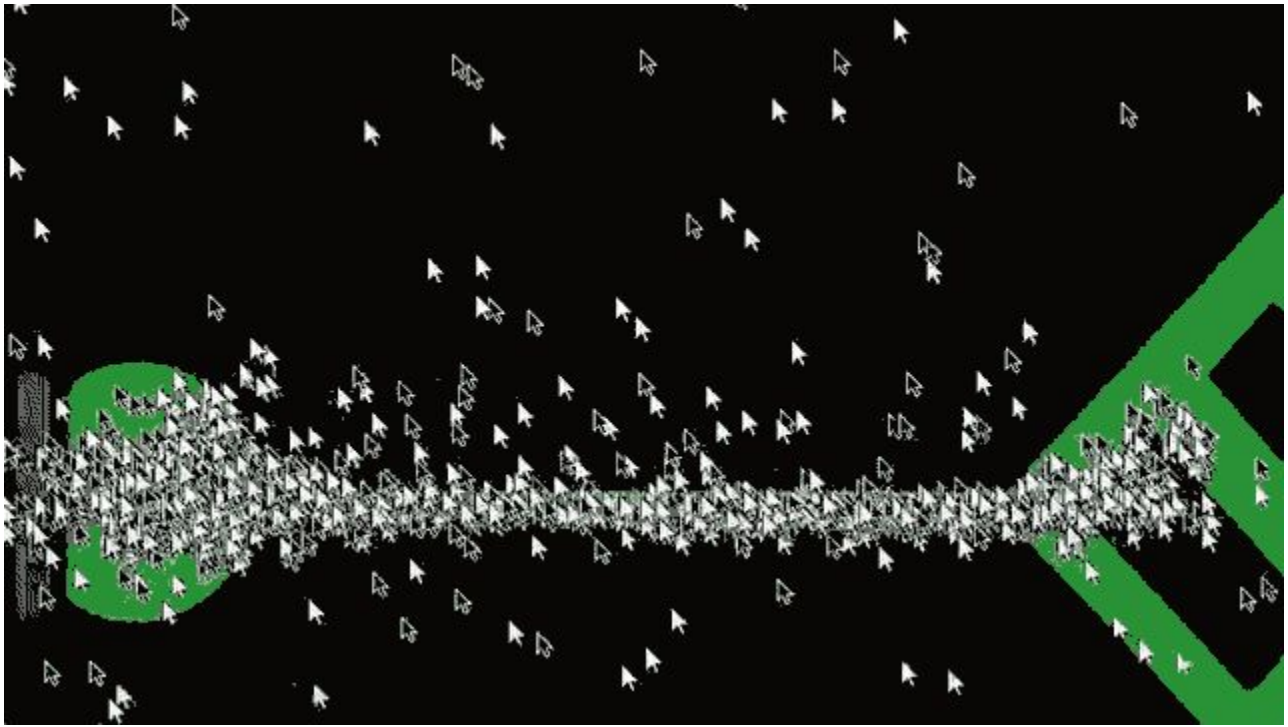
Aula 14 - Ponteiros

Sérgio Malandro

Roteiro da Aula

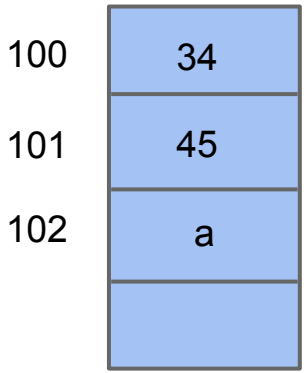
- Objetivos: conhecer o conceito de ponteiros e acesso a memória
- Fluxo:
  - Conceito
  - Declaração
  - Derreferenciamento
  - Aritmética de Ponteiros
  - Passagem de parâmetros por Referência

Às vezes, precisamos é apontar!



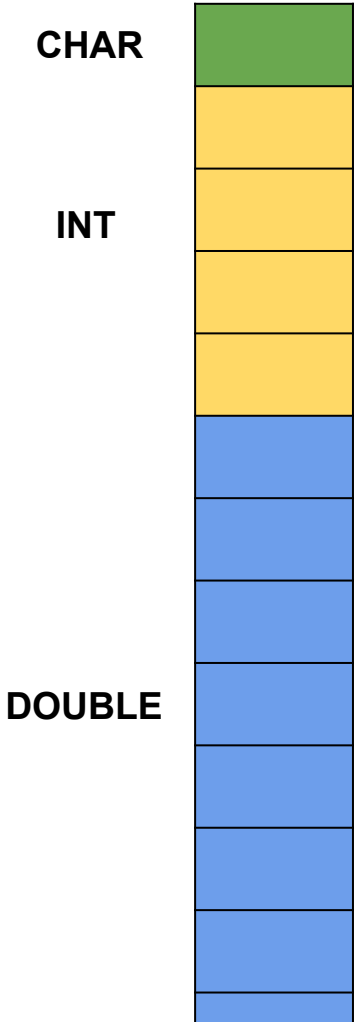
Memória e Variáveis

- Toda variável corresponde a um pedaço de memória
  - Guarda um valor
  - Tem um endereço



