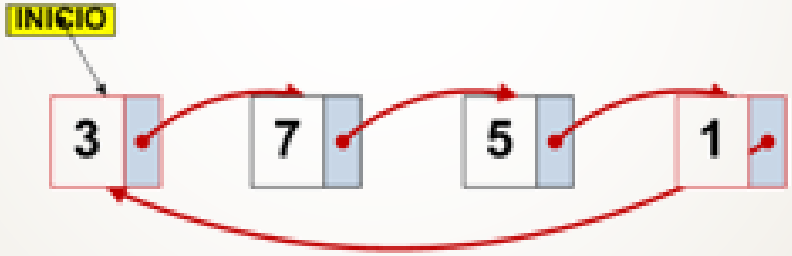
Analista Programador Universitario

Estructura de Datos

Unidad I: Punteros

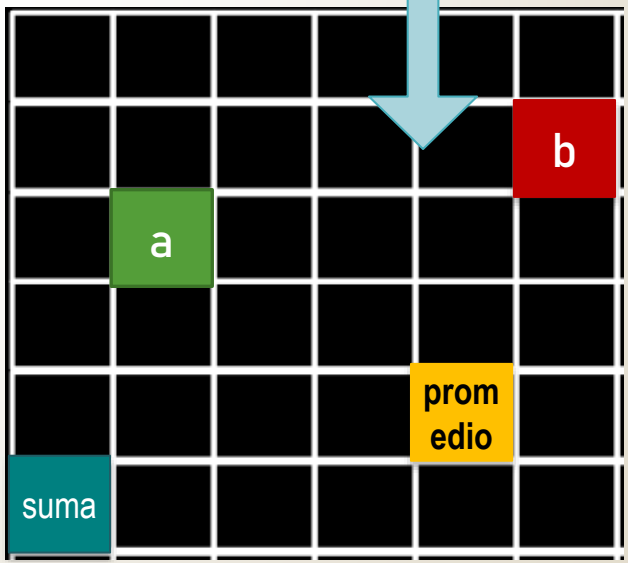


Facultad de Ingeniería
Universidad Nacional de Jujuy

Variables Estáticas y Dinámicas (1)

- Una variable estática se crea en la memoria al ejecutar un programa. Ésta existe hasta que el programa finaliza.
- El espacio de memoria reservado no está disponible para otras aplicaciones hasta que el programa finalice.

```
PROGRAMA ejemplo
CONSTANTES
   MAX=20
TIPOS
  …
VARIABLES
 a, b, suma: entero
 promedio: real
INICIO
     instrucciones
FIN
```

