DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Análisis de Algoritmos y Estructura de Datos

Conceptos de TDA y punteros

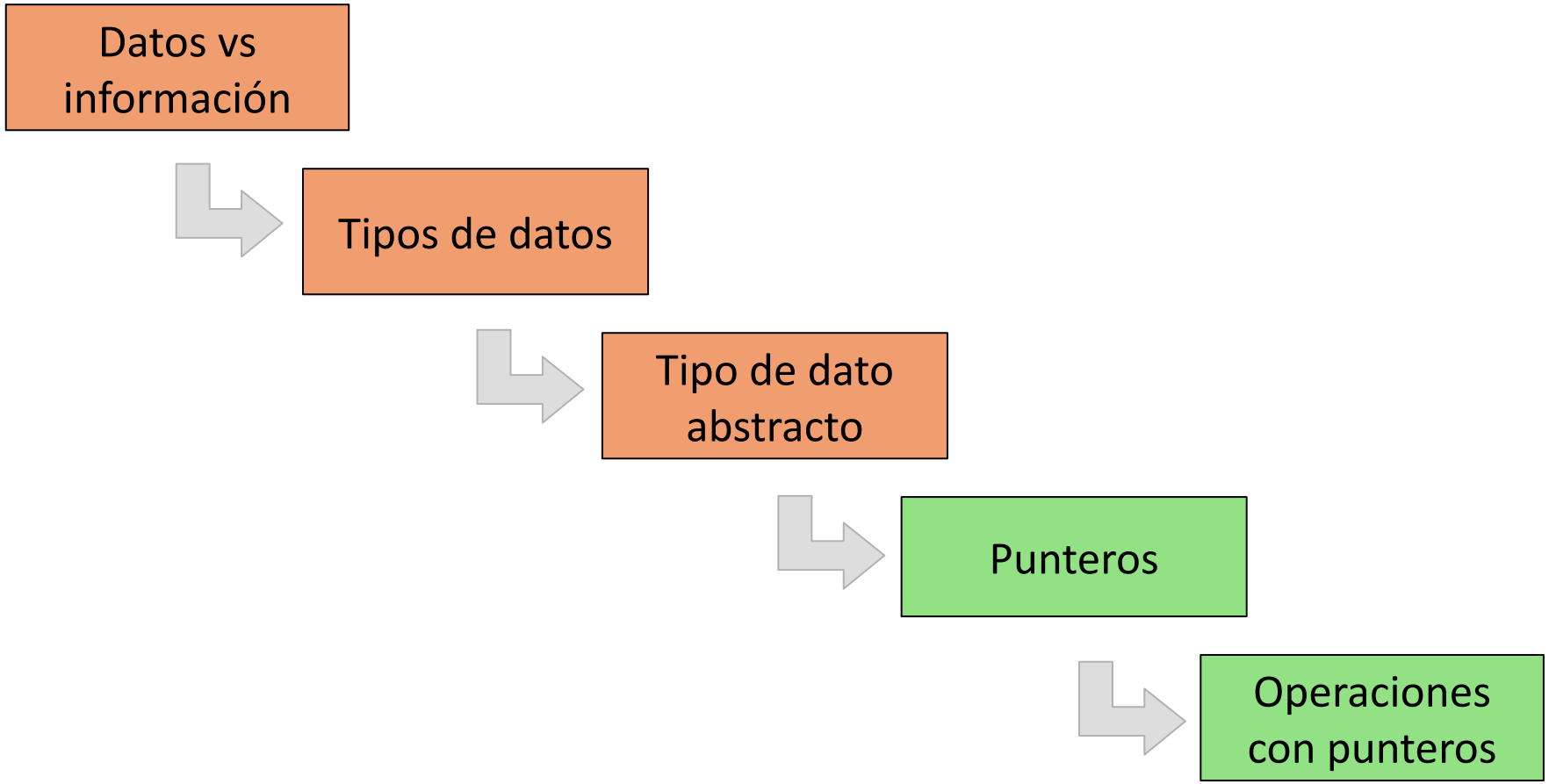
Prof. Violeta Chang C.

