LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

INGENIERÍA EN INFORMÁTICA

PROGRAMACIÓN

Unidad 6 - Parte 1: Variables puntero. Operadores. Punteros y direcciones. Punteros y arreglos.

2023

Lic. Mariela A. Velázquez

Revisamos la tarea pendiente

1. Qué es un arreglo en C?

Un arreglo es una colección de variables ordenadas e indexadas, todas de idéntico tipo.

2. ¿Cuál es la diferencia entre una variable y un arreglo en C?

La diferencia principal es que una variable almacena un solo valor, mientras que un arreglo almacena una colección de valores del mismo tipo.

3. ¿Cómo se declara un arreglo en C?

```
tipo nombre[tamaño];
```

4. ¿Cuál es la importancia de la longitud de un arreglo?

La longitud de un arreglo es el número de elementos que contiene y se establece cuando se declara. No puede cambiar durante tiempo de ejecución.

Revisamos la tarea pendiente

5. ¿Cuál es la forma de acceder a los elementos individuales de un arreglo?

Los elementos individuales de un arreglo se acceden utilizando un índice.

6. Qué es el índice de un arreglo y cómo se utiliza?

El índice de un arreglo es un número entero que se utiliza para acceder a un elemento específico.

7. ¿Cómo se inicializa un arreglo en C?

- ✓ Por omisión: Un arreglo declarado en forma global, se inicializa por en ceros binarios
- ✓ Por inicialización explícita: se puede inicializar al declararlo, especificando sus elementos entre llaves.
- ✓ En tiempo de ejecución: Las componentes del arreglo tomarán valores que son leídos

Revisamos la tarea pendiente

8. ¿Cuál es la diferencia entre un arreglo unidimensional y un arreglo multidimensional?

Un arreglo unidimensional tiene una sola dimensión, mientras que un arreglo multidimensional tiene múltiples dimensiones.

9. ¿Cómo se realiza la copia de un arreglo en otro en C?

Puedes copiar un arreglo en otro utilizando un bucle y asignando cada elemento o utilizando la función `strcpy` la biblioteca estándar de C.

10. ¿Qué es una cadena de caracteres en C?

Una cadena de caracteres en C es un tipo especial de arreglo de caracteres que termina con un carácter nulo ('\0').

11. ¿Cómo se puede determinar la longitud de una cadena de caracteres en C?

La longitud de una cadena de caracteres se puede determinar utilizando la función strlen de la biblioteca estándar de C.

Revisamos la tarea pendiente

12. ¿Cómo se puede recorrer un arreglo utilizando bucles (for, while)? ¿se puede recorrer con do-while?

```c
for (int i = 0; i < cant; i++)
{
   printf("arre[%d]: %d", i, arre[i]);

}
```

```c
i = 0;
while(i<cant)
{
  printf("arre[%d]: %d", i, arre[i]);
   i++;
}
```

```c
int i = 0;
do {
      printf("arre[%d]: %d", i, arre[i]);
      i++;
   } while (i < cant);
```

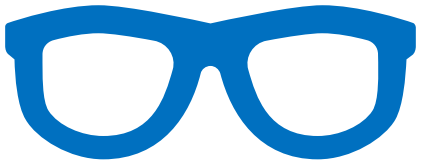
13. ¿Cómo se pasa un arreglo a una función como argumento?

Para pasar un arreglo a una función como argumento, puedes especificar el nombre del arreglo. La función recibirá un puntero al primer elemento del arreglo.

14.¿Una función puede retornar un arreglo?

En C, una función no puede devolver un arreglo.

Tener en cuenta



Todo lo que leerá a partir de este momento es solo una guía para la clase de teoría.

Usted como estudiante debe tomar apuntes de lo tratado en clase de manera de enriquecer sus conocimientos.

