**Ministério da Educação**
**Universidade Tecnológica Federal do Paraná**
**Câmpus Pato Branco**

Professora: Rúbia Eliza de Oliveira Schultz Ascari
Departamento Acadêmico de Informática (Dainf)
Tecnologia em Análise e Desenvolvimento de Sistemas
Estruturas de Dados 1

# Exercícios com Ponteiros

1. O código abaixo imprime os valores e endereços de um vetor e de uma string. Por que os endereços do vetor v aumentam de 4 em 4 e os endereços da string c aumentam de 1 em 1?

```c
#include <stdio.h>
#include <string.h>

void main(void) {
   int v[] = {1, 2, 3, 4, 5};
   char s[] = "ola";
   int i = 0;

   for(i = 0; i < 5; i++) {
       printf("valor: %d, end.: %ld\n", v[i], &v[i]);
   }

   printf("\n");
   for(i = 0; i < strlen(s); i++) {
       printf("valor: %c, end.: %ld\n", s[i], &s[i]);
   }
}
```

   R: Porque o tamanho inteiro ocupa 4 bytes, enquanto o char ocupa apenas 1.

2. Suponha que os elementos do vetor (v) são do tipo int e cada int ocupa 8 bytes no seu computador. Se o endereço de v[0] é 10000, qual o valor da expressão v + 5?
   R: 10040

3. (Exercício extraído do livro SCHILDT, Herbert, C Completo e Total, 3ª edição). Identifique qual é o problema do código:

```c
void main(void) {
    int x, *p;
    x = 10;
    *p = x;
}
```

   R: Está sendo atribuído valor para o conteúdo do ponteiro p, mas ele ainda não está apontando para nenhum endereço de memória.

4. (Exercício extraído do material do prof. Paulo Feofiloff - https://www.ime.usp.br/~pf/algoritmos/). Execute o programa abaixo e verifique sua saída:

```c
#include <stdio.h>

void func1(int x) {
   x = 9 * x;
}

void func2(int v[]) {
   v[0] = 9 * v[0];
}

void main(void) {
   int x, v[2];
```

```
   x = 111;
   v[0] = 111;

   func1(x);
   printf("x: %d\n", x);

   func2(v);
   printf("v[0]: %d\n", v[0]);
}
```

x e v[0] possuem valores iguais? Por que isso acontece?

R: Porque o vetor v é passado por referência (v[] é um ponteiro que guarda o primeiro endereço do vetor), então alterações feitas internamente na func2 tem seu efeito mantido posteriormente. Enquanto que no caso da func1, ela recebe uma cópia da variável x, e as alterações feitas internamente só tem efeito internamente.

5. Qual será a saída do programa?

```
#include <stdio.h>

int main(void){
   int v[] = {10, 20, 30};
   int *p;
   p = v;
   p++;
   printf("%d\n", *p);
   return 0;
}
```

Se em vez de:

```
p++;
```

fosse:

```
(*p)++;
```

Qual seria a saída?

R: 11, pois ao invés de usar a aritmética de ponteiros para avançar uma posição no vetor (v[1] = 20), é incrementado o valor da primeira posição do vetor, que antes era 10.

6. (Exercício extraído do material do prof. Paulo Feofiloff - https://www.ime.usp.br/~pf/algoritmos/). O que está acontecendo no código abaixo?

```
#include <stdio.h>
#include <string.h>

void imprime(char *s, int n) {
   char *c;
   for (c = s; c < s + n; c++)
      printf ("%c", *c);
}

void main(void){
    char s[] = "bom dia";
    imprime(s, strlen(s));
}
```

R: O ponteiro *c aponta para a variável s, e o laço de repetição faz a posição no vetor c avançar de um em um, enquanto imprime o conteúdo da variável como um único caractere (%c), apenas a primeira posição da string atual. Se fosse utilizado printf("%s", c), o texto todo seria impresso a cada laço, alterando apenas a posição de início da string.

7. O que o código abaixo vai imprimir? (Tente descobrir manualmente e depois execute o código para conferir).

```c
#include <stdio.h>

void main(void){
    int v[3], *p, *a;

    p = v;
    *p = 10;
    *(p + 1) = 20;
    *(p + 2) = 30;

    a = &v[1];
    *a = 40;
    *(a - 1) = 50;

    printf("%d\n", v[0]);
    printf("%d\n", v[1]);
    printf("%d\n", v[2]);
}
```

R: 50, 40, 30.