Semestre 2 – 2023

Conceptos de TDA y punteros

- **Contenidos**:
  - Conceptos básicos de TDA
  - Conceptos básicos de punteros
- **Objetivos**:
  - Comprender concepto de TDA y ejemplificarlo
  - Entender nociones básicas de punteros (definición y operaciones)

Ruta de la sesión



Datos vs. Información

- **Dato**:
  - Conjunto de caracteres con sólo representación simbólica y que no ofrece algún significado específico
  - Ejemplos: 20, omega, etc
- **Información**:
  - Conjunto de datos capturado, organizado, clasificado e integrado dentro de un contexto y significado
  - Ejemplos: Laura tiene 20 años, omega representa la velocidad uniforme etc

Tipos de datos

- Todos los lenguajes de programación soportan ciertos tipos de datos
- Ejemplo: Lenguaje de programación numérica Matlab soporta tipos de datos tales como
  - enteros
  - reales
  - caracteres
  - arreglos
  - entre otros.

¿Cuántos tipos de datos soporta el lenguaje de programación C?

Tipos de datos

- Un tipo de dato se define formalmente como un conjunto de valores (dominio) y un conjunto de operaciones definidas sobre estos valores

dominio + operaciones

- Ejemplo: Reales=<$\Re$, {+,*,/,-,^,...}>

Tipos de datos

- Los lenguajes de programación consideran a las variables y constantes de un programa como instancias de un tipo de datos
- Un tipo de dato proporciona una descripción de sus instancias que indican al compilador información como:
  - Cuánta memoria se debe asignar para una instancia
  - Cómo interpretar los datos en memoria
  - Cuáles operaciones son permisibles sobre estos datos

Tipos de datos

- Ejemplo: Si en C se escribe una declaración de variable como

float num

- se está declarando una instancia denominada *num* del tipo de datos *float*.
- El tipo de dato *float* indica al compilador que:
  - Reserve 32 bits de memoria para almacenar su valor
  - Las operaciones como sumar, restar y multiplicar están aceptadas
  - La operación % (división en módulo) no es permitida

Tipos de datos

- **Tipos de datos primitivos o predefinidos**:
  - son los que se construyen en un compilador
- **Tipos de datos estructurados o compuestos**:
  - son organizaciones especiales de tipos primitivos para cuando se necesita procesar colecciones de datos del mismo tipo

Tipos de datos

- El modelo matemático o lógico de una organización particular de datos se conoce como **estructura de datos**.
- Los elementos de una estructura de datos particular se conocen como nodo, elemento, registro, entre otros, dependiendo del tipo de estructura, el contexto en el que se utiliza y quién la utiliza.

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

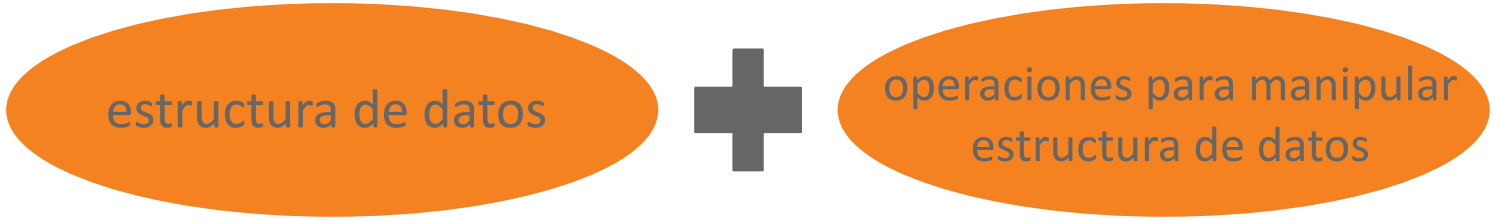
Operaciones con estructuras de datos

Tipo de dato abstracto

- La mayoría de lenguajes de programación permiten definir tipos de datos compuestos o estructurados
- Un tipo de datos definido por el programador y no por el autor del compilador se denomina **tipo de dato abstracto**
- Ejemplo: Un tipo de dato llamado *Punto* que representa las coordenadas *x* y *y* de un sistema de coordenadas para un pixel en la pantalla

