**EXERCICIO PARA ESTUDO DE PONTEIROS:**

**1. Seja o seguinte trecho de programa:**

**int i=6,j=3;**

**int \*p, \*q;**

**p = &i;**

**q = &j;**

**2. Qual é o valor das seguintes expressões?**

**a) p == &i**

**b) \*p - \*q**

**c) \*\*&p**

**d) 3 - \*p/(\*q)+7**

**e) (\*p)++**

**3. Qual é a maneira correta de referenciar (acessar o valor de) num, assumindo que o**

**endereço de num foi atribuído ao ponteiro indica:**

**a) \*indica**

**b) int \*indica**

**c) indica**

**4. Qual será a saída deste programa supondo que i ocupa o endereço 4094 na memória?**

**int main() {**

**int i=5, \*p;**

**p = &i;**

**printf(“%p %d %d %d %d \n”, p,\*p+2,\*\*&p,3\*\*p,\*\*&p+4);**

**}**

**5. Se i e j são variáveis inteiras e p e q ponteiros para int, quais das seguintes expressões**

**de atribuição são ilegais?**

**a) p = &i; b) \*q = &j; c) p = &\*&i; d) i = (\*&)j;**

**e) i = \*&j; f) i = \*&\*&j; g) q = \*p; h) i = (\*p)++ + \*q**

**6. Qual é o objetivo da variável temp no código abaixo? A variável está sendo usada de**

**forma correta? Em caso negativo, qual é o erro? Caso tenha erra, pense em uma**

**possível solução.**

**int \*q, \*p, \*temp;**

**int i,j;**

**p = &i;**

**q = &j;**

**\*temp = \*p;**

**\*p= \*q;**

**\*q = \*temp**

**7. Quais os valores impressos no código abaixo? Compile e execute esse código e**

**verifique se você acertou? Caso tenha errado, tente dar uma justificativa.**

**int main(){**

**int i = 5, j = 3, \*p, \*q;**

**p = &i;**

**q = &j;**

**printf("%d %d \n", \*p, (\*p)++ + \*q);**

**}**

**RESPOSTAS:**

**1 - Seja o seguinte trecho de programa:**

**int i=6,j=3;**

**int \*p, \*q;**

**p = &i;**

**q = &j;**

1. a) O correto seria p = &i, que daria resultado **6**

b) **3**

c) \*\*&p é mesmo que \*p pois (\*&) são cortados e \*p= i = **6**

d) **8**

e) (\*p)++ = **6** agora se fosse ++(\*p) = 7 pois o “++ contador” pela esquerda acrescenta o ponteiro, porém pela direita não.

1. **B** – Pois deve se deve colocar o \* na frente para referenciar o ponteiro e declará-lo também do mesmo tipo da sua variável que está sendo referenciado.
2. Saída – **4094 7 5 15 9**

quando joga o **p** no printfque não seja referente a função ele irá indicar o endereço da variável i=4094; \*p+2=5+2=7; \*\*&p=\*p=5; 3\*\*p=3\*5=15; \*\*&p+4=\*p+4=5+4=9.

1. a) **correto** b) **incorreto** (coreto seria q=&j) c) **correto** (p=&i)

d)**incorreto** (i != j) e)**incorreto** (i != j) f) **incorreto** (i != j) g) **incorreto** (\*p=6 e q=&j) h) **incorreto** (pois i=6 e (\*p)++ + \*q = 6+3=9)

Em alguns desses casos não daria erro de compilação porém apresentaria outro resultado de saída. Exemplo na questão **h** o **i** deixaria de ser 6 e receberia 9.

1. **O objetivo da variável temp é servir como variável auxiliar para trocar os valores de i e j. A variável não está sendo usada de forma correta pois ela não deveria ser ponteiro e sendo assim apresenta erro de compilação ao executar o programa.**

**Uma possível solução seria conforme abaixo:**

**#include<stdio.h>**

**int main() {**

**int \*q, \*p, temp;**

**int i=2,j=3;**

**printf ("\nValor de I e J respectivamente: %d %d\n",i, j);**

**p = &i;**

**q = &j;**

**temp = \*p;**

**\*p = \*q;**

**\*q = temp;**

**printf ("\nNovo valor de I e J respectivamente: %d %d\n",i, j);**

**return 0;**

**}**

**Foi incrementado valores nas variáveis I e J para validação.**

1. **Os valores impressos no código abaixo são 6 8**

**Isso ocorre pois o printf em C executa da direita para a esquerda (o virgula faz a separação). Ou seja:**

**printf("%d %d \n", \*p, (\*p)++ + \*q); 6 , 5 + 3**

**Da virgula para a direita ele pega o \*p=(5) e soma \*q=(3), após esse comando ele incrementa o ponteiro \*p com o ++=(+1) exibindo na tela o 6 (5+1).**

Caso fosse da maneira abaixo:

printf("%d %d \n", \*p, ++(\*p) + \*q); 6 , 6 + 3

Ele executaria primeiro a direita da virgula, normalmente, e nesse caso incrementaria o **\*p** pois o **++** está a esquerda do ponteiro \*p **(++(\*p)).** E após isso executaria normalmente o lardo esquerdo da virgula com o \*p já incrementado.