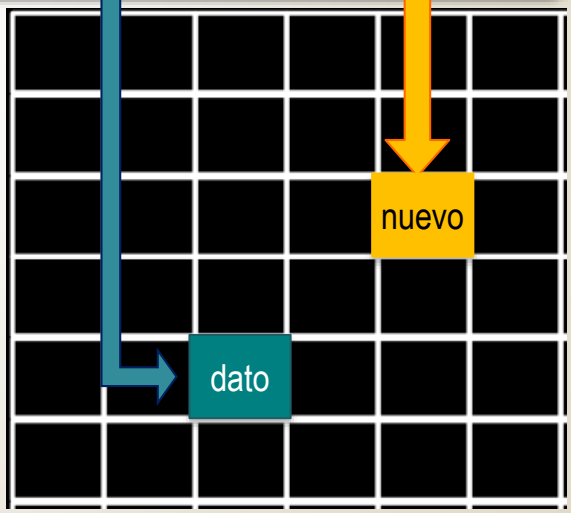


Variables del programa almacenadas en memoria

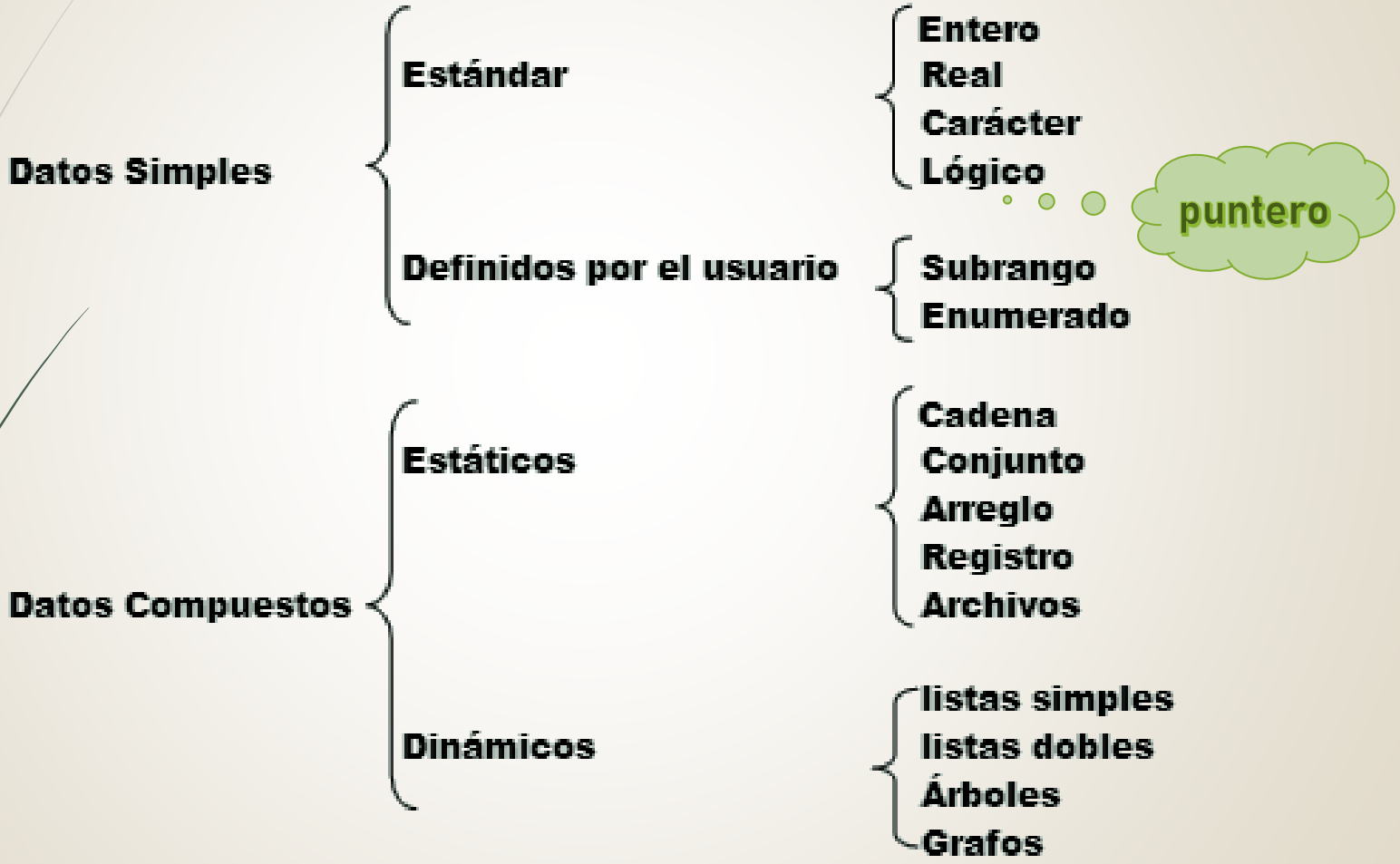
Variables Estáticas y Dinámicas

- Una variable dinámica es aquella que se crea durante la ejecución de un programa, a partir del espacio de memoria libre. Para ello, se reserva un espacio de memoria y una etiqueta. Una variable dinámica puede eliminarse en cualquier momento (liberando espacio de memoria).

```
PROGRAMA ejemplo
CONSTANTES
   MAX=20
TIPOS
  …
VARIABLES
 a, b, suma: entero
 promedio: real
INICIO
   reservar memoria
   …
   liberar memoria
FIN
```