Tipos de dato abstracto

- Un tipo de datos abstracto es un tipo de datos que consta de:

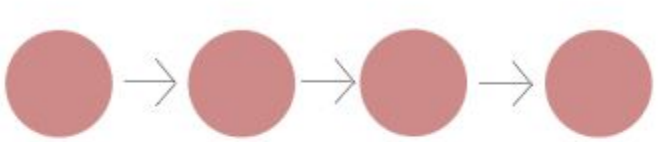


- Los tipos de datos abstractos brindan varias ventajas:
  - Permiten una mejor conceptualización y modelamiento del mundo real
  - Permiten la comprobación de tipos para evitar errores de tipo en tiempo de ejecución
  - Permiten la reutilización de funciones

Estructura de datos

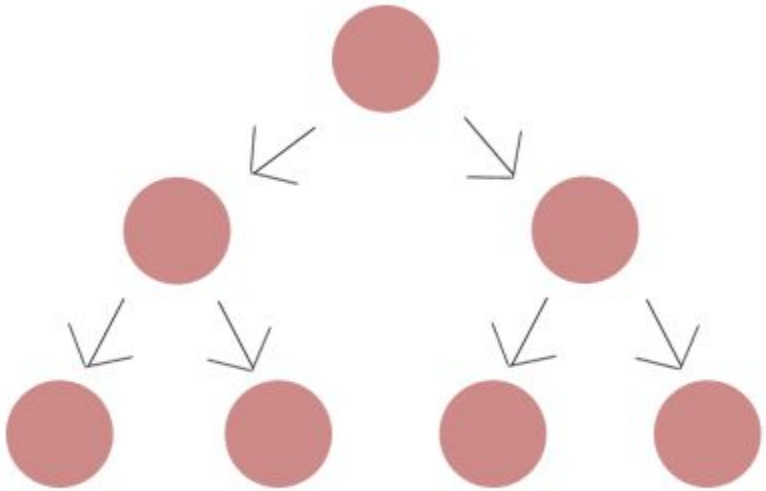
- Estructuras de datos lineales y no lineales

Estructuras de datos lineales



Ejemplos:
- arreglos
- pilas
- colas
- listas enlazadas

Estructuras de datos no lineales



Ejemplos:
- grafos
- árboles

Megan Wong, Introduction to linked lists

Tipo de dato abstracto

- Ejemplo: TDA arreglo
  - **Estructura de datos**: almacena linealmente elementos del mismo tipo *tipoElemento*. Tiene un espacio de memoria definido que puede estar ocupado parcial o completamente.
  - **Operaciones**: calcularTamaño, insertar, eliminar, arregloVacío, recorrer, buscar, ordenar

Tipo de dato abstracto

- Ejemplo: TDA cadena
  - **Estructura de datos**: almacena elementos de tipo caracter dispuestos linealmente pudiendo aumentar o reducir su tamaño pero no su espacio de memoria. Toda cadena tiene un largo (cantidad de caracteres)
  - **Operaciones**: concatenar, agregar, remover, cadena vacía, comparar, inversa, calcularLongitud

Tipo de dato abstracto

- Ejercicio 1: definir TDA matriz
  - **Estructura de datos**: ?
  - **Operaciones**: ?

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Intermedio en Menti

- Ir a menti.com e ingresar código 82689993

Tipo de dato abstracto

- **<u>Ejercicio 2</u>: TDA pixel**
  - Sea P(x,y) un pixel en la pantalla con coordenadas (x,y) horizontal y vertical, respectivamente. Cada pixel en la pantalla se visualiza de un determinado color, dentro de una paleta de colores permisibles. Por ejemplo, la paleta de tonos de gris está compuesta por 256 colores diferentes (0: negro, 255: blanco). Con un pixel determinado, se puede querer apagar el pixel (no permitir que se visualice su color en pantalla), cambiarle el color, cambiarle de posición arbitrariamente (vertical u horizontal), etc

  - Se pide definir el TDA pixel, especificando su estructura de datos y al menos 6 operaciones para trabajar con dicha estructura.

