Ponteiros

Lista de Exercícios - 08

CEFET-MG

Programação de Computadores I

Professor: Edwar Saliba Júnior

Exemplos de Utilização de Ponteiro e Alocação de Memória Dinâmica em Linguagem C

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

int main()
{
    int val, *p_val;

    p_val = &val;

    printf("\nDigite um valor: ");
    scanf("%d", &val);

    printf("\n\nValor armazenado na variável val: %d", *p_val);
    printf("\n\nO endereço da variável val é: %x", p_val);
    printf("\n\nO endereço da variável p_val é: %x", &p_val);
    printf("\n\n");

    return 0;
}
```

Figura 1: Ponteiro: utilização simplificada.

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

int main()
{
    char frase[30], *ptr;

    printf("\nDigite uma frase: ");
    gets(frase);

    // ptr recebe o endereço de frase na posição 0.
    ptr = &frase[0];


    // Percorre o vetor até encontrar '\0'.
    while(*ptr != '\0')
        *ptr++;

    // Imprime a frase invertida.
    printf("\n\nFrase invertida:\n");
    while(ptr >= frase)
        putchar(*ptr--);

    printf("\n");

    return 0;
}
```

Figura 2: Ponteiro: utilização com vetor de caracteres.

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

void imprimeVetor(int n, int *v){
    int j;

    for(j = 0; j < n; j++)
        printf("\nvetor[%d] = %d", j, v[j]);
}

int main()
{
    int *vetor, i, n;

    printf("\nEntre com o tamanho do vetor: ");
    scanf("%d",&n);

    // Alocação dinâmica de memória.
    vetor = (int *) malloc(n * sizeof(int));

    for(i = 0; i < n; i++){
        printf("\nDigite um valor: ");
        scanf("%d", &vetor[i]);
    }

    imprimeVetor(n, vetor);

    // Liberação da memória.
    free(vetor);

    return 0;
}
```

Figura 3: Ponteiro: Alocação dinâmica de memória (Vetor).

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

void imprimeVetor(int n, int *v){
    int j;

    for(j = 0; j < n; j++)
        printf("\nvetor[%d] = %d", j, v[j]);

    printf("\n");
}

int main()
{
    int *vetor, i, n;

    // Definindo o tamanho do vetor em tempo de execução.
    n = 2;

    // Alocação dinâmica de memória.
    vetor = (int *) malloc(n * sizeof(int));

    // Preenchendo o vetor.
    for(i = 0; i < n; i++){
        vetor[i] = i + 1;
    }

    imprimeVetor(n, vetor);

    // Mudando o tamanho do vetor para 3.
    n = 3;
    vetor = (int *) realloc(vetor, n * sizeof(int));

    // Atribui valor para última posição do vetor.
    vetor[2] = 5;

    imprimeVetor(n, vetor);

    // Liberação da memória.
    free(vetor);

    return 0;
}
```

Figura 4: Ponteiro: Realocação de memória dinâmica. Alterando o tamanho de um vetor.

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

float soma(float v1, float v2){
    return(v1 + v2);
}

int main()
{
    float valor1, valor2, total;

    // Declaração de um ponteiro para função.
    float (*fun)(float, float);

    printf("\nDigite o 1o. valor: ");
    scanf("%f", &valor1);
    printf("\nDigite o 2o. valor: ");
    scanf("%f", &valor2);

    // Atribuição do ponteiro para a função 'soma'.
    fun = soma;

    // Executando a função 'soma' através do ponteiro.
    total = (*fun)(valor1,valor2);

    printf("\n\nTotal: %f\n\n", total);

    return 0;
}
```

Figura 5: Ponteiro: Criação de um ponteiro para uma função.

Para a resolução dos exercícios a seguir, faça uso de ponteiros ou registros (*struct*).

1) Crie e teste a função: `void Troca3(int *a, int *b, int *c)` que retorna `a = b`, `b = c` e `c = a`.

2) Crie e teste a função: `void Ordena3(int *a, int *b, int *c)` que retorna `a >= b >= c`.

3) Crie e teste a função: `int Tamanho(char a[])` que retorna o tamanho da *string* digitada sem contar o caractere `'\0'`.

4) Criar um tipo (`struct`) `CONTA` que representa a conta corrente de um dado cliente de um Banco. O tipo `CONTA` deve possuir os seguintes campos:
   - `Nome` – Nome do correntista.
   - `NumCC` – Número da conta corrente (sem traço).

   Para testar este novo tipo realize as seguintes tarefas:
   - Criar uma função `CONTA fillConta(CONTA C1)` que preenche os campos da variável `C1`, do tipo `CONTA`, com informações de um dado correntista.

- Criar uma função `void mostraConta(CONTA C1)` que imprime na tela os dados relativos aos campos de uma variável `C1` do tipo `CONTA`.
- Testar as funções desenvolvidas nos itens anteriores através do programa principal que realiza as seguintes ações:
  - Criar um vetor `V` do tipo `CONTA` de tamanho `n` determinado em tempo de execução.
  - Empregar um laço para preencher os elementos de `V` através da função `fillConta`.
  - Mostrar cada um dos elementos de `V` com um laço que chama a função `mostraConta`.

5) Implemente as seguintes funções e um *software* para testá-las:

   `char *strrchr(char *s, char ch)`, que retorna o endereço da última ocorrência de `ch` em `s`. Caso não exista retorna `NULL`.

   `char *strstr(char *str1, char *str2)`, que retorna o endereço de `str1` em que ocorre pela primeira vez a *substring* `str2`, caso não exista retorna `NULL`.

   `char *primeiraVogal(char *s)`, que retorna o endereço em que ocorre a primeira vogal da *string* `s`. Caso não exista retorna `NULL`.

   `char *strins(char *str1, char *str2)`, que insere no início da *string* `str1`, a *string* que está em `str2`, retornando `str2`.

6) Escreva um programa que tenha no mínimo uma função. Faça um ponteiro para esta função e substitua as chamadas para a função pelo ponteiro criado.