esse ponteiro, podemos acessar e alterar diretamente os elementos.

- No caso, uma função pode receber um ponteiro char (*matriz)[COLUNAS], o que representa uma matriz onde cada linha é um array de COLUNAS caracteres. A manipulação dentro da função usará aritmética de ponteiros para acessar os nomes.

Exemplo 4

- Dada uma matriz 3 por 50 solicite ao usuário digitar 3 nomes de pessoas para gravar na matriz nomes. Utilize uma função que receba os dados da matriz usando ponteiro e solicite ao usuário alterar os 3 nomes. Necessário que imprima a matriz na função main quando for alterado os nomes.

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

void trocaNomes();

int main()
{
    char nomes[3][50];
    int i;
    for (i = 0; i < 3; i++)
    {
        printf("Digite o %o nome: ", i+1);
        gets(nomes[i]);
    }
    printf("\n");
    trocaNomes(nomes);
    printf("\n");

    for (i = 0; i < 3; i++)
    {
        printf("Nome %d: %s\n", i+1,nomes[i]);
    }
    printf("\n");

    return 0;
}
```

```c
void trocaNomes(char (*troca)[50])
{
    int j;
    for (j = 0; j < 3; j++)
    {
        printf("Digite o novo nome para a pessoa %d: ", j + 1);
        gets(troca[j]);
    }
}
```

```c
void trocaNomes(char (*troca)[50])
{
    int j;
    for (j = 0; j < 3; j++)
    {
        printf("Digite o novo nome para a pessoa %d: ", j + 1);
        //gets(troca[j]);
        fgets(*(troca + j), 50, stdin);
    }
}
```

```
Digite o 1 nome: Rio Grande do Sul
Digite o 2 nome: Santa Catarina
Digite o 3 nome: Paraná

Digite o novo nome para a pessoa 1: Mato Grosso
Digite o novo nome para a pessoa 2: Goiás
Digite o novo nome para a pessoa 3: Distrito Federal

Nome 1: Mato Grosso
Nome 2: Goiás
Nome 3: Distrito Federal


--------------------------------
Process exited after 36.63 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Funções por referência - Struct

- Na linguagem C, é obrigatório usar o operador & ao passar uma struct por referência para uma função, a menos que a variável já seja um ponteiro.

- Por que é obrigatório?

- O operador & fornece o endereço de memória de uma variável para que ela possa ser manipulada por referência. Se você passar a struct diretamente sem usar &, estará passando uma cópia da struct, e as alterações feitas na função não afetarão o original.

- Obs: No próximo exemplo usei tanto usando o operador & e sem indicá-lo. Nos testes que realizei não apareceu mensagem de erro no DEVC++ sem usar o &. Mas seguindo a maioria dos livros que li sobre isso é indicado o uso do & desde que o struct não seja um ponteiro.

Exemplo 5

- Dada uma struct chamada cliente crie as variaveis nome, idade e peso. Na função main crie um vetor struct de cliente chamado lista com 5 elementos. Chame a função cadastro onde usará um ponteiro para receber a struct e cadastrar os 5 clientes. Imprima os dados da struct em outra função chamada consulta usando também o ponteiro de referência e após imprima na função principal.

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

struct cliente
{
    char nome[50];
    int idade;
    float peso;
};
void cadastro();
void consulta();

int main()
{
    struct cliente lista[5];
    cadastro (lista,5);
    system ("pause");
    consulta(lista,5);
    system ("pause");
    system("cls");

    int i;
    for (i = 0; i < 5; i++)
    {
        printf("%d - Nome: %s. Peso: %.2f. Idade: %d", i + 1, lista[i].nome, lista[i].peso, lista[i].idade);
        printf("\n");
    }
    return 0;
}
```

```c
void cadastro (struct cliente *dados,int tam)
{
    int i;
    for (i = 0; i < tam; i++)
    {
        printf("Nome do %do cliente: ", i + 1);
        gets(dados[i].nome);
        fflush(stdin);
        printf("Peso do %do cliente: ", i + 1);
        scanf("%f",&dados[i].peso);
        fflush(stdin);
        printf("Idade do %do cliente: ",i + 1);
        scanf("%d",&dados[i].idade);
        fflush(stdin);
        printf("\n");
    }
}

void consulta(struct cliente *consulta,int tamanho)
{
    int i;
    for (i = 0; i < tamanho; i++)
    {
        printf("Cliente %d - Nome: %s. Peso: %.2f. Idade: %d", i + 1, consulta[i].nome, consulta[i].peso, consulta[i].idade);
        printf("\n");
    }
}
```

```
Nome do 1o cliente: Roberto Carlos
Peso do 1o cliente: 60
Idade do 1o cliente: 80

Nome do 2o cliente: Bruce Dickinson
Peso do 2o cliente: 75
Idade do 2o cliente: 70

Nome do 3o cliente: Marisa Monte
Peso do 3o cliente: 55
Idade do 3o cliente: 47

Nome do 4o cliente: Joelma Calypso
Peso do 4o cliente: 77
Idade do 4o cliente: 52

Nome do 5o cliente: Sergio Reis
Peso do 5o cliente: 110
Idade do 5o cliente: 69

Pressione qualquer tecla para continuar. . .
Cliente 1 - Nome: Roberto Carlos. Peso: 60.00. Idade: 80
Cliente 2 - Nome: Bruce Dickinson. Peso: 75.00. Idade: 70
Cliente 3 - Nome: Marisa Monte. Peso: 55.00. Idade: 47
Cliente 4 - Nome: Joelma Calypso. Peso: 77.00. Idade: 52
Cliente 5 - Nome: Sergio Reis. Peso: 110.00. Idade: 69
Pressione qualquer tecla para continuar. . .
```

```
1 - Nome: Roberto Carlos. Peso: 60.00. Idade: 80
2 - Nome: Bruce Dickinson. Peso: 75.00. Idade: 70
3 - Nome: Marisa Monte. Peso: 55.00. Idade: 47
4 - Nome: Joelma Calypso. Peso: 77.00. Idade: 52
5 - Nome: Sergio Reis. Peso: 110.00. Idade: 69

--------------------------------
Process exited after 291.3 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Fim da aula 8

- Autor: Fernando Eduardo.

- Dúvidas encaminhe e-mail para:
dunano@outlook.com