Tipo de dato abstracto

- **<u>Ejercicio 3</u>: TDA diccionario**
  - Un diccionario es un tipo de dato que permite acceder a elementos de datos por su contenido. Por lo mismo, cada elemento del diccionario tiene un nombre y una definición. Se pueden insertar, eliminar y modificar elementos del diccionario.

  - Definir el TDA diccionario, especificando su estructura de datos y al menos 6 operaciones para trabajar con dicha estructura.

Punteros

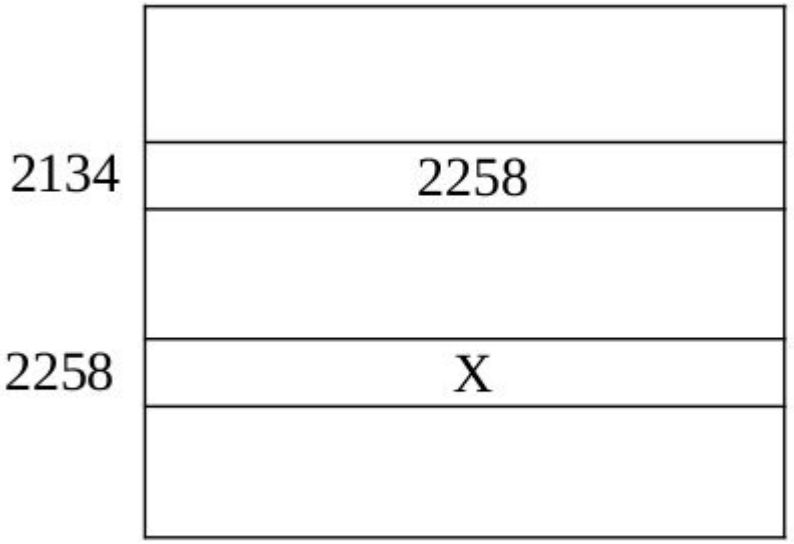
- Para facilitar que la memoria del computador está organizada en forma tal que se necesita la dirección de la posición para poder acceder a su contenido, la mayoría de variables se referencian mediante un nombre y no por direcciones
- Según esto, si necesitamos almacenar un dato, definimos una variable al inicio del programa y le asignamos un valor.
- Este valor ocupa una o más posiciones de memoria.
- Para cada dato diferente, debemos definir una variable diferente y, si es necesario, ocupar otras posiciones de memoria.

Punteros

- Un puntero es una variable que se utiliza para almacenar direcciones en vez de valores.
- Es decir, un puntero almacena la dirección de memoria donde se encuentra almacenada otra variable, conocida como *variable apuntada*.

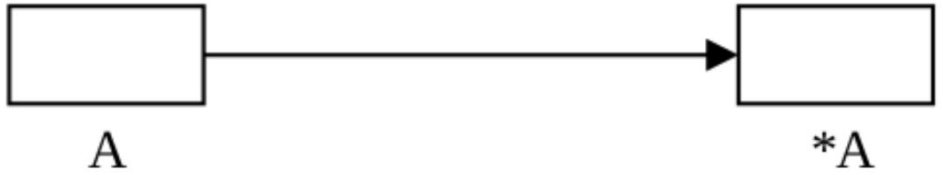
Punteros

- Ejemplo: Supongamos que A es una variable puntero hacia otra de tipo caracter y, además, sabemos que A se encuentra en la dirección de memoria cuya dirección es 2134, entonces

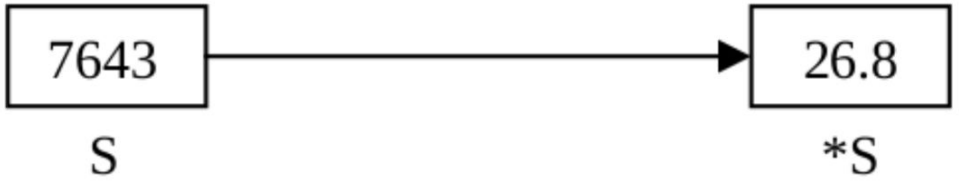


Punteros

- Los punteros se representan mediante diagramas:



