Lista Endereço de Memória e Ponteiros

Prof. Bruno Travençolo

1) Crie um programa que contenha a seguinte variável

int val[5] = {2,4,**5**,8,10}

Utilizando a função Scanf, altere o valor de 5 para 6. Não use o operador & no scanf. Utilize uma variável do tipo unsigned int para guardar o endereço da posição do vetor.

2) Existe um tipo de dado em linguagem C chamado "ponteiro para inteiro". Esse tipo de dado serve para armazenar um endereço de memória de uma variável inteira. Existem ponteiros para outros tipos de dados (ex. float, double, char).
Um ponteiro é uma variável como qualquer outra do programa – sua diferença é que ela não armazena um valor inteiro, real, caractere ou booleano. Ela serve para armazenar **endereços de memória** (que, no fundo, são valores inteiros sem sinal).

Para declarar uma variável do tipo ponteiro, use a seguinte sintaxe:
tipo_de_dado ***nome_da_variável;**

Note que, a única diferença na declaração de uma variável do tipo ponteiro é a adição de um símbolo ***. Assim, para declarar um ponteiro para inteiro devemos fazer assim:

```
int *p;
int *proximo;
int *anterior;
int *abacaxi;
```

Como um ponteiro serve para receber um endereço de memória, podemos fazer, por exemplo, a seguinte operação:

```
int a = 40; // cria uma variável do tipo inteiro, chamada a, e inicializa
                  // com valor 40
int *p; // cria uma variável do tipo ponteiro para inteiro, chamada p, e o
        //conteúdo inicial é lixo
p = &a; // faz p receber o endereço de a. Dizemos que p aponta para a
```

Neste exemplo, dizemos que "p aponta para a" ou "p referencia a". Sabendo isso, continue o exercício 1 (não precisa criar um novo projeto) para:

(a) Copie o código acima e mostre o endereço da variável a de duas formas: uma usando **&a** e outra usando o ponteiro **p**. Os endereços devem ser os mesmos.
(b) Altere o valor da variável **a** usando o scanf sem usar o operador &.
(c) Utilizando a função scanf, altere o valor de 10 do vetor **val** para 20. Não use o operador & no scanf. Utilize uma variável do tipo ponteiro para inteiro para guardar o endereço da posição do vetor.
(d) Mostre o mapa de memória deste código ao final da execução.

3) Existem operadores específicos em C para trabalhar com ponteiros. Um desses operadores é o símbolo *. Note que o mesmo símbolo é usado para declarar um ponteiro e também para multiplicação – mas essas operações não são relacionadas. O operador * serve para desreferenciar (*dereferencing*) um ponteiro – ou

seja, ele retorna o **conteúdo** do endereço de memória que ele referencia/aponta. Ao usar o operador *, o tipo retornado será o mesmo tipo apontado pelo ponteiro. Veja um exemplo:

```
int a = 40; // cria uma variável do tipo inteiro, chamada a, e inicializa com valor 40
int *p; // cria uma variável do tipo ponteiro para inteiro, chamada p, e o conteúdo inicial é lixo
p = &a; // faz p receber o endereço de a

printf("\n O valor da variavel a eh: %d", *p);
```

O resultado do printf é 40. Note que foi usado %d (inteiro), mas a variável usada foi a **p** (que é int *), mas **p** foi antes desreferenciada. Utilizando o conceito de desreferenciamento, continue o exercício 1, mostrando agora todo o vetor val em um printf, mas com o segundo argumento sendo um ponteiro para os elementos de val. Não use a variável val. Mostre o mapa de memória deste código ao final da execução

4) No exercício anterior vimos que o desreferenciamento serve para obtermos o conteúdo da variável apontada pelo ponteiro. O desreferenciamento serve também para alteramos os valores das variáveis apontadas pelos ponteiros. Veja um exemplo

```
int a = 40; // cria uma variável do tipo inteiro, chamada a, e inicializa com valor 40
int *p; // cria uma variável do tipo ponteiro para inteiro, chamada p, e o conteúdo inicial é lixo
p = &a; // faz p receber o endereço de a
*p = 59;

printf("\n O valor da variavel a eh: %d", a);
```

O resultado do printf é 59. Note que está sendo mostrada a variável **a**. Utilize o conceito de desreferenciamento para alterar todo vetor val, subtraindo de cada elemento uma unidade.

5) Os operadores + e – funcionam com ponteiros. Chamamos de aritmética de ponteiros. Considere o código abaixo. Complete o código e preencha a tabela abaixo com os endereços de cada printf.

```
int  v1 = 1;
double v2 = 2;
char c = 'a';

int *p_v1;
double *p_v2;
char *p_c;

p_v1 = &v1;
p_v2 = &v2;
p_c = &c;


printf("\nEndereço de v1 %u", p_v1);
printf("\nEndereço de v2 %u", p_v2);
printf("\nEndereço de c %u", p_c);

printf("\nEndereço+1 de v1 %u", p_v1+1);
printf("\nEndereço+1 de v2 %u", p_v2+1);
printf("\nEndereço+1 de c %u", p_c+1);

printf("\nEndereço-1 de v1 %u", p_v1-1);
printf("\nEndereço-1 de v2 %u", p_v2-1);
printf("\nEndereço-1 de c %u", p_c-1);
```

Nome variável/ Endereços	Endereço var-2	Endereço var-1	Endereço var	Endereço var+1	Endereço var+2
p_v1					
p_v2					
p_c					

6) Discuta porque no exercício anterior ao somarmos/subtrairmos um valor de ponteiro há um comportamento diferente para cada variável.

7) Utilizando aritmética de ponteiros, mostre na tela o conteúdo da string char nome[] = “José Augusto”. Utilize o printf com %c e não %s

8) Ponteiros void: é um ponteiro genérico, que aponta para qualquer tipo. Sua declaração é deste tipo:

```
void *nome_ponteiro
```

Um ponteiro void pode apontar para qualquer tipo. Entretanto, antes de usá-los, é necessário fazer um type cast para o tipo que ele aponta. Por exemplo: (int *) ou (float *) ou (double *).
Faça um programa com dois números, um inteiro e um double. Crie um ponteiro void e use-o para ler os valores desses dois números com o comando **scanf**. Após a leitura, mostre na tela os números lidos, mas use no printf somente o ponteiro para mostrar os valores.

9) Imprima o conteúdo de um vetor de double de 10 posições utilizando aritmética de ponteiros e SEM declarar variáveis do tipo ponteiro (ou seja, o nome do vetor terá que ser usado como o ponteiro)

10) Imprima o conteúdo de um vetor de int de 10 posições da última posição até a primeira utilizando aritmética de ponteiros e SEM declarar variáveis do tipo ponteiro (ou seja, o nome do vetor terá que ser usado como o ponteiro)