**Aluno: Humberto José Barbalho de Melo Junior**

**Lista de Exercícios (Ponteiros)**

1. Qual a utilidade do aprendizado do uso de ponteiro para aplicações em engenharia?

A Utilidade do aprendizado de ponteiros provém um melhor uso das variáveis de um código devido ao fato de evitar possíveis problemas do conteúdo das variáveis, utilizando a mecânica de armazenamento de endereço de memória.

1. Seja o seguinte trecho de programa:

**int**

i=

**3**

,j=

**5**

;

**int**

\*p, \*q;

p = &i;

q = &j;

Determine o valor das seguintes expressões:

◦ p == &i; ----- p = 0x28ff44

◦ \*p - \*q; ------ O Valor é -2, pois p recebe o valor de 3 e j recebe o valor de 5

◦ \*\*&p; ------ \*\*&p é um ponteiro que aponta para outro ponteiro, logo, recebendo o valor de 3.

◦ 3 - \*p/(\*q) + 7; ------ a expressão traduzida é 3 – 3 / (5) + 7, porém é indeterminada. Não dá

pra dividir 0/12.

1. Mostre o que será impresso por programa supondo que i ocupa o endereço 4094 na memória.

main(){

**int**

i=

**5**

, \*p;

p = &i;

printf(

"%x %d %d %d %d"

, p,\*p+

**2**

,\*\*&p,

**3**

\*\*p,\*\*&p+

**4**

)

;

}

Resposta = 28ff44 7 5 15 9

Dentro do Printf não tinha virgula, então deixei assim na resposta também.

1. Se i e j são variáveis inteiras e p e q ponteiros para int, quais das seguintes expressões de atribuição são ilegais?
   1. p = &i; ------ Expressão Legal
   2. \*q = &j; ------Expressão Ilegal
   3. p = &\*&i; ------Expressão Legal
   4. i = (\*&)j; ------ Expressão Ilegal
   5. i = \*&j; ------ Expressão Legal
   6. i = **&**&j; ------ Expressão Legal
   7. q = \*p; ------ Expressão Ilegal
   8. i = (\*p)` + \*q; ------ A Expressão é ilegal, mas apenas devido ao acento ali perdido. N sei se foi erro de digitação, porém, se tirar o acento, a expressão é Perfeitamente Legal.
2. Determine o que será mostrado pelo seguinte programa (compile-o, execute-o e verifique se foram obtidas as respostas esperadas).

**int**

**main**

()

{

**int**

valor;

**int**

\*p1;

**float**

temp;

**float**

\*p2;

**char**

aux;

**char**

\*nome =

"Ponteiros"

;

**char**

\*p3;

**int**

idade;

**int**

vetor[

**3**

;

]

**int**

\*p4;

**int**

\*p5;

/\* (a) \*/

valor =

**10**

;

p1 = &valor;

\*p1 =

**20**

;

printf(

"%d

\n

"

, valor);

/\* (b) \*/

temp =

**26.5**

;

p2 = &temp;

\*p2 =

**29.0**

;

printf(

"%.1f

\n

"

, temp);

/\* (c) \*/

p3 = &nome[

**0**

;

]

aux = \*p3;

printf(

"%c

\n

"

, aux);

/\* (d) \*/

p3 = &nome[

**4**

;

]

aux = \*p3;

printf(

"%c

\n

"

, aux);

/\* (e) \*/

p3 = nome;

printf(

"%c

\n

"

, \*p3);

/\* (f) \*/

p3 = p3 +

**4**

;

printf(

"%c

\n

"

, \*p3);

/\* (g) \*/

p3--;

printf(

"%c

\n

"

, \*p3);

/\* (h) \*/

vetor[

**0**

]

=

**31**

;

vetor[

**1**

]

=

**45**

;

vetor[

**2**

]

=

**27**

;

p4 = vetor;

idade = \*p4;

printf(

"%d

\n

"

, idade);

/\* (i) \*/

p5 = p4 +

**1**

;

idade = \*p5;

printf(

"%d

\n

"

, idade);

/\* (j) \*/

p4 = p5 +

**1**

;

idade = \*p4;

printf(

"%d

\n

"

, idade);

/\* (l) \*/

p4 = p4 -

**2**

;

idade = \*p4;

printf(

"%d

\n

"

, idade);

/\* (m) \*/

p5 = &vetor[

**2**

]

-

**1**

;

printf(

"%d

\n

"

, \*p5);

/\* (n) \*/

p5

`

;

printf(

"%d

\n

"

, \*p5);

**return**

(

**0**

)

;

}

Resposta :

a) 20 h) 31

b) 29.0 i) 45

c) P j) 27

d) e l) 31

e) P m) 45

f) e n) 45

g) t

1. Determine o que será mostrado pelo seguinte programa (compile-o, execute-o e verifique se foram obtidas as respostas esperadas).