- Ejemplo: La variable S contiene 7643 que es la dirección de memoria donde está la variable real apuntada 26.8



Punteros

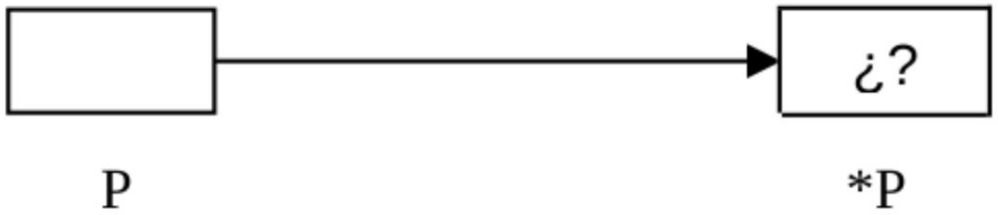
- Cuando un puntero no apunta a ninguna otra posición de memoria se le llama puntero nulo
- Cuando se tiene un puntero nulo, no se puede hacer referencia a la variable apuntada
- Cuando se define un puntero es necesario indicar el tipo de variable a la que apunta
  - Se puede declarar un puntero a una variable de tipo caracter, un puntero a un entero o a un real, un puntero a un arreglo, a un registro, etc.

Operaciones con punteros

- Luego de declarar una variable puntero, se puede utilizar como cualquier otra variable
- La posición de memoria apuntada por el contenido de la variable puntero P se representa como ***P**
- Se pueden asignar valores a *P y utilizar esos valores en expresiones
- Sin embargo, el primer paso es inicializar el puntero, luego de esto se puede asignar valores, comparar y liberar memoria ocupada

Operaciones con punteros

- **Inicialización de punteros:**
  - La declaración de una variable puntero no define una posición de memoria a la cual apuntar, es decir, si se declara *p* como una variable puntero, **p* no existe todavía
  - Es necesario asignar espacio de memoria suficiente para almacenar el tipo de datos de la variable apuntada (**p*), por lo que la única condición es que exista espacio libre de memoria



Operaciones con punteros

- **Asignación de punteros:**
  - Cuando se trabaja con variables dinámicas (apuntadas), puede existir uno o más punteros haciendo referencia a la misma variable apuntada
  - Una variable puntero puede apuntar a diferentes variables dinámicas en distintos instantes de tiempo

Operaciones con punteros

- **Asignación de punteros:**
- Sean P y Q dos variables puntero definidas hacia valores reales; entonces puede ocurrir cualquiera de dos casos:
  - Asignación de variables puntero (P←Q)
  - Asignación de variables apuntadas (*P←*Q)

Operaciones con punteros

- **Asignación de variables puntero (P←Q)**
  - Los punteros P y Q apuntan a la misma variable real dinámica
  - *P y *Q designan la misma posición de memoria y, por lo mismo, tienen el mismo valor

