Universidade Estadual do Norte do Paraná
Bacharelado em Ciência da Computação
Centro de Ciências Tecnológicas

Lista VIII | Programação I | Ponteiros em C

Programação I

Prof. Maurício M. Arimoto / Prof. Paulo R. Anastácio

Aluno: Ana Julia Vieira Machado **Matrícula:** 202311113030017

Data: 08.11.2023

1. Suponha a declaração de um programa: $int\ vet[10]$, $*ptr$, $value$. Quais das expressões abaixo são válidas? Justifique sua resposta.

   a) $ptr = vet++$ Não é valida

   b) $vet[1] = ptr[3]$

   c) $ptr = vet + 1$ *ptr = &vet[0]; &vet[1]

   d) $value = (*ptr)++$

2. Suponha a declaração de um vetor: $int\ vet[50]$. Qual expressão abaixo referência o quinto elemento do vetor?

   ,-> elemento 0 1 2 3 4 <- 5to

   a) $*(vet + 4)$ int vet[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

   b) $*(vet + 5)$

   c) $vet + 4$ Endereço do 5to elemento !

   d) $vet + 5$

   e) nenhuma das anteriores

3. Analise a sequência de instruções a seguir:

```
1 int x = 10, y = 5;
2 int *ptr1 = &x;
3 int *ptr2 = &y;
```

   Quais expressões abaixo são válidas e quais não são válidas? Justifique sua resposta.

   .-> A saída seria 0 (falso)

   (a) y = ptr1 == ptr2; É uma expressão valida, mas falsa, pois apontam pra regiões diferentes

   (b) ptr1 += ptr2; Invalida, não é possível somar dois endereços diretamente

   .-> caso seja utilizado em sequencia de a, em que y = 0, logo o valor de x se manteria 10

   (c) x = (*ptr1) - (*ptr2); Valida, pois subtrai o conteúdo dos endereços

   (e) x = ptr1 || ptr2; Valido, pois ambos (ptr1 e ptr2) apontam para endereços validos (!= NULL)

   (e) y = (*ptr2)++; Valida, está alterando o valor de y

4. Reescreva o programa abaixo usando ponteiros.

```c
#include <stdio.h>
#define MAX 255

int main() {
    char str[MAX], caractere;
    int count = 0;

    printf("Entre com a string: ");
    fgets(str, MAX, stdin);
    printf("Entre com o caractere: ");
    scanf(" %c", &caractere);

    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == caractere) {
            printf("%d\n", i);
            count++;
        }
    }
    if (count == 0)
        printf("-1\n");
    return 0;
}
```

```c
#include <stdio.h>
#define MAX 255

int main (){
 char str[MAX] , caractere ;
 int count = 0 ;

 printf( "Entre com a string: " ) ;
 fgets(str, MAX, stdin) ;
 printf("Entre com o caractere : " ) ;
 scanf(" %c" , &caractere) ;

  char *p = str;

 for (int i = 0;  *(p + i) != '\0'; ++ i ) {
   if (  *(p + i) == caractere ){
     printf ("%d ", i) ;
     count++;
   }
 }
  if ( count == 0 )
    printf( " -1\n" );

return 0;
}
```

5. Indique quais as saídas produzidas pelo programa a seguir. Faça o teste de mesa de cada instrução e verifique os resultados. Depois, você pode executar o código comparando os resultados.

```c
#include <stdio.h>

int main() {
    int x, y = 27;
    int *pt1 = &x;
    int *pt2 = &y;
    int **ppt = &pt1;
    **ppt = *pt2;
    (*pt2)++;  y = 28
    x--;  x = 26
    printf("%d %d\n", *pt1, *pt2);  saída =  26 (x) , 28 (y)
    (**ppt) += --(*pt2);  x = x + (y-1) = 53
    printf ("%d\n", **ppt);  Saída = 53
    printf ("%d %d\n", x, y); Saída = 53, 27
    printf ("%d\n", pt1 == &y); Saída = 0 --> Falso
    printf("%d\n", &x != pt2); Saída = 1 --> Verdadeiro
    return 0;
}
```

*pt1 -> x = 27 26

*pt2 -> y = 27 28

*ppt -> *pt1 -> x = pt2 = 27

6. Indique quais as saídas produzidas pelo programa a seguir. Faça o teste de mesa de cada instrução e verifique os resultados. Depois, você pode executar o código comparando os resultados.

```c
#include <stdio.h>

int main() {
    int vet1[] = {1, 2, 3, 4, 5, 6, 7};
    int vet2[] = {7, 6, 5, 4, 3, 2, 1};
    int *ptr1 = vet1;
    int *ptr2 = vet1 + 3;
    int *ptr3 = vet2 + 5;
    (*ptr1)++;
    (*ptr2)++;
    (*ptr3)--;
    printf("vet1[0]: %d, vet1[3]: %d\n", vet1[0], vet1[3]);
    printf("vet2[0]: %d, vet2[5]: %d\n", vet2[0], vet2[5]);
    return 0;
}
```

*ptr1 -> vet1[0] = 1

*ptr2 -> vet1[3] = 4

*ptr3 -> vet2[5] = 2

vet1[0] = 2

vet1[3] = 5

vet2[5] = 1

Saída = 2, 5

Saída = 7, 1

7. Faça um programa que receba um vetor de 20 elementos inteiros, em seguida, percorra o vetor através do ponteiro `ptr-inicio` – a partir do início do vetor e outro ponteiro `ptr-fim` a partir do final do vetor, até os dois ponteiros se encontrarem no meio do vetor.

8. Utilizando aritmética de ponteiros, mostre como exibir a frase "não gosto de programar em C" como "gosto de programar em C".

9. Implemente uma função que receba um vetor de inteiros, o tamanho do vetor e um inteiro *pos* passado por referência. A função retorna o maior elemento do vetor e, na variável *pos*, a posição do maior elemento do vetor.

10. Faça um programa que leia uma matriz quadrada de ordem `4 X 4` de números inteiros. Depois, leia um número `x` e verifique quantas vezes `x` aparece na matriz.

11. Faça um programa que leia uma string de no máximo 100 caracteres. Em seguida, implemente uma função para calcular e mostrar o total de palavras da string lida. Para isso, utilize o protótipo de função a seguir.

```c
int totalPalavras(char *str) {

}
```