8. Corrija o problema do código abaixo:

```c
#include <stdio.h>

void troca(int *a, int *b) {
    int *t;
    *t = *a;
    *a = *b;
    *b = *t;
}

void main(void){
   int a = 10;
   int b = 20;
   printf("a: %d, b: %d\n", a, b);
   troca(&a, &b);
   printf("a: %d, b: %d\n", a, b);
}
```

R: variável t não precisa ser ponteiro.

```c
#include <stdio.h>

void troca(int *a, int *b) {
    int t;
    t = *a;
    *a = *b;
    *b = t;
}

int main(void){
   int a = 10;
   int b = 20;
   printf("a: %d, b: %d\n", a, b);
   troca(&a, &b);
   printf("a: %d, b: %d\n", a, b);

   return 0;
}
```

9. Escreva um programa para somar dois valores usando ponteiros.

```c
#include <stdio.h>

int soma(int *a, int *b) {
    return *a + *b;
```

```
}

int main(void){
    int a, b;
    printf("Informe dois valores numericos: ");
    scanf("%d %d", &a, &b);
    printf("Soma: %d\n", soma(&a, &b));

    return 0;
}
```

10. Escreva um programa que imprime um vetor. Percorra os itens do vetor através de aritmética de ponteiros.

```
#include <stdio.h>

int main(void){
    int vetor[] = {10, 20, 30, 40, 50};
    int *i;
    for (i = vetor; i < vetor + 5; i++)
        printf("i: %ld - valor: %d \n", i, *i);

    return 0;
}
```

11. Escreva uma função que imprime uma string. Percorra os itens da string através de aritmética de ponteiros.

```
#include <stdio.h>
#include <string.h>

int main(void){
    char texto[] = "Aula de Ponteiros";
    char *c;
    for (c = texto; *c != '\0'; c++)
        printf("%c", *c);

    return 0;
}
```

12. Escrever uma função para contar a quantidade de caracteres de uma string, usando ponteiros para percorre-la.

```
#include <stdio.h>
#include <string.h>

int conta(char *s) {
    char *c;
    int num = 0;
    for (c = s; *c != '\0'; c++)
        num++;
    return num;
}

int main(void){
    char texto[30];
    printf("Informe um texto: ");
    scanf("%[^\n]s", texto);
    printf("Quantidade de caracteres: %d", conta(texto));

    return 0;
}
```

13. Implemente uma função que copia o conteúdo de uma string de origem em uma string de destino. O protótipo da função deve ser void copia_string(char *destino, char *origem). Utilize ponteiros.

```
#include <stdio.h>
#include <string.h>

void copia(char *destino, char *origem) {
    char *c;
    *destino = '\0';
    for (c = origem; *c != '\0'; c++) {
        *destino = *c;
       if (*(c+1) != '\0')
          destino++;
    }
}

int main(void){
    char textoA[30];
    char textoB[30];
    printf("Informe um texto A: ");
    scanf("%[^\n]s", textoA);
    copia(textoB, textoA);
    printf("Valor copiado para o texto B: %s", textoB);
    return 0;
}
```

14. Implemente uma função que concatena uma string "b" no final de uma string "a". O protótipo da função deve ser void concatena_strings(char *sa, char *sb). Utilize ponteiros.

```
#include <stdio.h>
#include <string.h>

void concatena_strings(char *sa, char *sb) {
    char *c;
    int tam = strlen(sa);
    sa = sa + tam;
    for (c = sb; *c != '\0'; c++) {
        *sa = *c;
        if (*(c + 1) != '\0')
          sa++;
    }
}

int main(void){
    char textoA[30];
    char textoB[30];
    printf("Informe um texto A: ");
    scanf("%[^\n]s", textoA);
    printf("Informe um texto B: ");
    scanf(" %[^\n]s", textoB);
    concatena_strings(textoA, textoB);
    printf("Novo valor do texto A: %s", textoA);
    return 0;
}
```

15. Implemente uma função que compara duas strings. O protótipo da função deve ser: int compara_string(char *sa, char *sb), retornando um número inteiro para indicar se são iguais ou não. Utilize ponteiros.

```
#include <stdio.h>
#include <string.h>

int compara_string(char *sa, char *sb) {
    char *c;
    int iguais = 1;

    for (c = sb; *c != '\0'; c++) {
        if (*sa != *c) {
            iguais = 0;
            break;
```

```
        }
        if (*(c + 1) != '\0')
            sa++;
    }
    return iguais;
}

int main(void){
    char textoA[30];
    char textoB[30];
    printf("Informe um texto A: ");
    scanf("%[^\n]s", textoA);
    printf("Informe um texto B: ");
    scanf(" %[^\n]s", textoB);
    if (compara_string(textoA, textoB))
        printf("Textos iguais");
    else
        printf("Textos diferentes");
    return 0;
}
```