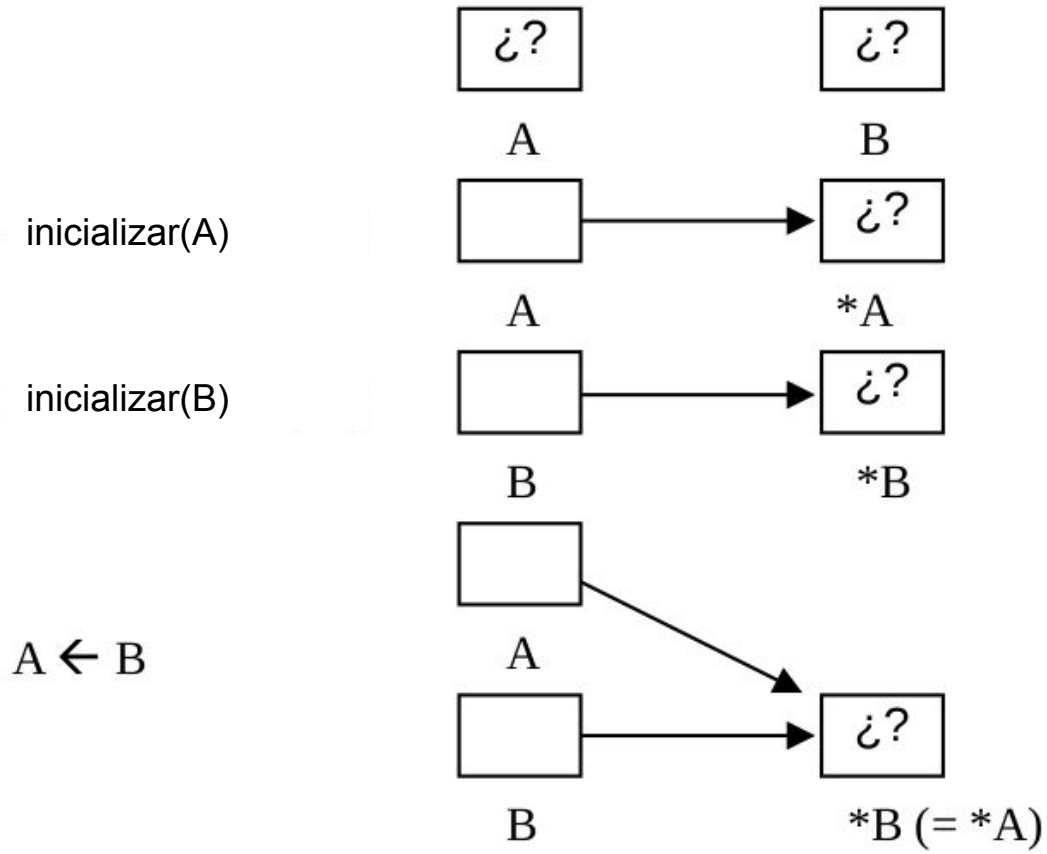
Operaciones con punteros

- **Asignación de variables puntero (P←Q)**
  - Ejemplo: Supongamos que A y B son punteros a tipo entero. Si se produce la sgte. secuencia de eventos:

    inicializar(A)

    inicializar(B)

    A ← B
  - Gráficamente se entiende como:



Operaciones con punteros

- **Asignación de variables apuntadas (*P←*Q)**
  - Las variables dinámicas *P y *Q tienen el mismo valor
  - Las variables dinámicas *P y *Q están en posiciones de memoria diferentes

Operaciones con punteros

- **Asignación de variables apuntadas (*P←*Q)**
  - Ejemplo: Supongamos que A y B son punteros a tipo entero. Si se produce la sgte. secuencia de eventos:

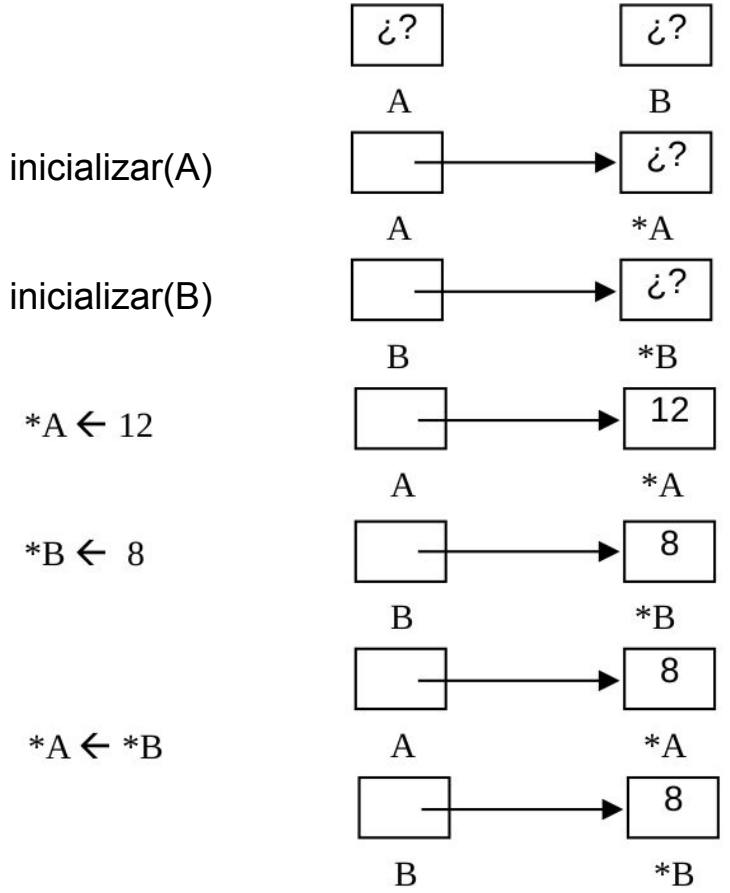
    inicializar(A)

