#### Exercícios aplicando *ponteiros*

***Soluções***

**Questão 1**

**Quais serão os valores de x, y e p ao final do trecho de código abaixo?**

**int x, y, \*p;**

**y = 0;**

**p = &y;**

**x = \*p;**

**x = 4;**

**(\*p)++;**

**--x;**

**(\*p) += x;**

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **p** | **\*p** |
| **?** | **0** | **?** | **?** |
| **?** | **0** | **&y** | **0** |
| **0** | **0** | **&y** | **0** |
| **4** | **0** | **&y** | **0** |
| **4** | **1** | **&y** | **1** |
| **3** | **1** | **&y** | **1** |
| **3** | **4** | **&y** | **4** |

**Questão 2**

**Qual será o valor de x e y ao final do trecho de código abaixo?**

**int x, y, \*p;**

**y = 0;**

**p = &y;**

**x = \*p;**

**x = 4;**

**(\*p)++;**

**(\*p) += --x;**

**printf ("x = %d\n", x);**

**printf ("y = %d\n", y);**

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **p** | **\*p** |
| **?** | **0** | **?** | **?** |
| **?** | **0** | **&y** | **0** |
| **0** | **0** | **&y** | **0** |
| **4** | **0** | **&y** | **0** |
| **4** | **1** | **&y** | **1** |
| **3** | **4** | **&y** | **4** |
| **3** | **4** | **&y** | **4** |

**Questão 3**

**Aponte os erros nos trechos de código a seguir e proponha uma forma correta.**



void main() {

int x, \*p;

x = 100;

p = x;

printf(“Valor de x: %d.\n”, \*p);

}

O erro está na instrução p = x; Corigindo...

void main() {

int x, \*p;

x = 100;

p = &x;

printf(“Valor de x: %d.\n”, \*p);

}



void troca (int \*i, int \*j) {

int \*temp;

\*temp = \*i;

\*i = \*j;

\*j = \*temp;

}

O erro está na declaração e acessos à variável **temp**. Corigindo...

void troca (int \*i, int \*j) {

int temp;

temp = \*i;

\*i = \*j;

\*j = temp;

}

int x, \*p;

x = 10;

\*p = x;

printf ("%d", \*p);

O erro é que o ponteiro \*p está indefinido. Corrigindo...

int x, \*p;

x = 10;

p = &x;

printf ("%d", \*p);

void main(){

char \*a, \*b;

a = "abacate";

b = "uva";

if (a < b)

printf ("%s vem antes de %s no dicionario", a, b);

else

printf ("%s vem depois de %s no dicionario", a, b);

}

O erro está na expressão logica do comando ***if***, que está comparando endereços e não conteúdos. **Nota**: **a** e **b** são ***vetores*** de ***char***, ou seja, ***strings***, portanto as atribuições são perfeitamente válidas. Corrigindo...

void main(){

char \*a, \*b;

a = "abacate";

b = "uva";

if (\*a < \*b)

printf ("%s vem antes de %s no dicionario", a, b);

else

printf ("%s vem depois de %s no dicionario", a, b);

}

1. Neste caso, corrija o código de forma que o trecho mostre o valor 10 na tela

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

p = &x;

q = &p;

x = 10;

printf("\n%d\n", &q);

**q** é um *ponteiro* para um *ponteiro* para inteiro, isto é, **q 🡪 p 🡪 x**;. Portanto, para mostrar o valor 10, o comando *printf* tem que referenciar o conteúdo apontado pelo *ponteiro* apontado por **q**, ou seja, \*\*q. Assim, corrigindo...

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

p = &x;

q = &p;

x = 10;

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

**Questão 4**

Qual o resultado d o código abaixo? Explique cada linha de código.

01: int x = 100, \*p, \*\*pp;

02: p = &x;

03: pp = &p;

04: printf("Que valor e´: %d?\n ", \*\*pp);

05: printf("Que valor e´: %d?\n ", \*pp);

06: printf("Que valor e´: %d?\n ", pp);

Linha 01: cria variável x e atribui a essa valor 100, cria ponteiro p, para int, cria ponteiro pp para ponteiro para int;

Linha 02: ponteiro p recebe endereço de x: **p 🡪 x = 100**;

Linha 03: ponteiro pp recebe endereço de p: **pp 🡪 p 🡪 x = 100**;

Linha 04: print no conteúdo apontado pelo ponteiro apontado por **pp** ou seja, print **100**;

Linha 05: print no conteúdo do ponteiro apontado por **pp**, ou seja, print **endereço de X**;

Linha 06: print no conteúdo de pp, ou seja **endereço de p**.

