Nicolas Gabriel Ceccato

EXERCÍCIOS DE FIXAÇÃO Nº 06.1 Básico sobre ponteiros

1. Qual a função dos operadores & e \* quando associados a ponteiros? Exemplifique com código em C

R: Na declaração da variável utiliza-se \* para indicar tratar-se de um ponteiro que está sendo declarado. Após isso utilizando \* em um ponteiro, acessa o valor apontado por ele. & aplicado sobre qualquer variável, recupera seu endereço.

1. Por que é importante inicializar um ponteiro antes do seu uso?

R: Se não inicializar o ponteiro ele irá apontar para um endereço desconhecido e o programa pode travar.

1. As variáveis são sempre armazenadas nos mesmos endereços?

R: Sim

1. O que é indireção?

R: é o acesso indireto de um valor por meio de um ponteiro.

1. Como o compilador distingue o \* usado para a multiplicação do \* usado para “desreferenciamento” (acesso às informações existentes no endereço contido em um ponteiro) e do \* usado para declarar um ponteiro?

R: na multiplicação o \* estará entre duas variáveis. No desreferenciamento estará em um ponteiro já inicializado. Para declarar o ponteiro você inicia com o tipo de valor que está no valor apontado por ele.

1. Como os elementos de uma matriz são armazenados na memória?

R: um ao lado do outro.

1. Mostre duas maneiras de obter o endereço do primeiro elemento da matriz data[].

R: &data[] e data. Considerando data do tipo inteiro:

Printf (“Endereço de data[] : %d , &data[]”);

Printf (“Endereço de data[] : %d , data”);

1. Quando uma matriz é passada para uma função, quais são as duas maneiras de determinar onde a matriz termina?

R: sizeof(matriz[][])/sizeof[tipo dessa matriz] = quantidade de termos dessa matriz

1. Cite seis operações que podem ser efetuadas com ponteiros e duas que não podem.

R: Passagem por referência, indireção, acesso direto à endereços de memória, alocação dinâmica de memória, polimorfismo

1. Suponha que você tenha dois ponteiros. Se o primeiro estiver apontando para o terceiro elemento de uma matriz do tipo int e o segundo para o quarto elemento da mesma matriz, que valor será obtido quando você subtrair o primeiro ponteiro do segundo?

R: -4

1. Suponha que a matriz da questão anterior contenha valores do tipo float, que valor seria obtido com a subtração dos dois ponteiros?

R: -8

1. Escreva uma declaração de um ponteiro chamado char\_ptr para uma variável do tipo char.

R: char a;

Char \*ptr = &a;

1. Se um programa contivesse uma variável int chamada coast, como você declararia e utilizaria um ponteiro chamado p\_coast para apontar para esta variável?

R: int coast;

Int \*p\_coast = &coast;

1. Continuando com o exercício 13, como você atribuiria o valor 100 à variável coast usando acesso direto e indireto?

R: direto: coast = 100;

Indireto : \*p\_coast = 100;

1. Continuando com o exercício 14, como você imprimiria o valor do ponteiro juntamente com o valor para o qual ele está apontando?

R: printf(“endereço: %i. Valor: %i”, p\_coast, coast); ou

printf(“endereço: %i. Valor: %i”, &a, \*p\_coast);

1. Mostre como atribuir o endereço de um valor do tipo float, chamado radius, a um ponteiro.

R: float radius;

float \*p\_radius = &radius;

1. Mostre duas maneiras de atribuir o valor 100 ao terceiro elemento da matriz data[].

R:

data[2] = 100;

&p\_data[2] = 100;

1. Explique o que faz cada linha do trecho do programa abaixo:

R: int x=1, y=2, z[10]; declara uma variavel x de valor 1, uma variável y de valor dois e um vetor de 10 elementos z. Todos do tipo inteiro.

int \*ip; declara um ponteiro do tipo int.

ip = &x; o ponteiro declarado acima ira apontar para o endereço da variável x.

y = \*ip; y recebe o valor de x.

\*ip = 0; x e y são zerados

ip=&z[0]; o ponteiro aponta p o primeiro elemento do vetor z.

1. Supondo o mesmo trecho de código do exercícios anterior, explique cada uma das operações aritméticas abaixo, que utilizam ponteiros:

R: a) y = \*ip+1; y recebe z[0] + 1;

b) \*ip += 1; z[0] = z[0] + 1;

c) ++\*ip; soma 1 a z[0];

d) (\*ip)++; soma 1 a z[0];

1. Considerando o fragmento de programa abaixo

{

int a[10];

int \*pa;

int aux;

...

pa = a;

}

Complete as equivalências abaixo usando os conceitos de aritmética de ponteiros:

aux = a[2] **aux = pa + 2**, usando “pa”;

aux = a[i] aux = **pa + i**, usando “pa”;

aux = a[2] aux = **a + 2**, usando “a”;

aux = a[i] aux = **a + i**, usando “a”

( a+2 ) **a[2]**, usando “a”;

(pa+1) **a[1]**, usando “a”