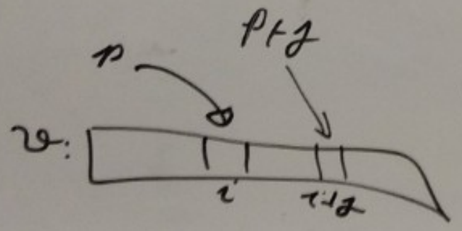
UNIVERSIDADE FEDERAL DO CEARÁ

CK0109 - 2019.2 - T02 - ESTRUTURAS DE DADOS

AULA 04 - 2019-08-19

PONTEIROS (CONT.)

1. CONVERSÃO VETOR-PONTEIRO: UMA EXPRESSÃO DE TIPO "VETOR DE (ELEMENTOS DE TIPO) T" PODE SER IMPLICITAMENTE CONVERTIDA PARA O TIPO "PONTEIRO PARA T". INFORMALMENTE, TAL CONVERSÃO ACONTECERÁ QUANDO A EXPRESSÃO FOR USADA NUM CONTEXTO EM QUE UM PONTEIRO, MAS NÃO UM VETOR, É ESPERADO. O PONTEIRO RESULTANTE APONTA PARA O 1º ELEMENTO DO VETOR.



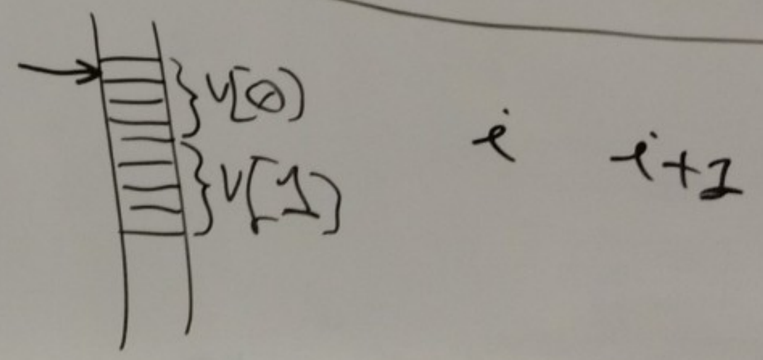
$0 \leq i+j \leq n$

EXEMPLO:

```
INT v[5];
INT *PRIMEIRO = v; //O MESMO QUE &v[0].
INT *ULTIMO = v+4; //O MESMO QUE &v[4].
/*OBSERVE QUE "v" CONTINUA SENDO UM VETOR
  DE INTEIROS; APENAS AS OCORRÊNCIAS DE "v"
  NAS 2 LINHAS ANTERIORES SÃO EXPRESSÕES QUE
  SOFREM CONVERSÃO IMPLÍCITA. */
```

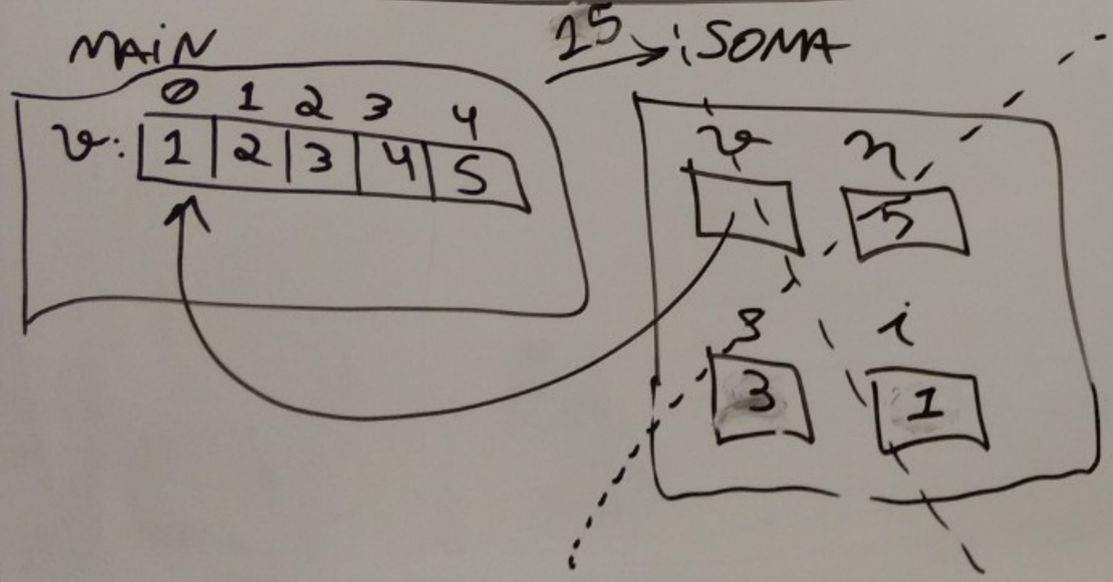
2. DEFINIÇÃO DO OPERADOR []: SE p É UM PONTEIRO E i É UM INTEIRO, ENTÃO

