

# Punteros

## Conceptos

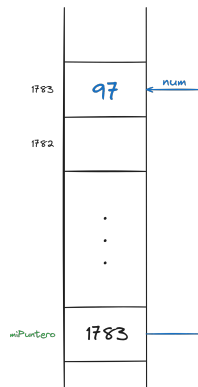
Tomás Peiretti

# Puntero: definición

Un puntero es una **variable que almacena la dirección en memoria** de otra variable. Se dice que los punteros "apuntan a" la variable cuya dirección almacenan.

Si tenemos `int * miPuntero`; diremos que `miPuntero` es un puntero que apunta a un entero

```
1 int main() {  
2     int num = 97;  
3     int * miPuntero = & num;  
4     // imprimir la direccion de num:  
5     cout << miPuntero << endl;  
6 }
```



# Punteros: operadores relacionados

Para trabajar con punteros, haremos uso de los siguientes operadores:

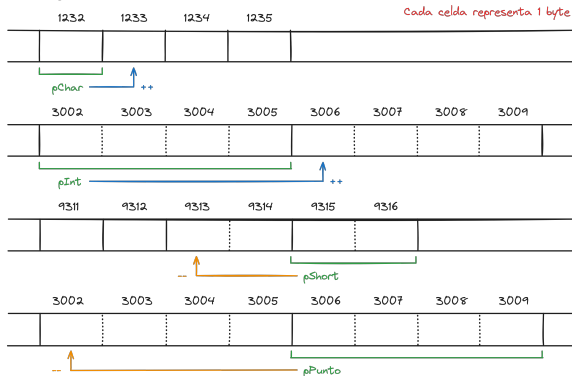
- Operador **&**: permite obtener la dirección en memoria de una variable
- Operador **\***: permite acceder al valor apuntado por un puntero (de allí su nombre operador de desreferencia)

```
1 struct Auto {
2     string marca;
3     int km;
4 };
5
6 int main() {
7     int x = 99899;
8     Auto mercedes = {"Mercedes", 13526};
9     Auto * pAuto = &mercedes;
10
11     cout << "el entero " << x << " se encuentra en " << &x << endl;
12     cout << "pAuto apunta a " << pAuto << " que es un: " << (*pAuto).marca << endl;
13     /* SALIDA:
14     el entero 99899 se encuentra en 0x71ff04
15     pAuto apunta a 0x71fee8 que es un: Mercedes */
16 }
```

# Operaciones sobre punteros

Aparte de los operadores presentados previamente, es posible realizar operaciones de suma y resta sobre los punteros. El comportamiento de estas operaciones varía según el **tamaño del tipo de dato** al que apunten.

```
1 struct Punto {  
2     short x;  
3     short y;  
4 }  
5  
6 int main() {  
7     char * pChar;  
8     int * pInt;  
9     short * pShort;  
10    Punto * pPunto;  
11  
12    pChar++;  
13    pInt += 1;  
14    --pShort;  
15    pPunto -= 1;  
16 }
```



# Punteros y arreglos

```
1 int main() {
2     int arr[5] = {10, 20, 30, 40, 50};
3     int * p = arr; // apunta al 1er elemento
4     cout << *p << endl; // imprime 10
5
6     // *(p+X) == arr[X]
7     // actualizar el 3er elemento
8     *(p+2) = 9999;
9     cout << arr[2] << endl; // imprime 9999
10
11    // imprimir el arreglo sin usar []
12    for (int * i = arr; i < arr+5 ; i++) {
13        cout << *i << " ";
14    }
15 }
```

# Punteros a punteros

C++ permite el uso de punteros que apuntan a punteros que apuntan a datos (o incluso hasta otros punteros). La sintaxis simplemente requiere de un asterisco **\*** por cada nivel de indirección:

```
1  int main() {
2      int num = 117;
3      int * pNum = &num;
4      int ** pp = & pNum;
5      int *** ppp = &pp;
6
7      // imprimir num desde p
8      cout << **p << endl;
9      // Imprimir num desde ppp
10     cout << ***ppp << endl;
11 }
```

# Punteros a punteros: matrices

```
1  const int N = 5;
2  const int M = 5;
3
4  // una matriz implementada con punteros
5  int main() {
6      // arreglo de punteros que apuntan a un arreglo de punteros que apuntan a un
        entero
7      int **m = new int*[N];
8      // creacion de la matriz completa
9      for (int ** pFila=m; pFila<m+N; pFila++) {
10         *pFila = new int[M];
11     }
12
13     // inicializacion de la matriz
14     for (int ** pFila=m; pFila<m+N; pFila++) {
15         for (int * pCol=*pFila ;pCol<*pFila+M; pCol++) {
16             cin >> *pCol;
17         }
18     }
19
20     // impresion de la fila deseada
21     int filaAlmprimir = 2;
22     int * pFila = *(m+filaAlmprimir);
23     for (int * pCol=pFila ;pCol<pFila+M; pCol++) {
24         cout << *pCol << " ";
25     }
26
27     return 0;
28 }
```

- Guía de ejercicios de práctica de la cátedra
- Ejercicios del capítulo 12 del libro *C++ para Ingeniería y Ciencias* de Gary J. Bronson.