Introducción

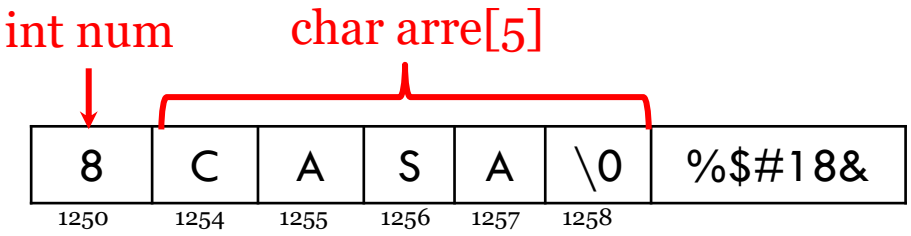
Cada vez que se declara una variable C, el compilador establece un área de memoria para almacenar el contenido de la variable.

Por ejemplo cuando se declara una variable int, el compilador asigna dos bytes de memoria. El espacio para esa variable se sitúa en una posición específica de la memoria, conocida como dirección de memoria . Cuando se referencia, es decir hace uso de la variable, el compilador de C accede automáticamente a la dirección de memoria donde se almacena el entero.

Se puede ganar en eficacia en el acceso a esta dirección de memoria utilizando un puntero .

Introducción

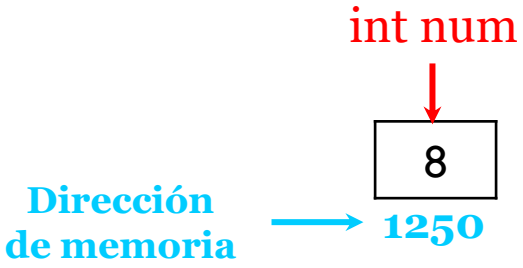
Cada vez que se declara una variable C



Representación de la memoria.

El compilador establece un área de memoria para almacenar el contenido de la variable

El espacio para esa variable se sitúa en una posición específica de la memoria, conocida como dirección de memoria



Se puede ganar en eficacia en el acceso a esta dirección de memoria utilizando un puntero

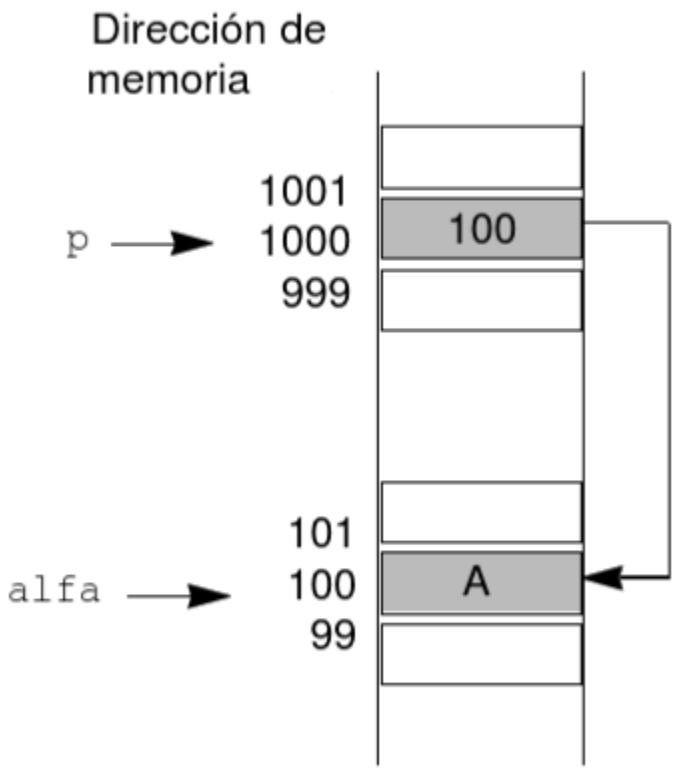
¿Qué es un puntero?

Un puntero es una variable que contiene la dirección de memoria de un dato o de otra variable que contiene al dato. Quiere esto decir que el puntero apunta al espacio físico donde está el dato o la variable.

