Lista 1 – Lab. De Estrutura de Dados 1 e Estrutura de Dados 1

Aluno: Anderson Carlos da Silva Morais

Matrícula: 2024011327

Questões

1. O que é e para que serve um ponteiro?

   Ponteiro é um tipo de variável usada para armazenar endereços de outras variáveis, sendo ideal para trabalhar com estruturas dinâmicas e com grande quantidade de dados evitando cópias desnecessárias de dados.

2. Declare uma variável e “printe” o valor dela e o seu endereço.

```
    char nome[20] = "Anderson";
    char * ptr = nome; // um array de caracteres, e em C/C++, o
nome de um array já é tratado como um ponteiro para o primeiro
elemento do array.
    int idade = 23;
    int * ptr2 = &idade;

    cout << nome << endl;
    cout << (void*)ptr << endl;
    cout << idade << endl;
    cout << ptr2 << endl;
```

3. Qual é a maneira correta de referenciar ch, assumindo que o endereço de ch foi atribuído ao ponteiro indica?

   Dereferenciando o ponteiro indica

```
string txt = "dereferenciando o ponteiro: *indica \n";
    char ch = 'C';
    char * indica = &ch;

    // p -> ponteiro
    printf("%s", txt.c_str());
    printf("Endereço: %p\n", indica);
    printf("Valor: %c \n",*indica);
```

4. Na expressão float *ptr; o que é do tipo float?

   O tipo do dado apontador por ptr, ou seja, o tipo da variável armazenada no endereço no ponteiro.

5. Como seria o output se eu desse “print” nas variáveis a seguir:

int x=68, y;

int *p;

p = &x;

y = *p + 200;

p seria o endereço da variável x;

x seria o valor 68;

y seria o valor da somar de x com 200, logo 268.

6. Assumindo que queremos ler o valor de x, e o endereço de x foi atribuído a px, a instrução seguinte é correta? Por que?

scanf (“%d”, *px);

Não está correta, pois a função “scanf” espera receber um endereço para atribuir o valor informado pelo usuário, a forma como está descrita está sendo deferenciado o valor de px, o correto seria: scanf (“%d”, px);

7. Desenvolva uma função que receba como parâmetro os ponteiros de dois vetores de 5 posições. O procedimento deverá imprimir na tela os valores contidos nos dois vetores de forma crescente (Utilize ponteiros).

Exemplo:

Vetor 1 = 2, 5, 9, 8, 3

Vetor 2 = 7, 4, 1, 10, 6

Saída: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

```
void ordenar2(int * px1, int * px2){
    int vet[10];
    int size = sizeof(vet)/sizeof(vet[0]);
    int * ptr = vet;

    for (int i=0; i < 5; i++){
        *(ptr + i) = *(px1 + i);
        *(ptr + i + 5) = *(px2 + i);
```

```
    }

    int rep = size - 1;
    // Ordena o array vet com Bubble Sort (usando índices [])
    for (int i = 0; i < rep; i++) { // size - passagens (para 10
elementos): n - 1
        for (int j = 0; j < rep - i; j++) { // Percorre até o
penúltimo elemento não ordenado
            // ex. size = 9 | 9 - 0 -> 9 | 9 - 1 -> 8 | 9 - 2 -> 7
            // O " - i " é uma otimização: a cada passagem do loop
externo (i), o maior elemento já está no final, então não
precisamos comparar até o fim novamente.
            if (*(vet + j) > *(vet + j + 1)) { // Compara elementos
adjacentes
                // Troca os valores
                int temp = *(vet + j); // maior
                *(vet + j) = *(vet + j + 1); // menor(vet[j + 1])
seja o anterior
                *(vet + j + 1) = temp; // maior seja o adjacente
            }
        }
    }

    for (int i=0; i < size; i++){
        cout << *(vet + i) << " ";
    }
}

void ex7(){
    int vet1[5] = {2, 5, 9, 8, 3};
    int vet2[5] = {7, 4, 1, 10, 6};

    ordenar2(vet1, vet2);
}

int main(){
    setlocale(LC_ALL, "Portuguese");

    ex7();
}
```

8. Assumindo que o endereço da variável x foi atribuído a um ponteiro px, escreva uma expressão que não usa x e divida x por 3.

```
void ex8(){
    int x = 30;
    int * px = &x;
```

```
    printf("Dividindo o valor armazenado em x(%d) por 3: %d", x,
(*px / 3));
}
```

9. Seja a seguinte seqüência de instruções em um programa C:

int *pti;

int i = 10;

pti = &i;

Qual afirmativa é falsa? Justifique a resposta

I. pti armazena o endereço de i
II. *pti é igual a 10
III. Ao se executar *pti = 20; i passará a ter o valor 20
IV. Ao se alterar o valor de i, *pti será modificado
V. pti é igual a 10

I. VERDADEIRO: pti é definido como um ponteiro, apontando para o endereço da variável i;
II. VERDADEIRO: ao dereferenciar pti, acessamos o valor armazenado no ponteiro pti, que é o valor de i, logo, 10;
III. VERDADEIRO: ao usar “*pti” estamos acessando o valor do ponteiro que é i, e em seguida estamos atribuindo um novo valor, logo, i passará a ter o novo valor
IV. VERDADEIRO: se alterarmos o valor de i, ao acessar o valor do ponteiro por meio de *pti, será exibido o valor de i que foi alterado anteriormente.
V. FALSO: pti é o endereço da variável i, não o valor da variável.