    inicializar(B)

    *A ← 12

    *B ← 8

    *A ← *B
  - Gráficamente se entiende como:



Operaciones con punteros

- **Comparación de punteros:**
  - Sean P y Q dos punteros:
    - P=Q es válido
    - P<>Q es válido
    - P>Q es ilegal
    - P<=Q es ilegal

Dos punteros P y Q son iguales siempre que apunten precisamente a la misma variable dinámica

Operaciones con punteros

- **Comparación de punteros:**
- En el caso de las variables apuntada, es válido utilizar cualquier operador de igualdad o relación.
  - *P < *Q es válido
  - *P <> *Q es válido
  - *P = *Q es válido

Operaciones con punteros

- **Liberación de punteros:**
- Cuando ya no se necesita hacer referencia a variable apuntada por un puntero:
  - Se debe liberar la memoria ocupada en la pila de variables dinámicas
  - La variable apuntada deja de existir puesto que la variable puntero queda indefinida

Observaciones importantes

- No se puede asignar un puntero de un tipo para que apunte a una variable dinámica de otro tipo
- Después de liberar un puntero, cualquier referencia a la variable apuntada por dicho puntero produce un valor incorrecto

Observaciones importantes

- Si P es un puntero y *P es la variable apuntada, entonces:
  - Escribir(*P) es válido porque *P contiene un valor del tipo de la variable apuntada (entero, real, carácter, etc)
  - P←58,45 no es válido porque no se puede almacenar un valor de tipo real en una variable puntero que sólo almacena direcciones de memoria

Ejercicio con punteros

- **<u>Ejercicio 4</u>: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor de *p y *q

```
inicializar(p)
inicializar(q)
*p←34
*q←12
q←p
*q←23
```

Ejercicio con punteros

- **<u>Ejercicio 5</u>: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor de *p y *q

```
inicializar(p)
*p←'S'
inicializar(q)
*q←2
*p←*q
```

Ejercicio con punteros

- **<u>Ejercicio 6</u>: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor de *p y *q

```
inicializar(p)
inicializar(q)
*q←16
*p←*q/8
q←p
*p←4
```

Ejercicio con punteros

- **<u>Ejercicio 7</u>: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor de *p y *q

```
inicializar(p)
inicializar(q)
*p←8
*q←*p
liberar(p)
```

Ejercicio con punteros

- **<u>Ejercicio 8</u>: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor final de *p y *q

```
inicializar(p)
inicializar(q)
*p←28
*q←12
*p←*q
liberar(p)
```

Actividad de cierre

- Ir a menti.com e ingresar código 89 76 79 3

Próximas fechas...

- Resumen de la semana:
  - Punteros
  - Tipo de dato abstracto

~~cátedra~~ – ~~refuerzo~~ – laboratorio

- Próxima semana:
  - **PEP1: martes 03/octubre**
  - **Tarea1: miércoles 04/octubre**

- Próxima clase:
  - TDA lista enlazada
  - TDA lista enlazada circular
  - TDA lista doblemente enlazada

U2 - S4