Los punteros se rigen por estas reglas básicas…

- Un puntero es una variable como cualquier otra;
- Una variable puntero contiene una dirección que apunta a otra posición en memoria;
- En esa posición se almacenan los datos a los que apunta el puntero;

Grafica de un puntero…



p contiene el valor 100, que es la dirección de alfa

*p es el valor del elemento al que apunta p. Por consiguiente, *p toma el valor 'A'

Declaración de punteros

La declaración de un puntero consiste en:

tipo *nombre-vble-puntero

Donde:

tipo: cualquier tipo de c (int, char, float).

*: Operador de indirección

nombre-vble-puntero: identificador.

Ejemplo:

```
int *puntero; // Declaración de un puntero a un entero
```

Declaración de punteros

Esta declaración significa que:

- La variable definida de tipo puntero apunta a objetos del tipo base especificado.
- La variable puntero es capaz de almacenar la dirección del objeto al cual apunta.
- El valor de un puntero es una dirección.
- El objeto al cual apunta un puntero se llama objeto referenciado. Los objetos referenciados son del tipo base especificado.

Los operadores de un puntero

& Operador de dirección

El &, devuelve la dirección de memoria de su operando.

p1 = &car; //p1 recibe la dirección de car.

Podemos ver que el operador & sirve para acceder a la dirección, NO al contenido del objeto referenciado.

Recuerdan:

```
scanf("%d", &numero);
```

Ahora podemos entender, que la función scanf(), tiene como parámetro la dirección de memoria de la variable en la cual queremos almacenar un valor.

Los operadores de un puntero

- * Operador de indirección

El operador * es el complemento del operador **&**, devuelve el contenido del objeto referenciado.

Importante: Los operadores & y * tienen precedencia superior a todos los operadores aritméticos.

Asignación de direcciones

Puedes asignar la dirección de memoria de una variable a un puntero utilizando el operador de dirección &.

Por ejemplo:

```
int numero;
int *pnumero;

pnumero = &numero;
```

Como otras variables en C, los punteros también pueden ser inicializados en su definición. Por ejemplo:

```
int numero;
int *pnumero = &numero;
```

Asignación de direcciones

Veamos los datos que tenemos al asignar una direccion a un puntero y las diferentes

```c
#include <stdio.h>

int main()
{
    int numero=11, *pnumero;

    pnumero = &numero;

    printf("El contenido de lo que apunta el puntero: %d\n", *pnumero);
    printf("Lo que contiene de dato el puntero: %p\n", pnumero);
    printf("La direccion de memoria de la vble numero: %p\n", &numero);
    printf("Lo que contiene de dato la vble numero: %d\n", numero);

    return 0;
}
```

Ejemplos de uso de punteros

Supongamos que tienes una variable que almacena la cantidad de dinero en dólares y desea calcular su equivalente en otra moneda utilizando un puntero.