$p[i] \overset{\text{def.}}{=\!=} *(p+i)$.



EXEMPLO:

```
#INCLUDE <IOSTREAM>
DOUBLE SOMA (DOUBLE *V, INT N)
  {
   DOUBLE S=0; INT I;
   FOR(I=0; I<N; ++I) S += V[I];
   RETURN S;
  }
INT MAIN ()
  {
   DOUBLE V[5] = {1,2,3,4,5};
   STD::COUT << SOMA(V,5) << '\N';
  }
```



3. PONTEIRO NULO: UM PONTEIRO PODE ARMAZENAR O VALOR ZERO; NESSE CASO, É GARANTIDO QUE ELE NÃO ESTARÁ APONTANDO PARA QUALQUER VARIÁVEL OU SEMELHANTE OBJETO DO PROGRAMA, POIS NENHUM OBJETO ESTARÁ ARMAZENADO NESTE ENDEREÇO. O VALOR ZERO É, PORTANTO, UM VALOR ESPECIAL PARA UM PONTEIRO, E COMO TAL É UTILIZADO PARA INDICAR CIRCUNSTÂNCIAS ESPECIAIS (UM EXEMPLO TÍPICO SERÁ APRESENTADO A SEGUIR).

EMBORA SEJA COMPLETAMENTE VÁLIDO ATRIBUIR E COMPARAR PONTEIROS E A CONSTANTE INTEIRA "0", É PREFERÍVEL UTILIZAR A CONSTANTE NULLPTR, QUE SERVE ESPECIFICAMENTE PARA REPRESENTAR UM PONTEIRO NULO, E NÃO SE CONVERTE IMPLICITAMENTE PARA INTEIRO.

EXEMPLO:

```
INT i;
if (&i == NULLPTR) COUT << "ABSURDO!\n";
INT *P = NULLPTR; // OU "... = 0;"
if(P != &i) COUT << "SEMPRE TRUE!\n";
```

ALOCAÇÃO DINÂMICA

4. DURAÇÃO DE ARMAZENAMENTO: É A INFORMAÇÃO DA DURAÇÃO MÍNIMA DE UM OBJETO NA MEMÓRIA DE UM PROGRAMA EM C++. OS QUATRO TIPOS SÃO:

a) ESTÁTICA: O OBJETO DURA POR TODA A EXECUÇÃO DO PROGRAMA.