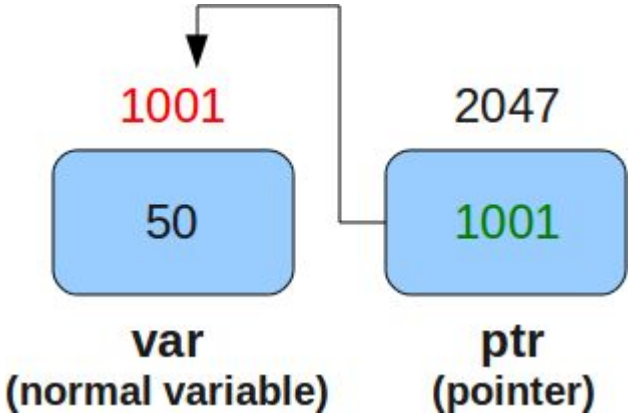
Memória e Variáveis

- Cada tipo de variável possui um tamanho diferente!



Ponteiros!

- Tipos especiais de variáveis, que guardam o valor de um endereço na memória
  - Apontam para uma variável, por exemplo



Então...qual é a ideia?

- Um ponteiro em C é definido na seguinte forma:
  - `tipo de dado * nome_ponteiro;`

Ex:

```
int* ptr;
```

Cuidado com a inicialização!

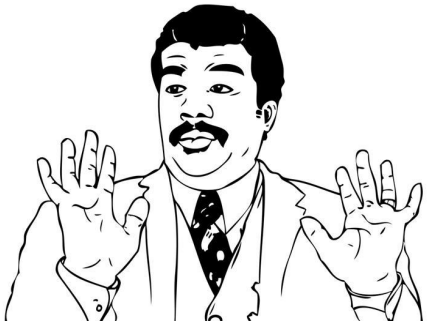
- Variáveis não inicializadas contém valores lixo.

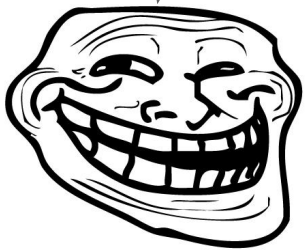
- Ponteiros não inicializados iniciam o apocalipse.

Imagine o seguinte....

int *ptr; (aponta para??)

*ptr = 5;





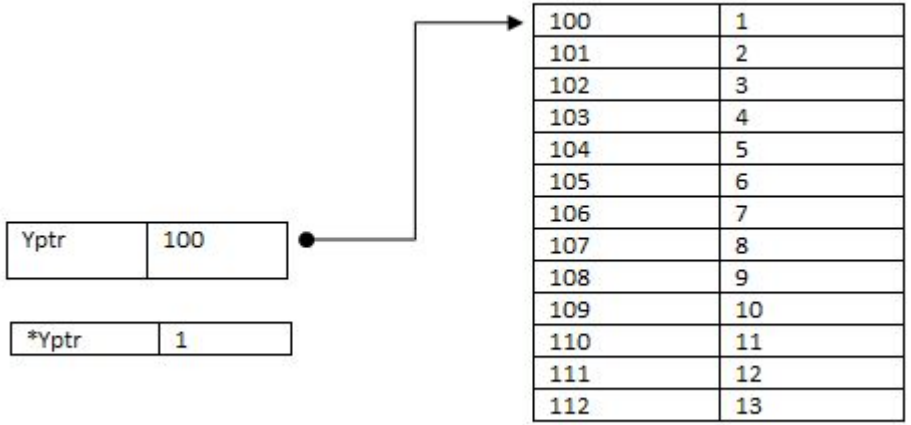
Lembra do operador &?