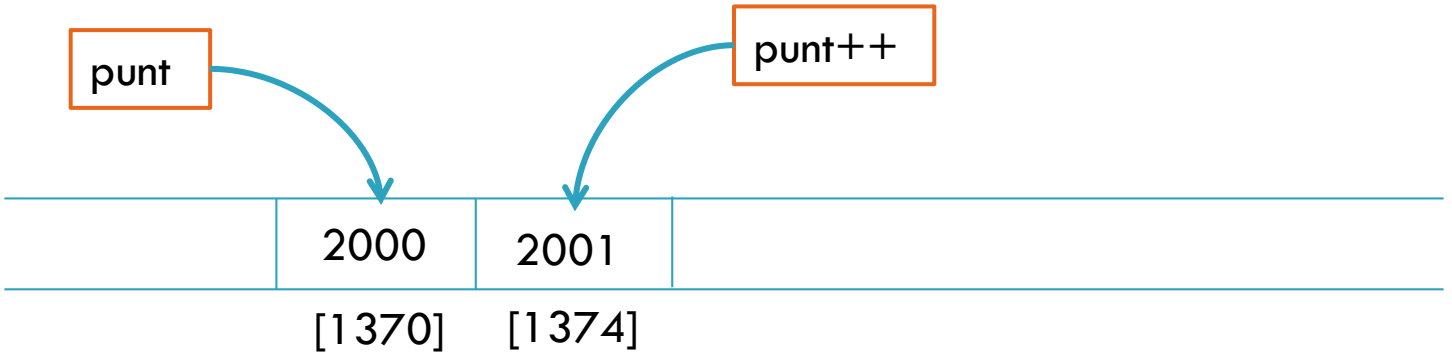
```c
#include <stdio.h>

int main() {
    float monto_dolares = 500.0;  // Cantidad de dinero en dólares
    float tipo_cambio = 920.0;     // Tipo de cambio
    float monto_otra_moneda;      // Cantidad de dinero en la otra moneda

    // Declaración de un puntero a una variable de tipo float
    float *ptr_monto_otra_moneda = &monto_otra_moneda;

    // Realizar la conversión de dólares a la otra moneda a través del puntero
    *ptr_monto_otra_moneda = monto_dolares * tipo_cambio;

    printf("Cantidad de dinero en dolares: $%.2f\n", monto_dolares);
    printf("Cantidad de dinero en la otra moneda: %.2f\n", *ptr_monto_otra_moneda);

    return 0;
}
```

Aritméticas de Punteros (de Direcciones)

Solo se pueden realizar dos operaciones: +, -

Para entender que ocurre en la aritmética de direcciones, lo veremos mediante un ejemplo:

```
int *punt;
```