**Nota**: a instrução printf("%d\n ", &pp); dá print no **endereço de pp**.

**Questão 5**

Sabe-se que na linguagem C, toda variável do tipo ***vetor*** na verdade é um ***ponteiro*** para o **endereço** da primeira posição do vetor. Por exemplo, quando se declara:

int vet[10];

na verdade, **vet** é um ponteiro para a posição [0] do vetor, ou seja: vet = &vet[0].

Sabe-se ainda que:

vet+1 = &vet[1]; vet+2 = &vet[2]; ... vet+9 = &vet[9]

O **conteúdo** de qualquer elemento pode ser acessado com a construção \*(vet+i) ou seja:

\*(vet+i) = vet[i]

Considerando as premissas acima da linguagem C, seja **a[ ]** um vetor qualquer declarado em C, independente do tipo de dado e tamanho do vetor, e **pa** um ponteiro para o mesmo tipo de **a[ ]**. Avalie as premissas a seguir e responda V ou F, justificando a resposta nestes casos:

1. Após a atribuição pa = &a[0]; **pa** e **a** possuem valores idênticos. V
2. A atribuição da letra a) acima é equivalente a: pa = a; V
3. **a[i]** é equivalente a **\*(a+i)**. V
4. **&a[i]** é equivalente a **a+i**. V
5. **a+i** é o *endereço* do i-ésimo elemento do vetor **a**. V
6. **pa = a;** é uma instrução válida e atribui ao ponteiro **pa** o endereço do vetor **a**. V
7. **pa++;** é uma instrução válida e atribui a **pa** o endereço do elemento posterior ao que ele apontava antes da instrução. V
8. **Pa--;** é uma instrução válida e atribui a **pa** o endereço do elemento anterior ao que ele apontava antes da instrução. V
9. a = pa é uma instrução válida. F – uma variável *vetor* recebe o endereço no momento em que é alocado e não pode mais ser atribuído com outro endereço.
10. a++ é uma instrução válida. F – o endereço da variável ***a*** (vetor) não pode ser modificado.

**Questão 6**

No caso de passagem de parâmetros por ***referência***, os ponteiros são utilizados para permitir que seja acessada uma região de memória ocupada por uma variável não "enxergada" pela função chamada, ou seja, por uma variável fora do escopo da função. Para isso, a função recebe e armazena em *variáveis dinâmicas* (***ponteiros***) os endereços das variáveis fora do seu escopo.

Escreva uma função ***minmax*** que receba um vetor de inteiros v[0..n-1] e os endereços de duas variáveis inteiras, digamos ***min*** e ***max***, pesquise e atribua nessas variáveis, respectivamente, os valores do menor e maior elemento do vetor. Escreva ainda uma função ***main*** que gere aleatoreamente valores para o vetor e use a função ***minmax*** para conhecer e mostrar na tela os valores mínimo e máximo do vetor.

#include <stdio.h>

#include <time.h>

#define N 10

void minmax(int \*v, int \*min, int \*max){

\*min = \*max = v[0];

int i;

for(i= 1; i < N; i++)

if(v[i] > \*max)

\*max = v[i];

else

if(v[i] < \* min)

\*min = v[i];

}

void main(){

int vet[N], k, menor, maior;

srand(time(NULL));

for(k = 0; k < N; k++)

vet[k] = rand()%101;

minmax(vet, &menor, &maior);

printf("\n Vet = { ");

for(k = 0; k < N-1; k++)

printf("%2d, ", vet[k]);

printf("%2d }\n", vet[N-1]);

printf("\n Valor MINIMO em Vet: %d", menor);

printf("\n Valor MAXIMO em Vet: %d\n", maior);

}

**Questão 7**

Faça um programa com uma função que receba como parâmetro os *ponteiros* de três números inteiros, A, B e C, classifique e retorne estes números de forma que o menor esteja no parâmetro A e o maior no parâmetro C. O programa deve ainda ter uma função ***main*** que leia pelo teclado os 3 números, chame a função para ordená-los e mostre os números ordenados.

#include <stdio.h>

void troca(int \*i, int \*j){

int temp = \*i;

\*i = \*j;

\*j = temp;

}

void ordena(int \*a, int \*b, int \*c){

int temp;

if(\*a > \*b)

troca(a, b);

if(\*a > \*c)

troca(a, c);

if(\*b > \*c)

troca(b, c);

}

void main(){

int A, B, C;

printf("Informe valor de A: ");

scanf("%d", &A);

printf("Informe valor de B: ");

scanf("%d", &B);

printf("Informe valor de C: ");

scanf("%d", &C);

printf("\n\n Valores como lidos: %d, %d, %d. ", A, B, C);

ordena(&A, &B, &C);

printf("\n\n Valores ordenados: %d, %d, %d. \n", A, B, C);

}