Ponteiro

Um ponteiro é uma variável que contém um endereço de memória

Forma geral de declarar uma variável ponteiro
*Tipo *nome*

Existem dois operadores especiais para ponteiro

- & : Devolve o endereço de memória do seu operando
- *: Devolve o valor da variável localizada no endereço que o segue.

Exemplo prático 1 : Elabore o programa e observe o resultado das variáveis "x" e "*pt".
Quais são os valores exibidos em tela antes e depois da atribuição feita em *pt ? Explique sua resposta

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

float x = 10.2;
float *pt;
main()
{
  pt = &x;    -→ pt recebe o endereço de memória de x

  printf("%f\n",x); -→ mostra o valor de x
  printf("%f\n",*pt); -→ mostra o valor contido no endereço de memória
                                referenciado pelo ponteiro pt
  *pt = 11.4;  -→ Altera o conteúdo da memória indicada pelo ponteiro pt

  printf("%f\n",x);
  printf("%f\n",*pt);

  system("pause");
}
```

O que acontece se acrescentarmos os comandos abaixo ?

```
  printf("%d\n",&x);
  printf("%d\n",pt);
```

Exemplo prático 2 : Observe que o conteúdo das variáveis se alterou mas não houve alteração no endereço de memória apresentado nos dois "prints". Por que ?

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

int cont = 10;
int *pt;
main()
{
  pt = &cont;

  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  *pt = 12;

  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);
  system("pause");

}
```

O que acontece se alterarmos o comando do printf para o comando abaixo ?

```
  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, &pt);
```

explique a sua análise.

Exemplo prático 3 : Observe que :

- 1^0 printf - pt não aponta para nenhum endereço de memória. Por que isso gera erro ? Como eliminar o problema ?
- Porque o conteúdo exibido em cont e *pt são iguais ? (Segundo comando printf)
- Os conteúdos de "num, cont e *pt" são iguais ou diferentes ?
- Porque o ultimo endereço de memória impresso em tela é diferente dos demais ?

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

int cont = 10;
int *pt;
int num = 0;
main()
{
  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  pt = &cont;

  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  *pt = 12;
  num = *pt;

  printf("cont = %d;  pt = %d; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);
  printf("num = %d no endereco = %d \n\n\n", num, &num);
  system("pause");
}
```

Exemplo prático 4 :
O que você consegue observar na impressão depois dos comandos “pt++” e “pt--” ?

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

int cont = 10;
int *pt;
main()
{
  pt = &cont;

  printf("cont = %d;  pt = %d ; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  *pt = 12;
  printf("cont = %d;  pt = %d ; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  pt++;
  printf("cont = %d;  pt = %d ; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);

  pt--;
  printf("cont = %d;  pt = %d ; endereco de (cont,pt) = (%d,%d) \n", cont, *pt, &cont, pt);
  system("pause");
}
```

Exemplo prático 5 :
Os dois laços de repetição exibem o mesmo resultado ? Qual é a diferença entre eles ? Explique

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

main()
{
  int V[5]={10,20,30,40,50};
  int *p;
  int i;

  p = V;

  for (i=0;i<5;i++)
    printf("%d ",V[i]);

  printf("\n");

  for (i=0;i<5;i++)
    printf("%d ",*(p++));

    system("pause");
}
```

Exemplo prático 6 :
Qual é o objetivo do comando malloc ?

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

main()
{
  int *p;
  int i;

  printf("\n aumentando o numero de elementos ");
  p = (int *)malloc(6*sizeof(int));
  *(p+0) = 10;
  *(p+1) = 20;
  *(p+2) = 30;
  *(p+3) = 40;
  *(p+4) = 50;
  *(p+5) = 60;
  for (i=0;i<6;i++)
    printf("%d ",*(p++));


  printf("\n aumentando o numero de elementos ");
  p = (int *)malloc(7*sizeof(int));
  *(p+0) = 10;
  *(p+1) = 20;
  *(p+2) = 30;
  *(p+3) = 40;
  *(p+4) = 50;
  *(p+5) = 60;
  *(p+6) = 70;
  for (i=0;i<7;i++)
    printf("%d ",*(p++));

    system("pause");
}
```

Exemplo prático 7 :
No exemplo abaixo foi criado algum vetor ?
Qual é a diferença entre ponteiro e vetor ?

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

main()
{
  int *p;
  int i;
  int qtde;

  printf("Deseja cadastrar quantos elementos ? \n");
  scanf("%d",&qtde);

  p = (int *)malloc( qtde * sizeof(int));

  printf("cadastrando os elementos \n" );
  for (i = 0; i < qtde ; i++)
     scanf("%d",&p[i]);

  printf(" \n imprimindo os elementos \n" );
  for (i = 0; i < qtde; i++)
    printf("%d ",p[i]);


    system("pause");
}
```

Exemplo prático 8 :
Qual é a diferença entre os comandos “malloc” e “realloc” ?

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

main()
{
  int *p;
  int i;
  int qtde;

  printf("Deseja cadastrar quantos elementos ? \n");
  scanf("%d",&qtde);

  p = (int *)malloc( qtde * sizeof(int));

  printf("cadastrando os elementos \n" );
  for (i = 0; i < qtde ; i++)
     scanf("%d",&p[i]);

  printf(" \n imprimindo os elementos \n" );
  for (i = 0; i < qtde; i++)
    printf("%d ",p[i]);


  printf(" \n Adicionando um novo elemento = 999 \n" );
  qtde++;
  p = (int *)realloc( p, qtde * sizeof(int));
  p[qtde-1] = 999;



  printf(" \n imprimindo os elementos \n" );
  for (i = 0; i < qtde; i++)
    printf("%d ",p[i]);


  system("pause");
}
```