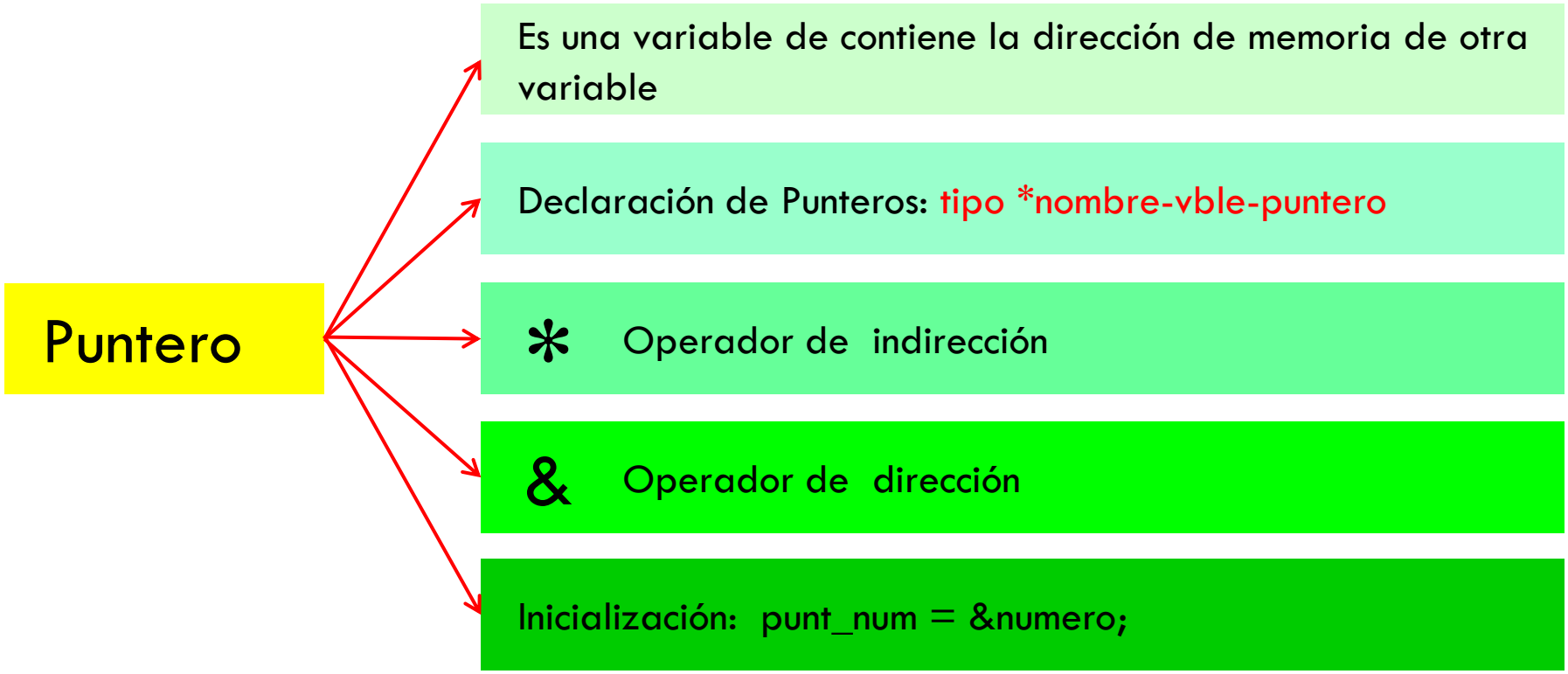


Aritméticas de Punteros (de Direcciones)

Cada vez que incrementamos el puntero punt, apuntará tantas direcciones más como bytes tenga el tipo base.

Pensemos un entero tiene 4 bytes de longitud, cuando se incrementa un puntero entero, su valor crece en dos. Es decir, el incremento o decremento se produce en función de su tipo base.

En resumen

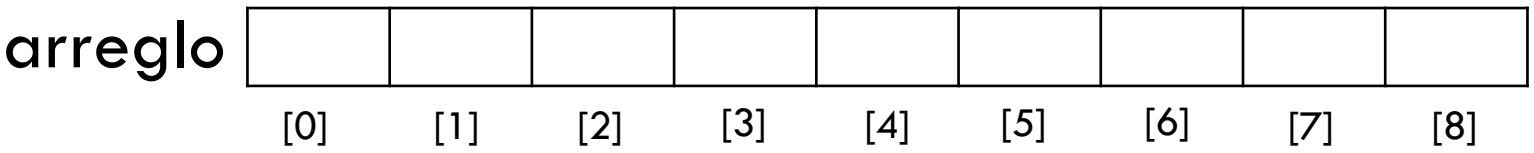

Puntero
Es una variable de contiene la dirección de memoria de otra variable
Declaración de Punteros: tipo *nombre-vble-puntero
* Operador de indirección
& Operador de dirección
Inicialización: punt_num = №

Punteros y Arreglos

- En C existe una fuerte relación entre arreglos y punteros.
- Cualquier operación que pueda lograrse por indexación de un arreglo, también puede realizarse con punteros.
- Las versiones con punteros son más rápidas.