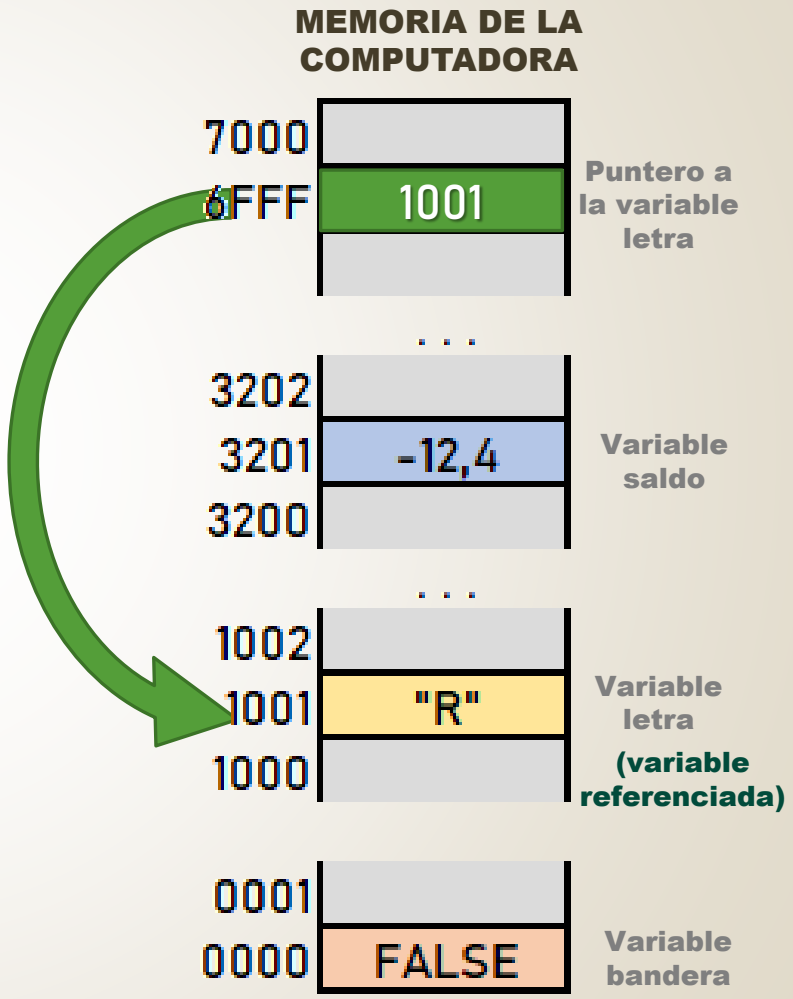


Tipos de Datos



Punteros (1)

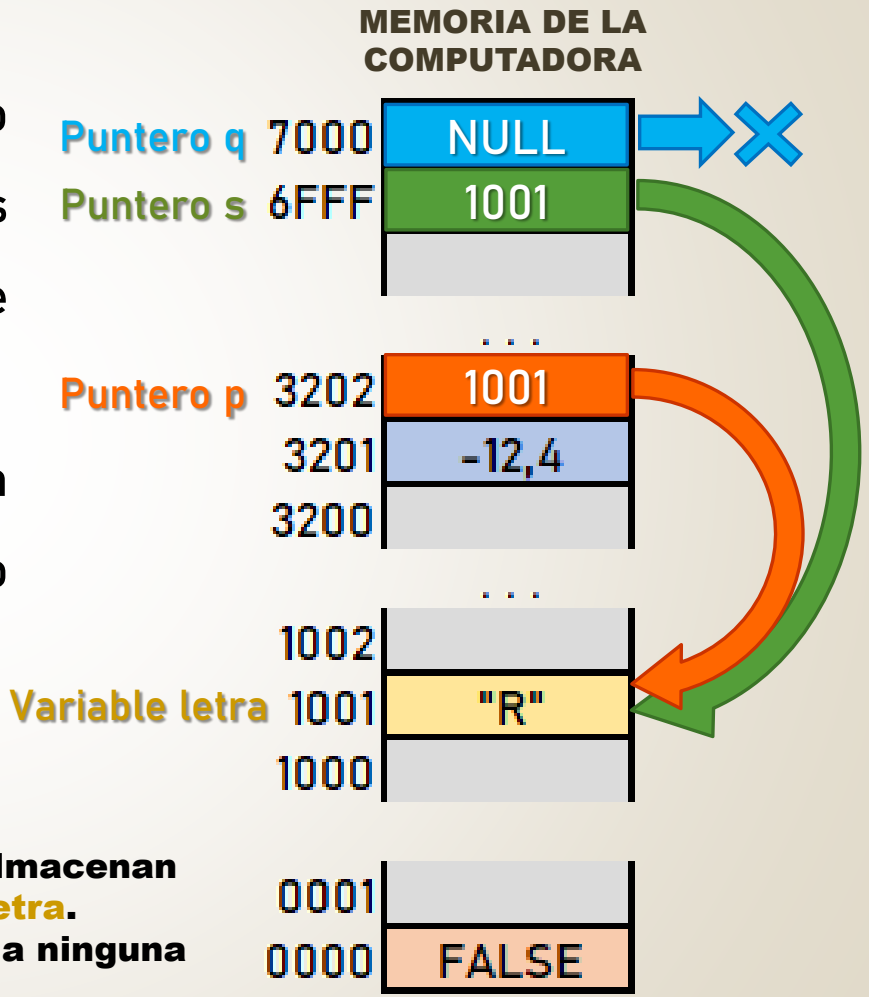
- El TDA puntero es un tipo de dato simple que almacena la dirección de memoria de otra variable, denominada variable referenciada.
- Un puntero "apunta" al espacio de memoria ocupado por otra variable.



