LISTA AEDS 5 – PONTEIRO

EXERCÍCIO NÚMERO 1

#include <stdio.h>

int main() {

int x, y;

printf("digite dois números:\n");

scanf("%d %d", &x, &y);

printf("x: %p\n", &x);

printf("y: %p\n", &y);

if (&x > &y){

printf("o maior valor é: %p", &x); //p= qunado quer o endereço, &= para combinar com o endereço

} else {

printf("o maior valor é: %p", &y);

}

return 0;

}

EXERCÍCIO NÚMERO 2

//p++ significa que vai somar o valor de P mais 1

//(\*p)++ significa que vai somar o valor que p ponteiro indica mais 1

//\*(p++) significa que vai somar o valor do endereço de P mais 1

//a informação que se refere a expressão \*(p+10) é a \*(p++), mas ao invés de adicionar +1, vai adicionar +10

EXERCÍCIO NÚMERO 3

#include <stdio.h>

int main() {

int valor=5, \*valorPtr = &valor;

printf("valor: %p\n", &valor);

printf("&valorPtr: %p\n", &valorPtr);

printf("\*valorPtr: %d\n", \*valorPtr);

return 0;

}

//letra B esta certa porque o valor do ponteiro apota para o valor da variável

//letra C esta certa porque O valorPtr aponta para o endereço de valor da variável valor

EXERCÍCIO NÚMERO 4

#include <stdio.h>

int main()

{

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

p = &x;

q = &p;

x = 10;

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

return(0);

}

//para resolver o erro basta trocar o &q (endereço de q) por \*\*q (que faz referência ao p, que faz referência ao x, que é o valor) , ou seja, o & imprime o endereço do ponteiro, não o seu valor

EXERCÍCIO NÚMERO 5

#include <stdio.h>

int main(void) {

int x = 1, \*px = &x; //ponteiro de x é igual ao endereço do x

double y = 5, \*py = &y; //ponteiro de y é igual ao endereço do y

char c = 'f', \*pc = &c; //ponteiro de c é igual ao endereço do c

printf("o valor inteiro de X é %d e seu endereço é: %p\n", x, &x); //%p é para mostrar o endereço e não o valor em si do que ta lá dentro

printf("o valor do ponteiro do inteiro é %d e seu endereço é: %p\n", \*px, &px);

printf("\no valor inteiro de Y é %.2lf e seu endereço é: %p\n", y, &y);

printf("o valor do ponteiro do real é %.2lf e seu endereço é: %p\n", \*py, &py);

printf("\no valor inteiro de C é %c e seu endereço é: %p\n", c, &c);

printf("o valor do ponteiro do caracter é %c e seu endereço é: %p\n", \*pc, &pc);

printf("\ndigite um novo valor de x:\n");

scanf("%d", px); //px = vai mudar pelo próprio ponteiro, porque o que acontecer com ele, também acontece com o X

printf("digite um novo valor de y:\n");

scanf("%lf", py);

printf("digite um novo valor de c:\n");

scanf(" %c", pc);

printf("\nNOVOS VALORES\n");

printf("o novo valor inteiro de X é %d e seu novo endereço é: %p\n", x, &x);

printf("o novo valor do ponteiro do inteiro é %d e seu novo endereço é: %p\n", \*px, &px);

printf("\no novo valor inteiro de Y é %.2lf e seu novo endereço é: %p\n", y, &y);

printf("o novo valor do ponteiro do real é %.2lf e seu novo endereço é: %p\n", \*py, &py);

printf("\no novo valor inteiro de C é %c e seu novo endereço é: %p\n", c, &c);

printf("o novo valor do ponteiro do caracter é %c e seu novo endereço é: %p\n", \*pc, &pc);

}

EXERCÍCIO NÚMERO 6

//a diferença entre eles é que no código 1, ele chama o endereço do ptr e no código 2, ele chama o valor do ptr

//no código 1 o valor impresso é ptr: 15, porque vai fazer 10=10+1, 11=11+1, 12=12+1, 13=13+1 e 14=14=1 que resultará em 15

//no código 2 o valor impresso é free(): invalid pointer, porque

EXERCÍCIO NÚMERO 7

#include <stdio.h>

int valores (int \*valor1, int \*valor2) {

int troca;

if (\*valor1 > \*valor2) {

printf("\na primeira variável é: %d e a segunda variável é: %d\n", \*valor1, \*valor2);

} else {

printf("\na primeira variável é: %d e a segunda variável é: %d\n", \*valor2, \*valor1);

troca = \*valor1;

\*valor1 = \*valor2;

\*valor2 = troca;

}

}

int main(void) {

int v1, v2;

printf("digite dois valores: \n");

scanf("%d\n %d", &v1, &v2);

valores (&v1, &v2); //& = tá enviando o endereço da variável 1 e da 2, aí tudo que fizer com elas vai acontecer na variável do main

printf("\nconteúdo da primeira variável: %d\n", v1);

printf("conteúdo da segunda variável: %d\n", v2);

return 0;

}

EXERCÍCIO NÚMERO 8

#include <stdio.h>

#include <stdio.h>

int ordem (int \*valor1,int \*valor2,int \*valor3) {

int troca1;

if(\*valor1 == \*valor2 && \*valor2 == \*valor3) {

return(1);