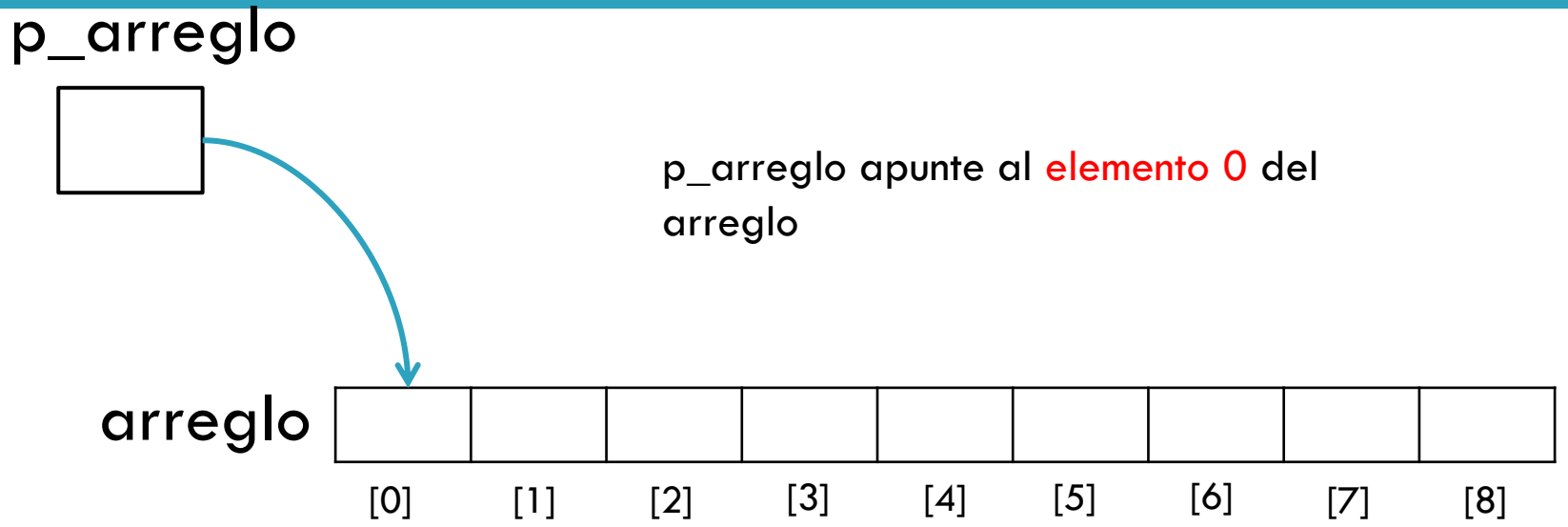
Declaraciones

```
int arreglo[9];
```




```
int *p_arreglo;

p_arreglo = &arreglo[0]; //asignación
```

Asignación

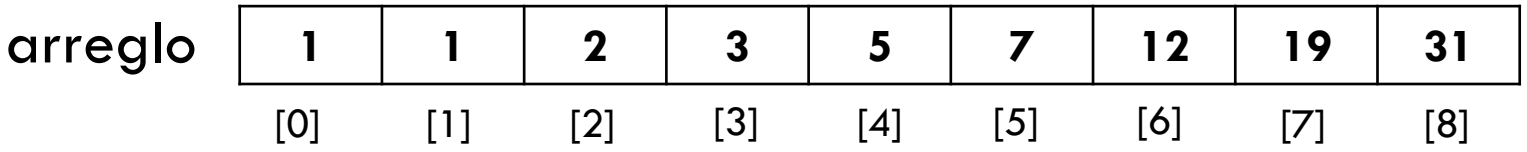


Sabemos que, el nombre del un arreglo es sinónimo a la dirección del primer elemento del arreglo, por lo tanto la siguiente asignación es válida:

p_arreglo = arreglo;

Asignación

```
int aux;
int arreglo[9];
int *p_arreglo;

p_arreglo = arreglo;
aux = *p_arreglo; // copia el contenido del arreglo[0] en aux
```

