NOTAS DE CLASE CECS 2202
Por: Prof. Blanca Tallaj
SOBRE ARREGLOS Y PUNTEROS …

EXPRESIONES CON PUNTEROS Y ARITMETICA DE PUNTEROS

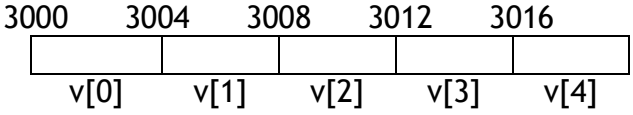
Los punteros son operandos válidos en las siguientes operaciones:

1. Expresiones aritméticas
2. Expresiones de asignación
3. Expresiones de comparación

Las ***Operaciones aritméticas*** con punteros tienen significado si se realizan en un arreglo. Podemos usar las siguientes operaciones aritméticas con punteros (mas abajo se explica el significado de estas operaciones):

- Incremento (++)
- Decremento (--)
- Suma de un numero entero a un puntero (+, +=)
- Resta de un numero entero a un puntero (-, -=)
- Resta entre dos punteros

Supongamos que se tiene un arreglo definido como: int v[5];



Un puntero definido como: **int *vPtr**;, se puede iniciar para que apunte al arreglo v; es decir, para que guarde la direccion de memoria del primer elemento del arreglo v, con cualquiera de las siguientes instrucciones:

```
vPtr = v;      //vPtr apunta al primer elemento del arreglo v (localización 3000)
vPtr = &v[0]; //vPtr apunta al primer elemento del arreglo v (localización 3000)
```

Si se le suma 2 al puntero, el resultado de la suma NO es 3002. Cuando se trabaja con punteros, sumarle dos es saltar a dos elementos mas adelante en el arreglo v, como sigue:

```
vPtr += 2;  //Ahora el puntero no apunta a v[0], sino a v[2] (localizacion 3008)
```

Si ahora al puntero se le resta un 1, se mueve a apuntar a v[1] (localizacion 3004)

```
vPtr--;
```

La resta entre dos punteros nos da el numero de elementos del arreglo entre un puntero y el otro; así, x contiene el valor 2 después de ejecutarse la siguiente

operación (suponga que v2Ptr contiene la localización 3008 y vPtr contiene la localización 3000 al ejecutarse esta operación):

```
x = v2Ptr – vPtr;  //El resultado es 2, el número de elementos en el arreglo
                      entre un puntero y el otro
```

```
// Fig. 5.20: fig05_20.cpp
// Using subscripting and pointer notations with arrays

#include <iostream>

using std::cout;
using std::endl;

int main()
{
   int b[] = { 10, 20, 30, 40 }, i, offset;
   int *bPtr = b;    // set bPtr to point to array b

   cout << "Array b printed with:\n"
        << "Array subscript notation\n";

   for ( i = 0; i < 4; i++ )
      cout << "b[" << i << "] = " << b[ i ] << '\n';

   cout << "\nPointer/offset notation where\n"
        << "the pointer is the array name\n";

   for ( offset = 0; offset < 4; offset++ )
      cout << "*(b + " << offset << ") = "
           << *( b + offset ) << '\n';

   cout << "\nPointer subscript notation\n";

   for ( i = 0; i < 4; i++ )
      cout << "bPtr[" << i << "] = " << bPtr[ i ] << '\n';

   cout << "\nPointer/offset notation\n";

   for ( offset = 0; offset < 4; offset++ )
      cout << "*(bPtr + " << offset << ") = "
           << *( bPtr + offset ) << '\n';

   return 0;
}
```

Array b printed with:
Array subscript notation
b[0] = 10
b[1] = 20
b[2] = 30
b[3] = 40

Pointer/offset notation where
the pointer is the array name
***(b + 0) = 10**
***(b + 1) = 20**
***(b + 2) = 30**
***(b + 3) = 40**

Pointer subscript notation
bPtr[0] = 10
bPtr[1] = 20
bPtr[2] = 30
bPtr[3] = 40

Pointer/offset notation
***(bPtr + 0) = 10**
***(bPtr + 1) = 20**
***(bPtr + 2) = 30**
***(bPtr + 3) = 40**
Press any key to continue

ARREGLO DE CARACTERES (STRINGS) – OTROS EJEMPLOS

```
// Fig. 5.10: fig05_10.cpp
// Converting lowercase letters to uppercase letters
// using a non-constant pointer to non-constant data
#include <iostream>

using std::cout;
using std::endl;

#include <cctype>

void convertToUppercase( char * );

int main()
{
   char string[] = "characters and $32.98";

   cout << "The string before conversion is: " << string;
   convertToUppercase( string );
   cout << "\nThe string after conversion is:  "
        << string << endl;
   return 0;
}

void convertToUppercase( char *sPtr )
{
   while ( *sPtr != '\0' ) {

      if ( islower( *sPtr ) )
         *sPtr = toupper( *sPtr );  // convert to uppercase

      ++sPtr;  // move sPtr to the next character
   }
}
```

The string before conversion is: characters and $32.98
The string after conversion is: CHARACTERS AND $32.98
Press any key to continue

LIBRERIA DE MANIPULACION DE STRINGS: <cstring>

Lea la Fig. 5.29 del libro de Deitel donde se describen algunas funciones de uso comun cuando se trabaja con datos de tipo string. Estas funciones necesitan que se incluya la librería <cstring> en el programa C++ que las utiliza. Por medio de estas funciones podemos hacer, entre otras cosas, lo siguiente:

- comparar strings,
- buscar un conjunto de caracteres dentro de un string,
- separar strings en partes,
- determinar la longitud de un string

VEAMOS ALGUNOS EJEMPLOS

Estudie los ejemplos 5.31, 5.32, 5.33, 5.34 de Deitel.

En las siguientes variaciones del problema de convertir a mayúsculas el string literal, la función main permacece exactamente igual a la versión original.

```
// Fig. 5.10: fig05_10.cpp – version 2
Prototipo: void convertToUppercase( char [] );

void convertToUppercase( char string[] )
{
      int i = 0;

   while ( string[i] != '\0' ) {

      if ( islower( string[i] ) )
         string[i] = toupper( string[i] );  // convert to uppercase

        ++i;

  }
}

// Fig. 5.10: fig05_10.cpp – version 3

Prototipo: void convertToUppercase( char [] );

void convertToUppercase( char string[] )
{

   for(int i = 0; string[i] != '\0'; i++ ) {

      if ( islower( string[i] ) )
         string[i] = toupper( string[i] );  // convert to uppercase
   }
}

// Fig. 5.10: fig05_10.cpp – version 4

Prototipo: void convertToUppercase( char * );

void convertToUppercase( char *sPtr )
{

   for(; *sPtr != '\0'; *sPtr++ ) {

      if ( islower( *sPtr ) )
         *sPtr = toupper( *sPtr );  // convert to uppercase
   }
}
```

```cpp
// Fig. 5.10: fig05_10.cpp – version 5
// Desarrollo de la solucion con una sola funcion main, pero usando el
// puntero para recorrer el arreglo

#include <iostream>

using std::cout;
using std::endl;

#include <cctype>

int main()
{
   char string[] = "characters and $32.98";
   char *sPtr;

   sPtr = string;

   cout << "The string before conversion is: " << string;

   while ( *sPtr != '\0' ) {

      if ( islower( *sPtr ) )
         *sPtr = toupper( *sPtr );  // convert to uppercase

      ++sPtr;  // move sPtr to the next character
   }

   cout << "\nThe string after conversion is:  "
        << string << endl;

   return 0;
}
```