2 - USANDO PONTEIROS

- A linguagem C é totalmente dependente de ponteiros.
- Quando falamos de ponteiros, estamos falando em fazer manipulação la no endereço de memoria para onde o valor da variavel foi alocado.
- Trabalhando com endereço de memoria, estamos trabalhando a baixo nivel na linguagem. Estamos fazendo acesso ao hardware logico da memoria.

- Vamos criar um progama.

```
int main(){{\big|//inicio_main} \\
    printf("\n"); \\
    int n; \\
    printf("informe um numero:\n"); \\
    scanf("%d",&n); \\
    printf("0 numero informado foi : %d\n",n); \\
    printf("Endereço de memoria: %x/n",&n); \end{array}
```

```
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S10(ponteiros)\2-USANDO_PONTEIROS

> cmd /c .\"p22.exe"

informe um numero:
6
0 numero informado foi : 6
Endereco de memoria: 61ff1c

PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S10(ponteiros)\2-USANDO_PONTEIROS

> cmd /c .\"p22.exe"

informe um numero:
3
0 numero informado foi : 3
Endereco de memoria: 61ff1c

PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S10(ponteiros)\2-USANDO_PONTEIROS

> Tendereco de memoria: 61ff1c
```

- Alguns sistemas como o windows usando o mesmo espaço alocado, outros reservam um a cada execução do progama.

- inicializando um ponteiro

```
int n; //ddeclarando_variavel - variavel que guarda seu valor na memoria
int *p; //declarando_variavel(ponteiro) - variavel que guarda um endereço de memoria.
```

- O ponteiro recebe um valor de uma variavel quando passamos o endereço dessa variavel para ele:

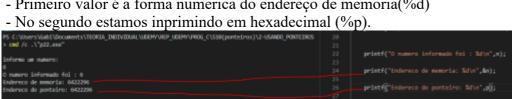
```
p = &n; //ponteiro recebe valor informado pelo usuario
```

- O local que o SO alocar para a variavel n deve ser informado para a variavel ponteiro.

```
printf("Endereco do ponteiro: %p\n",p); You, a few
```

informe um numero: 6 O numero informado foi : 6 Endereco de memoria: 6422296 Endereco do ponteiro: 0061FF18

- Primeiro valor é a forma numerica do endereço de memoria(%d)



- Os dois são iguais!