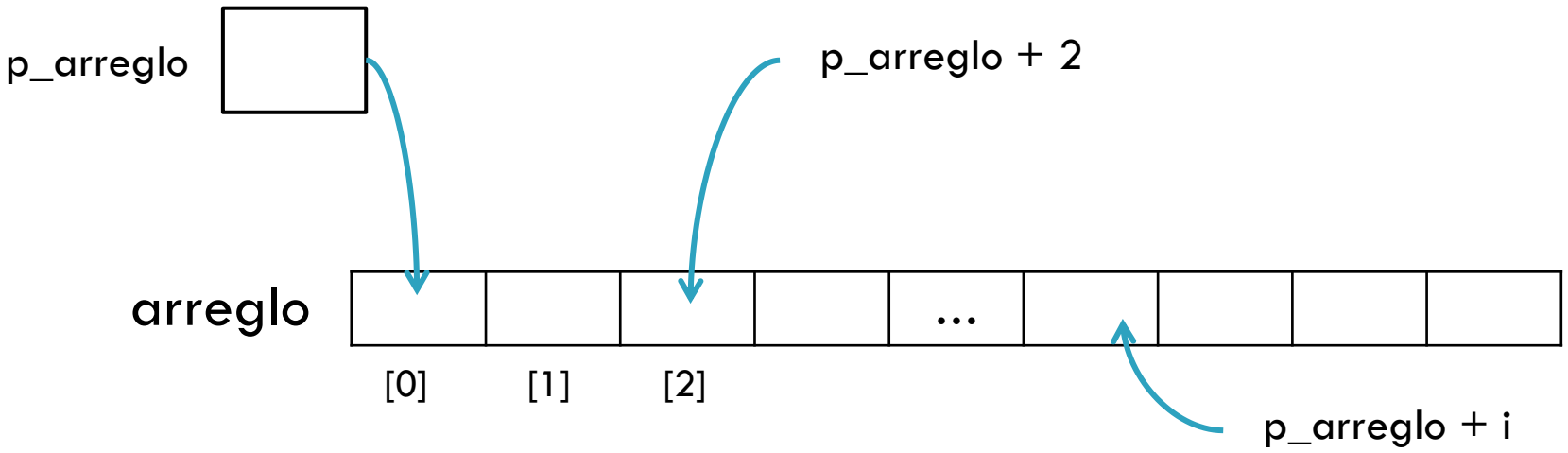


En general

Si p_arreglo apunta a un elemento en particular del arreglo,

p_arreglo+1 apunta al siguiente elemento y p_arreglo+i apunta i

elementos después de la dirección de arreglo[0]



Ejemplo

```c
#include <stdio.h>
#include <ctype.h>

int main()
{
    char cadena[80];
    int  i;

    puts("Introducir una cadena en mayuscula");
    gets(cadena);

    puts("Esta es la cadena en miniscula");

    for(i=0; cadena[i]!='\0'; i++)
    {
        printf("%c", tolower(cadena[i]));
    }

    return 0;
}
```

ARREGLOS

```c
#include <stdio.h>
#include <ctype.h>

int main()
{
    char cadena[80], *p_cadena;

    puts("Introducir una cadena en mayuscula");
    gets(cadena);

    puts("Esta es la cadena en miniscula");
    p_cadena = cadena;

    while(*p_cadena != '\0')
    {
        printf("%c", tolower(*p_cadena) );
        p_cadena++;
    }

    return 0;
}
```

PUNTEROS

Punteros

La velocidad es un bien preciado en programación, se debe saber que entre ambas versiones, la de punteros es más rápida.

Existe una diferencia muy importante entre nombre del arreglo y un apuntador.

Un puntero es una variable, es licito escribir:

```
p_cadena = cadena ;

p_cadena ++;
```

En tanto: un nombre de arreglo NO es una variable, por lo cual no es correcto escribir:

```
cadena++;

cadena = punt;
```

Tarea para la casa

1. ¿Qué es un puntero en C y para qué se utiliza?
2. ¿Cómo se declara un puntero en C? Proporciona ejemplos.
3. ¿Cuál es la diferencia entre un puntero y una variable normal en C?
4. ¿Qué es la dirección de memoria de una variable y cómo se obtiene?
5. ¿Cómo se asigna la dirección de memoria de una variable a un puntero?
6. ¿Qué es la indirección de punteros y cómo se realiza en C?
7. ¿Cómo se realiza la aritmética de punteros en C? Proporciona ejemplos.
8. ¿Cuál es la diferencia entre punteros y arreglos en C? ¿Cómo se relacionan?
9. ¿Por qué son importantes los punteros al trabajar con cadenas de caracteres en C?

Lo revisamos la próxima clase.

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```