Universidade Federal do Piauí – UFPI

Campus Senador Helvídio Nunes de Barros – CSHNB Curso de Sistemas de Informação Bloco: IV Disciplina: Estruturas de Dados II

Professora: Juliana Oliveira de Carvalho

Acadêmico: Dayan Ramos Gomes Matrícula: 20209001285

ATIVIDADE DE FIXAÇÃO 1

1. void altera1(int \*\*p, int \*a)

{ \*\*p = \*a;

\*a = \*a +50;

}

void altera2(int \*\*p, int \*b)

{ \*p = b;

\*b = \*b +30;

}

int main(){

int x, y, \*px, \*py;

x = 10;

y = x + 20;

px = &x; py = &y;

printf("x = %d, End. x = %p, px = %p, y = %d \n”,x,&x, px,y); altera1(&px, &y);

printf("x = %d, End. x = %p, px = %p, y = %d \n”,x,&x, px,y); getchar();

printf("y = %d, End. y = %p, py = %p, x = %d \n”,y,&y, py,x); altera2(&py, &x);

printf("y = %d, End. y = %p, py = %p, x = %d \n”,y,&y, py,x); getchar();

return (0);

}

Dado o código acima, responda as seguintes questões:

* 1. Qual a diferença entre px e x?

**R = px guarda o endereço de memória de x, já a variável x, guarda um valor inteiro 10 inicialmente.**

* 1. Qual a diferença entre px e py?

**R = px guarda o endereço de memória de x inicialmente, já o py guarda o endereço de y.**

* 1. Quais são os valores impressos pelo primeiro printf?

**R = x = 10, End x = 0061FF1C, px = 0061FF1C, y = 30.**

* 1. O que muda do primeiro printf para o segundo?

**R = No segundo printf o valor de x exibido é 30, e o valor de y exibido é 80.**

* 1. Quais os valores impressos pelo terceiro print?

**R = y = 80, End y = 0061FF18, px = 0061FF18, x = 30.**

* 1. O que muda do terceiro para o quarto print?

**R = py em vez de mostrar o endereço de memória de y mostra o endereço de memória de x, e o valor de x exibido é 60.**

* 1. Explique a diferença entre o altera1 e o altera2.

**R = A função altera1 recebe um ponteiro pra ponteiro para a posição de x e a variável y passada por referência, após isso faz x receber o valor de y e após isso acrescenta mais 50 ao valor de y.**

**A função altera2 recebe um ponteiro para ponteiro para a posição de y e a variável x passada por referência, após isso faz o primeiro ponteiro passado apontar para a posição de x (endereço de memória de x) e depois acrescenta mais 30 ao valor de x.**

1. Faça o rastreamento dos códigos a seguir e diga o que os mesmos fazem, dizendo qual a diferença entre eles.

void misterio1(char b[TAM], float \*dec, int p, int i)

{

if(p < strlen(b))

{

if(b[p] == '1')

\*dec = \*dec + pow(2,i); misterio1(b,dec,++p, - -i);

}

}

float misterio2(char b[TAM], float dec, int p, int i)

{

if(p < strlen(b))

{ dec = misterio2(b,dec,p+1,i-1); if(b[p] == '1')

dec = dec + pow(2,i);

}

return(dec);

}

Obs.: A string b, contém 0s e 1s.

Obs.: Na chamada de misterio1 e de misterio2 dec e p devem ser 0 e i deve ser strlen(b) - 1. Obs.: a função pow, calcula a potência de 2 elevado a i.

Obs.: O número da posição 0(zero) de b, será a posição strlen(b) -1 do número binário.

Obs.: rastrear código significa, fazer uma simulação com valores mostrando o que acontece no código.

1. **No misterio1, começa verificando se o valor de p que inicialmente é zero é menor que o tamanho do vetor que é 2 (b[0] e b[1]), neste caso irá entrar no if e verificar se char da posição b[0], é igual ao char que ele procura que é ‘1’, como não vai ser, irá pular o if chamar a função novamente, só que agora passando o p++ e a variável --i que era o tamanho de b menos 1, na próxima chamada verifica novamente se o valor de p é menor que o vetor, após entrar no if verifica se a posição de b[1] é igual char ‘1’, nesse caso irá entrar no if e calcular a potencia de 2 elevado a i, que como foi passado –1, será 0, neste caso ira somar o resultado da potência com \*dec e guardar dentro do próprio \*dec, após isso irá chamar a função novamente, só que não ira entrar no primeiro if, assim a função irá modificar p valor de \*dec no fim, pois ele foi passado por referência.**
2. **No misterio2 faz a mesma coisa, só que nela a variável dec não é passada por referência, então quando é feita a chamada da função dentro de misterio2, a variável dec recebe o retorno da função para que assim no fim da recursão o valor de dec seja alterado.**
3. O código a seguir deveria devolver um vetor contendo a intersecção entre os dois vetores de entrada ordenados, o código possui erros, localize-os e diga como corrigi-los. Depois reescreva sem usar recursividade.

void inter(int V1[TAM], int V2[TAM], int V3[TAM],int i, int j, int q1, int q2, int q3)

{

if(i < q1 || j < q2)

{

if(V1[i] < V2[j])

**q3 =** inter(V1,V2,V3, i, j+1,q1,q2,q3); else if(V1[i] > V2[j])

inter(V1,V2,V3, i, j+1,q1,q2,q3); else {

V3[q3] = V1[j];

**q3 =** inter (V1, V2, V3, i+1, j+1, q1, q2, q3+1);

}

}

**return(q3);**

}

Obs.: Os valores iniciais de i, j e q3 é 0.

**R = Para o corrigir o código, primeiros retiramos o q3 recebendo a função inter quando chamada, pois a função inter é do tipo void, ou seja, não tem retorno, depois tiramos o return da função também pelo mesmo motivo.**

**FUNÇÃO SEM USAR RECURSIVIDADE:**

**void inter (int V1[TAM], int V2[TAM], int V3[TAM], int i, int j, int q1, int q2, int q3)**

**{**

**for (i = 0; i < q1; i++)**

**{**

**for (j = 0; j < q2; j++)**

**{**

**if (V1[i] == V2[j])**

**{**

**V3[q3] = V2[j];**

**q3++;**

**}**

**}**

**}**

**}**