**int**

**main**

(

**void**

){

**float**

vet[

**5**

]

=

{

**1.1**

,

**2.2**

,

**3.3**

,

**4.4**

,

**5.5**

}

;

**float**

\*f;

**int**

i;

f = vet;

printf(

"contador/valor/valor/endereco/endereco"

)

;

**for**

(

i =

**0**

; i <=

**4**

; i++){

printf(

"

\n

i = %d"

,i);

printf(

"vet[%d] = %.1f"

,i, vet[i]);

printf(

"\*(f + %d) = %.1f"

,i, \*(f+i));

printf(

"&vet[%d] = %X"

,i, &vet[i]);

printf(

"(f + %d) = %X"

,i, f+i);

}

}

Resposta :

contador /valor /valor /endereco /endereco

i = 0 vet[0] = 1.1\*(f + 0) = 1.1&vet[0] = 61FF04(f + 0) = 61FF04

i = 1 vet[1] = 2.2\*(f + 1) = 2.2&vet[1] = 61FF08(f + 1) = 61FF08

i = 2 vet[2] = 3.3\*(f + 2) = 3.3&vet[2] = 61FF0C(f + 2) = 61FF0C

i = 3 vet[3] = 4.4\*(f + 3) = 4.4&vet[3] = 61FF10(f + 3) = 61FF10

i = 4 vet[4] = 5.5\*(f + 4) = 5.5&vet[4] = 61FF14(f + 4) = 61FF14

1. Assumindo que pulo[] é um vetor do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento do vetor?

◦ \*(pulo + 2);

◦ \*(pulo + 4);

◦ pulo + 4;

◦ pulo + 2;

Resposta: Letra A, \*(pulo + 2);

1. Considerando a declaração

**int**

mat[

**4**

]

, \*p, x;

quais das seguintes expressões são válidas? Justifique.

◦ p = mat + 1; Isso é Valido, pois o ponteiro está recebendo a posição do elemento mat +1

◦ p = mat; Isso é Válido, ele está recebendo o endereço de mat da primeira posição

◦ p =`mat; Isso é Válido, ele está recebendo o endereço de mat da primeira posição

◦ x = (\*mat);` Isto é Valido, uma variável está recebendo o valor da primeira posição do vetor

“mat”

1. O que fazem os seguintes programas em C?

**int**

**main**

(){

**int**

vet[] = {

**4**

,

**9**

,

**13**

}

;

**int**

i;

**for**

(

i

=

**0**

;i<

**3**

;i++){

printf(

"%d "

,\*(vet+i));

}

}

Resposta: Printa os Valores: 4 9 13

**int**

**main**

(){

**int**

vet[] = {

**4**

,

**9**

,

**13**

}

;

**int**

i;

**for**

(

i

=

**0**

;i<

**3**

;i++){

printf(

"%X "

,vet+i);

}

}

Resposta : Printa os Valores: 61FF10 61FF14 61FF18

1. Seja x um vetor de 4 elementos, declarado da forma TIPO x[4]. Suponha que depois da declaração, x esteja armazenado no endereço de memória 4092 (ou seja, o endereço de x[0]). Suponha também que na máquina seja usada uma variável do tipo char ocupa 1 byte, do tipo int ocupa 2 bytes, do tipo float ocupa 4 bytes e do tipo double ocupa 8 bytes. Quais serão os valores de x+1, x+2 e x+3 se:

◦ x for declarado como char? ------ x + 1 = 4093, x + 2 = 4094, x + 3 = 4095

◦ x for declarado como int? ------ x+1 = 4094, x + 2 = 4096, x + 3 = 4098

◦ x for declarado como float? ------ x + 1 = 4096, x + 2 = 4100, x + 3 = 4104

◦ x for declarado como double? ------ x + 1 = 4100, x + 2 = 4108, x + 3 = 4116

Implemente um programa de computador para testar estas suposições e compare as respostas oferecidas pelo programa com as respostas que você idealizou.