- Para pegar o endereço de memória de uma variável, usamos o operador & (sim, o mesmo do scanf!!!).

```
int a;
int *ptr_a;
ptr_a = &a;
```

Referenciando e Derreferenciando

- Quando temos um ponteiro ptr:
  - ptr é o endereço de memória;
  - *ptr é o valor que está no endereço;



Treinando

```
int main( )
{
   int i = 2;
   int j = i * i;
   int *k = &i;
   int m = *k * *k;
   *k = j * *k * m;

   printf( "%i %i %i %i", i, j, *k, m );
   return 0;
}
```

Treinando

```
int main( void )
{
   int  x = 5;
   int *y = &x;
   int z = *y;
   printf( "%i, %i, %i\n", x, *y, z );

   x = 7;
   printf( "%i, %i, %i\n", x, *y, z );

   *y = 2;
   printf( "%i, %i, %i\n", x, *y, z );
   return 0;
}
```

Treinando

```
int main( void )
{
   int x, y = 0, *p = NULL;
   p = &y;
   x = *p;
   x = 4;
   (*p)++;
   --x;
   (*p) += x;

   printf( "%i %i %i", x, y, *p );
   return 0;
}
```

Aritmética de ponteiros

- O que acontece quando eu faço uma operação com um ponteiro?

Ex:

```
ptr++;
ptr += 5;
```

Aritmética de Ponteiros

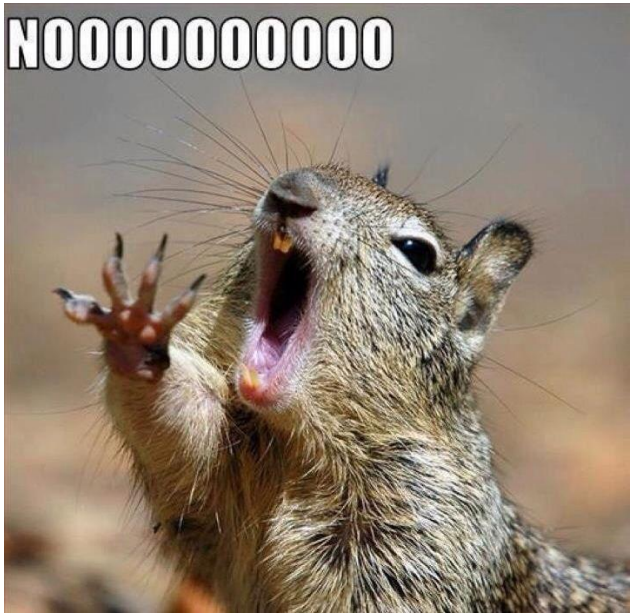
- Aplicação:
  - Estruturas de dados mutidimensionais
    - Vetores, Matrizes
    - Registros
  - Transferência de Dados

Vetores são ponteiros disfarçados!

- Os vetores são armazenados de forma sequencial na memória.
- Internamente, o C armazena um ponteiro para o início do vetor.
- Quando fazemos vetor[5], o C faz ptr_vetor+5!

Passagem de valor por referência

- Com ponteiros nós passamos o endereço da variável! Ou seja, ele consegue acessar a variável de outro escopo.



Treinando

```
void misterio( int *p, int *q ){
   int r = *p;
   *p = *q;
   *q = r;
}
```

```
int main ( ){
   int i = 1;
   int j = 2;
   misterio( &i, &j );
   printf( "%i %i\n", i, j );
   return 0;
}
```

Ok...isso serve para quê?

Uma ruma de coisa!

- **Passagem de estruturas multidimensionais para funções**
- **Estruturas de Dados**
- **Referências para funções (wtf???)**

Passando vetores e matrizes para funções

- Porque não é nenhum bobinho, C passa os arrays para as funções por referência! Ele faz isso porque é mais rápido :)
- Como é uma passagem por referência, temos que:
  - A função altera diretamente os valores do array (o que você faz na função vai alterar valendo!)
  - A função vai receber um ponteiro para o início do array, ela NÃO vai saber qual é o tamanho do array por padrão (apenas onde ele começa)

Formas de passar - Vetores

- Forma 1: como um ponteiro
  - `int aleluia(int* vetor){ bla; bla; bla; return milagre}`
- Forma 2: como um vetor com “tamanho”
  - `int aleluia(int vetor[10]){ bla; bla; bla; return milagre}`
- Forma 3: como um vetor sem “tamanho”
  - `int aleluia(int vetor[ ]){ bla; bla; bla; return milagre}`

Forma de passar - Matrizes

- Forma 1 - Como ponteiros
  - **int aleluia(int** matriz){ bla; bla; bla; return milagre}**
- Forma 2 - Com “tamanho” definido
  - **int aleluia(int matriz[10][10]){ bla; bla; bla; return milagre}**
- Forma 3 - Sem “tamanho” definido (apenas a primeira dimensão pode ficar vazia, as outras terão que ter algum valor associado)
  - **int aleluia(int matriz[ ][10]){ bla; bla; bla; return milagre}**

Dúvidas