

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO -  
UFERSA  
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA  
DISCIPLINA: Laboratório de Algoritmos e Estrutura de Dados I  
PROFESSOR: George Felipe Fernandes Vieira  
ALUNO: Enthony Araujo de Oliveira

## Atividade 1 - Laboratório de Algoritmos e Estrutura de Dados I

**Questão 1:** O que é e para que serve um ponteiro?

**Solução:**

Ponteiros são variáveis que armazenam endereços na memória, eles são utilizados para manipulação eficiente da memória.

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

**Solução:**

```
1 #include <stdio.h>
2
3 int main(){
4
5     int idade = 22;
6
7     printf("%d\n",idade);
8     printf("%p\n", &idade);
9
10    return 0;
11 }
```

Saída:

```
22
0x7ffddc37d204
```

Compilation successful

Figura 1: Questão 2

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

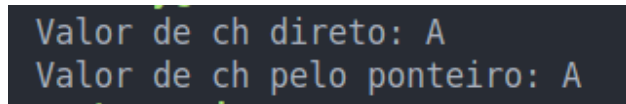
**Solução:**

```
1 #include <stdio.h>
2
```

```

3 int main(){
4
5     char ch = 'A';
6     char* indica;
7     indica = &ch;
8
9     printf("Valor de ch direto: %c\n", ch);
10    printf("Valor de ch pelo ponteiro: %c\n",*indica);
11
12    return 0;
13 }

```



```

Valor de ch direto: A
Valor de ch pelo ponteiro: A

```

Figura 2: Questão 3

**Questão 4:**Na expressão `float *ptr;` o que é do tipo `float`?

**Solução:** Na expressão o **float** não é uma variável, mas sim o valor para qual o ponteiro está apontando.

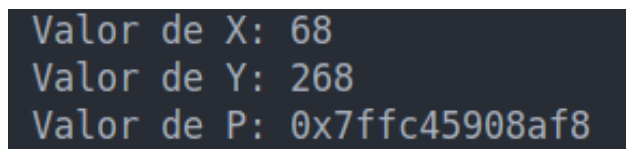
**Questão 5:**Como seria o output se eu desse “print” nas variáveis a seguir:

```

int x = 68, y;
int *p;
p = &x;
y = *p + 200;

```

**Solução:**



```

Valor de X: 68
Valor de Y: 268
Valor de P: 0x7ffc45908af8

```

Figura 3: Questão 5

**Questão 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 );
```

**Solução:** A instrução não está correta. A função **scanf** precisa receber o endereço da variável onde vai guardar o valor. Se usamos `&x`, passamos o endereço de `x`. Como `px` já guarda o endereço de `x`, podemos usar:

```
scanf("%d", px);
```

O erro está em usar `*px`, pois isso representa o conteúdo (valor de `x`), não seu endereço. Portanto, a instrução correta é:

```
scanf("%d", px);
```

**Questão 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).

**Solução:**

```
1 #include <stdio.h>
2
3 void imprime_vetor_crescente(int *v1, int *v2) {
4     int i, menor, maior;
5
6     menor = *v1;
7     maior = *v1;
8     for (i = 0; i < 5; i++) {
9         if (*(v1 + i) < menor) menor = *(v1 + i);
10        if (*(v1 + i) > maior) maior = *(v1 + i);
11        if (*(v2 + i) < menor) menor = *(v2 + i);
12        if (*(v2 + i) > maior) maior = *(v2 + i);
13    }
14
15    for (int val = menor; val <= maior; val++) {
16        for (i = 0; i < 5; i++) {
17            if (*(v1 + i) == val || *(v2 + i) == val) {
18                printf("%d ", val);
19            }
20        }
21    }
22 }
23
24 int main() {
25     int v1[5] = {2, 5, 9, 8, 3};
26     int v2[5] = {7, 4, 1, 10, 6};
27
28     printf("Saida: ");
29     imprime_vetor_crescente(v1, v2);
30
31     return 0;
32 }
```



Saida: 1 2 3 4 5 6 7 8 9 10

Figura 4: Questão 7

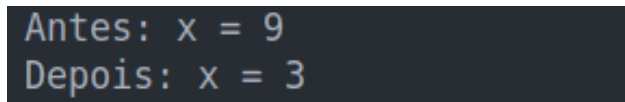
**Questão 8:** Assumindo que o endereço da variável `x` foi atribuído a um ponteiro `px`, escreva uma expressão que não usa `x` e divida `x` por 3.

**Solução:**

```

1 #include <stdio.h>
2
3 int main() {
4     int x = 9;
5     int *px = &x;
6
7     printf("Antes: x = %d\n", x);
8
9     *px = *px / 3;
10
11    printf("Depois: x = %d\n", x);
12
13    return 0;
14 }

```



```

Antes: x = 9
Depois: x = 3

```

Figura 5: Questão 8

**Questão 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

**Solução:** Alternativa falsa é a **V**. Pois, pti é um ponteiro que armazena o endereço de i, portanto, seu valor é um endereço de memória, não o valor 10. O valor 10 está armazenado em i, e pode ser acessado via \*pti, mas pti em si é um endereço.