} else {

if ((\*valor1 < \*valor2 && \*valor1 < \*valor3)||(\*valor1 == \*valor2 && \*valor1 < \*valor3)||(\*valor1 == \*valor3 && \*valor1 < \*valor2)||(\*valor2 == \*valor3 && \*valor1 < \*valor2 && \*valor1 < \*valor3)) {

if (\*valor2 < \*valor3||\*valor2 == \*valor3) {

return(0);

} else if (\*valor3 < \*valor2) {

troca1 = \*valor3;

\*valor3 = \*valor2;

\*valor2 = troca1;

return(0);

}

} else if((\*valor2 < \*valor1 && \*valor2 < \*valor3)||(\*valor2 == \*valor1 && \*valor2 < \*valor3)||(\*valor2 == \*valor3 && \*valor2 < \*valor1)||(\*valor1 == \*valor3 && \*valor2 < \*valor3 && \*valor2 < \*valor1)) {

if(\*valor1 < \*valor3||\*valor1 == \*valor3) {

troca1 = \*valor1;

\*valor1 = \*valor2;

\*valor2 = troca1;

return(0);

} else if(\*valor3 < \*valor1) {

troca1 = \*valor1;

\*valor1 = \*valor2;

\*valor2 = \*valor3;

\*valor3 = troca1;

return(0);

}

} else if((\*valor3 < \*valor1 && \*valor3 < \*valor2)||(\*valor3 == \*valor1 && \*valor3 < \*valor2)||(\*valor3 == \*valor2 && \*valor3 < \*valor1)||(\*valor2 == \*valor1 && \*valor3 < \*valor1 && \*valor3 < \*valor2)) {

if(\*valor1 < \*valor2||\*valor1 == \*valor2) {

troca1 = \*valor1;

\*valor1 = \*valor3;

\*valor3 = \*valor2;

\*valor2 = troca1;

return(0);

} else if(\*valor2 < \*valor1) {

troca1 = \*valor1;

\*valor1 = \*valor3;

\*valor3 = troca1;

return(0);

}

}

}

return(0);

}

int main(void) {

int x,y,z;

printf("digite 3 números:\n");

scanf("%d %d %d", &x, &y, &z);

if (ordem(&x, &y, &z) ==1) {

printf("todos valores são iguais: %d, %d e %d\n", x, y, z);

} else if (ordem (&x, &y, &z) ==0) {

printf("a ordem crescente é %d, %d e %d\n", x, y, z);

}

}

//os printf não podem ficar no int do ponteiro, tem que ficar no do main, porque no exercício pede para "exibir os valores ordenados na tela NO PROGRMA PRINCIPAL", que é o main

//no int do ponteiro, nas coisas da troca, é o seguinte: se ta escrito por exemplo: troca1=\*valor1; \*valor1=\*valor2; \*valor2=\*valor3; \*valor3=troca1; significa que o valor valor2 vai ser escrito no lugar do valor1; que o valor3 vai estar escrito no lugar do valor 2 e que o valor da troca (valor1) vai estar escrito no lugar do valor3

EXERCÍCIO NÚMERO 9

#include <stdio.h>

void calcCircunferencia (float R, float \*compr, float \*area) {

float PI = 3.14;

\*compr = 2 \* PI \* R;

\*area = PI \* (R\*R);

}

int main(void) {

float r, c, a; //r= raio, c= comprimento, a= area

printf("digite o valor do raio: ");

scanf("%f", &r);

calcCircunferencia (r, &c, &a);

printf("o valor do comprimento é: %.2f\n", c);

printf("o valor da área é: %.2f\n", a);

return 0;

}

EXERCÍCIO NÚMERO 10

/\*#include <stdio.h>

void func(int \*px, int \*py) {

px=py;

\*py = (\*py) \* (\*px);

\*px = \*px + 2;

}

int main () {

int x, y;

scanf("%d",&x);

scanf("%d",&y);

func(&x,&y);

printf("x = %d, y = %d", x, y);

return 0;

} \*/

// Início do programa após scanf:

// Valores: x=5 , y=6 , \*px= 0x1a1 , \*py= 0x1a2

// Endereços: x=0x1a1 , y=0x1a2 , \*px=0x1a3 , \*py=0x1a4

// Depois de entrar no módulo e fazer px=py:

// Valores: x=5 , y=6 , \*px= 0x1a2 , \*py= 0x1a2

// Endereços: x=0x1a1 , y=0x1a2 , \*px=0x1a3 , \*py=0x1a4

// Após \*py = (\*py) \* (\*px)

// Valores: x=5 , y=36 , \*px= 0x1a2 , \*py= 0x1a2

// Endereços: x=0x1a1 , y=0x1a2 , \*px=0x1a3 , \*py=0x1a4

// Após \*px = \*px + 2

// Valores: x=5 , y=38 , \*px= 0x1a2 , \*py= 0x1a2

// Endereços: x=0x1a1 , y=0x1a2 , \*px=0x1a3 , \*py=0x1a4

// Fim do programa

// Valores: x=5 , y=38 , \*px= 0x1a2 , \*py= 0x1a2

// Endereços: x=0x1a1 , y=0x1a2 , \*px=0x1a3 , \*py=0x1a4

// Exibe: x = 5, y = 38