La variable puntero almacena la dirección de memoria de la variable letra

Punteros (2)

- Las variables de tipo puntero pueden hacer referencia a variables reales, enteras, lógicas, de caracteres o datos compuestos.
- Una variable o dato de memoria puede ser referenciado por uno o más punteros.



Las variables punteros p y s referencian (almacenan la dirección de memoria) la variable letra.
El puntero q tiene el valor NULL, no apunta a ninguna posición de memoria.

Operaciones sobre punteros (1)

- El TDA puntero tiene asociadas las operaciones de asignación y comparación de punteros.
- La asignación permite almacenar una dirección de memoria en una variable de tipo puntero. El valor NULL indica que el puntero no guarda ninguna dirección.
- La operación de comparación permite determinar si 2 variables de tipo puntero referencian o no la misma dirección de memoria, o si un puntero tiene valor NULL.

variable_puntero=dirección_dato
variable_puntero=NULL

variable_puntero==variable_puntero
variable_puntero==NULL

variable_puntero!=variable_puntero
variable_puntero!=NULL

Operaciones sobre punteros (2)

```
#include <iostream>
#include <stdlib.h>
using namespace std;
typedef int *p_entero;
main()
{ int dato;
  p_entero p;
  cout << "Ingrese un dato:";
  cin >> dato;
  cout << "Valor ingresado: " << dato << endl;
  p=&dato;
  cout << "Direccion del dato: " << p << endl;
  system("pause");
}
```

```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Ingrese un dato:54
Valor ingresado: 54
Dirección del dato: 0x71ff08
Presione una tecla para continuar . . .
```

Guardando la dirección de la variable dato en el puntero p

Operaciones sobre punteros (2)

```
#include <iostream>
#include <stdlib.h>
using namespace std;
typedef int *p_entero;
main()
{ int dato;
  p_entero p;
  cout << "Ingrese un dato:";
  cin >> dato;
  cout << "Valor ingresado: " << dato << endl;
  p=&dato;
  cout << "Direccion del dato: " << p << endl;
  cout << "Valor apuntado por el puntero p: " << *p << endl;
  system("pause");
}
```

```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Ingrese un dato:67
Valor ingresado: 67
Dirección del dato: 0x71ff08
Valor apuntado por el puntero p: 67
Presione una tecla para continuar . . .
```

Guardando la dirección de la variable dato en el puntero p

Acceso al valor de la variable apuntada por p

Solicitud/Liberación de memoria (1)

- Para obtener espacio de memoria se solicita al SO el espacio suficiente para almacenar un tipo de dato específico mediante la operación new.

```
variable_puntero=new tipo_dato;
```

- Para liberar espacio de memoria se le indica al SO operativo que el espacio de memoria indicado ya no será utilizado por el programa mediante la operación delete.

```
delete(variable_puntero);
```

Solicitud/Liberación de memoria (2)

```
#include <iostream>
#include <stdlib.h>
using namespace std;
typedef int *p_entero;
main()
{ p_entero p;
  p=new int;
  if (p==NULL)
    cout << "No pudo reservarse memoria" << endl;
  else
   {  cout << "Direccion obtenida: " << p << endl;
      cout << "Ingrese valor: ";
      cin >> *p;
      cout << "Valor ingresado: " << *p << endl;
      delete(p);
      cout << "Memoria liberada " << endl;
    }
  system("pause");
}
```

```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Direccion obtenida: 0xce1240
Ingrese valor: 123
Valor ingresado: 123
Memoria liberada
Presione una tecla para continuar . . .
```

Reservando espacio de memoria para un entero

Verificación del puntero

Accediendo a la memoria reservada mediante p

Liberando memoria

Bibliografía

- Joyanes Aguilar *et al*. Estructuras de Datos en C++. Mc Graw Hill. 2007.
- De Giusti, Armando *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta. 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Hernández, Roberto *et al*. Estructuras de Datos y Algoritmos. Prentice Hall. 2001.