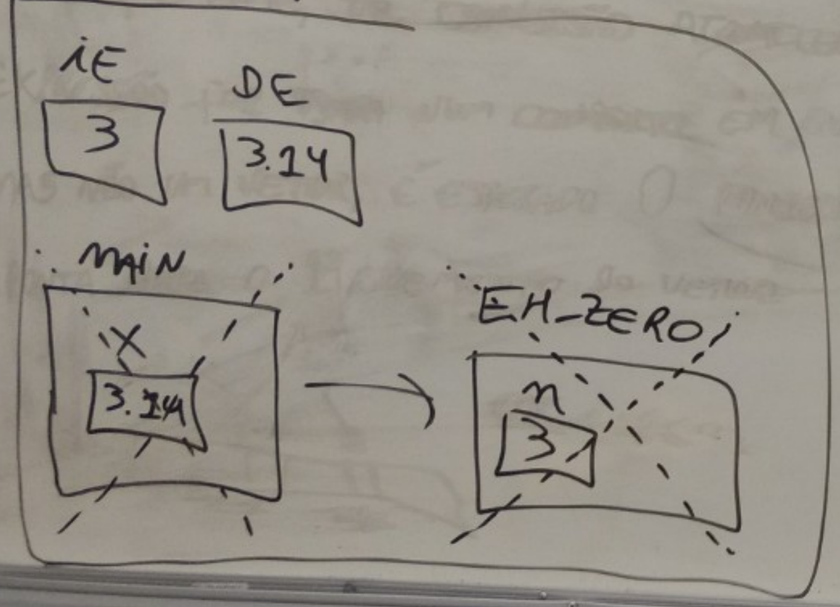
b) "THREAD": O OBJETO DURA ENQUANTO A "THREAD" A QUE ELE PERTENCE ESTIVER SENDO EXECUTADA.

c) AUTOMÁTICA: O OBJETO DURA ATÉ O FLUXO DA EXECUÇÃO DO PROGRAMA SAIR DO BLOCO EM QUE O OBJETO FOI CRIADO (O OBJETO É CRIADO APÓS O FLUXO ENTRAR NO BLOCO).

d) DINÂMICA: TANTO A "CRIAÇÃO" (RESERVA DA MEMÓRIA DO) OBJETO QUANTO A "DESTRUIÇÃO" (DESASSOCIAÇÃO DA MEMÓRIA AO PROGRAMA) DO MESMO ACONTECEM EM MOMENTOS DETERMINADOS PELO PROGRAMADOR.

Exemplo:

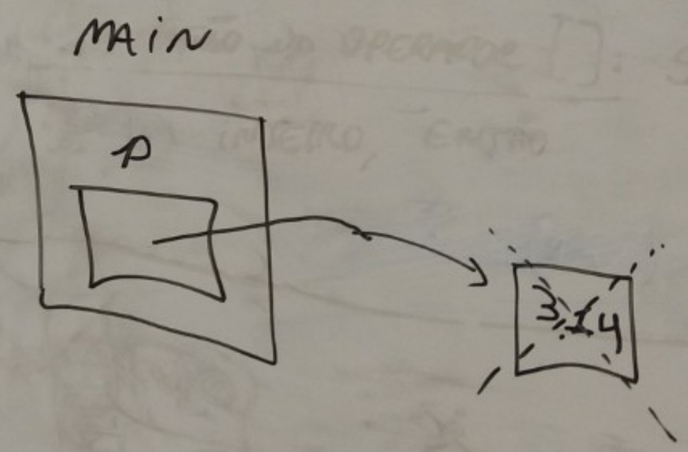
```
#include <iostream>
int ie = 3;  double de = 3.14;  // var. estática!
bool eh_zero (int n) { return (n==0); }
int main ()
  {
    double x = de;
    if(eh_zero(ie)) cout << "Nunca ocorre...\n";
  }
```



5. Alocação Dinâmica:

a) Não-vetores:

```
int main ()
  {
    double *p = new double; //teste omitido...
    *p = 3.14;
    delete p;
  }
```



b) VETORES:

```
#INCLUDE <IOSTREAM>
USING STD::CIN; USING STD::COUT;

DOUBLE SOMA (DOUBLE *v, INT n)
  { ... COMO ANTES... }

DOUBLE* LER_VETOR (INT n)
  {
   DOUBLE *v = NEW DOUBLE[n]; //TESTE OMIT...
   INT i;
   FOR(i=0; i<n; ++i)
    {
     COUT << "v[" << i << "]: ";
     CIN >> v[i];
    }
  RETURN v;
}
```

```
INT MAIN ()
  {
   INT n = 5;
   DOUBLE *v = LER_VETOR(n);
   COUT << "SOMA: " << SOMA(v,n) << '\n';
   DELETE[] v;
  }
```