Resposta:

x + 1 = 4093, x + 2 = 4094, x + 3 = 4095 / x + 1 = 4093, x + 2 = 4094, x + 3 = 4095

x+1 = 4094, x + 2 = 4096, x + 3 = 4098 / x + 1 = 4094, x + 2 = 4096, x + 3 = 4098

x + 1 = 4096, x + 2 = 4100, x + 3 = 4104 / x + 1 = 4096, x + 2 = 409A, x + 3 = 409E

x + 1 = 4100, x + 2 = 4108, x + 3 = 4116 / x + 1 = 409A, x + 2 = 40A2, x + 3 = 40AA

1. Suponha que as seguintes declarações tenham sido realizadas:

**float**

aloha[

**10**

]

, coisas

[

**10**

][

**5**

]

, \*pf, value =

**2.2**

;

**int**

i=

**3**

;

Identifique quais dos seguintes comandos é válido ou inválido:

* 1. aloha[2] = value; ------ Válido
  2. scanf("%f", &aloha); ------ Válido
  3. aloha = "value"; ------Inválido
  4. printf("%f", aloha); ------Válido
  5. coisas[4][4] = aloha[3]; ------ Válido
  6. coisas[5] = aloha; ------ Inválido
  7. pf = value; ------ Válido
  8. pf = aloha; ------Inválido

1. O que é um ponteiro para uma função? Pesquise na Internet referências sobre o assunto e escreva um pequeno programa exemplificando o uso deste recurso.

Resposta: Um ponteiro é um tipo de variável capaz de conter um dado de outra variável, sem ter que menciona-la, assim, preservando seu valor original, como se fosse um backup de segurança, onde caso o valor seja adulterado, outra parte do programa ainda possuirá o valor correto.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

int main ()

//Questão 17 da Lista de Ponteiros

//Aluno: Humberto José Barbalho de Melo Junior

{

setlocale(LC\_ALL, "Portuguese"); //Setando para conseguir usar acentos do português

int X = 10, \*p, \*q;

p = X - 1;

q = X +20;

printf ("%i\n", p);

printf ("%i\n", q);

}

1. Implemente em linguagem C uma função em um programa de computador que leia n valores do tipo float e os apresente em ordem crescente. Utilize alocação dinâmica de memória para realizar a tarefa.
2. Reimplemente o programa da questão anterior utilizando a função qsort() do C. Comente o seu código, explicando o que faz cada uma das linhas.
3. Utilize a ideia do ponteiro para função pela função qsort() para implementar sua própria função de ordenação. Para isso, sua função deverá receber, entre outros argumentos, um ponteiro para a função de comparação.
4. Procure na internet mecanismos que possibilitem medir tempos de execução de rotinas computacionais. Geralmente, estas medidas são realizadas com o auxílio de funções em C que lêem a hora no sistema (sistemas Unix e Windows geralmente usam funções diferentes). Utilizando os conhecimentos que você obteve com sua pesquisa, meça os tempos de execução das implementações que você criou para os dois problemas de ordenação anteriores e compare os resultados obtidos.
5. Escreva uma função em c que escreva em um vetor a soma dos elementos correspondentes de outros dois vetores (os tamanhos dos vetores devem ser fornecidos pelo usuário). Por exemplo, se o primeiro vetor contiver os elementos 1, 3, 0 e -2, e o segundo vetor contiver os elementos 3, 5, -3 e 1, o vetor de soma terá valores resultantes iguais a 4, 8, -3 e -1. A função deve receber 4 argumentos: os nomes dos três vetores e o número de elementos presentes em cada vetor.
6. Crie uma função capaz de realizar multiplicação matricial da forma C=A B. A função deve receber 6 argumentos: os ponteiros para as matrizes A, B e C, o número de linhas e colunas de A e o número de colunas de B (assuma que o número de coluna de A é igual ao número de linhas de B). O resultado da multiplicação deve ficar armazenado em C. Crie um programa para testar sua implementação, capaz de utilizar a função de multiplicação e imprimir as três matrizes. A função criada para multiplicação não deve realizar nenhum tipo de saída de dados no terminal.
7. Pesquise na Internet sobre o uso da biblioteca libGC, que implementa um coletor de lixo em C. O processo de instalação dessa biblioteca em sistemas Windows pode ser um pouco trabalhoso. Entretanto, em sistemas Unix, a instalação é bem simples, de sorte que se recomenda a resolução desta questão em uma máquina, por exemplo, com Linux instalado. Caso seja usuário Windows e não queira instalar sistemas alternativos em seu computador, use uma ferramenta de virtualização (ex: VirtualBox) para instalar e executar o Linux sem alterar a instalação original de seu computador. Prepare um pequeno programa-exemplo demonstrando como usar a biblioteca.
8. Com base no programa-exemplo da questão anterior, proponha uma aplicação que permita comparar o desempenho das funções tradicionais de alocação/liberação de memória da linguagem c (malloc/free) com as funções de gerenciamento de memória da biblioteca que você escolheu. A aplicação deverá ser capaz de ressaltar possíveis atrasos de tempo provenientes do coletor de lixo utilizado.