

Tablas de Símbolos

Programación de Sistemas de Telecomunicación Informática II

Departamento de Sistemas Telemáticos y Computación
(GSyC)

Universidad Rey Juan Carlos

Noviembre 2017



©2017 Grupo de Sistemas y Comunicaciones.

Algunos derechos reservados.

Este trabajo se distribuye bajo la licencia

Creative Commons Attribution Share-Alike

disponible en <http://creativecommons.org/licenses/by-sa/2.1/es>

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

Tablas de símbolos

- La tabla de símbolos es una estructura de datos también conocida por los siguientes nombres: **mapa**, **array asociativo**, **diccionario**.
- La **tabla de símbolos** es una estructura de datos que almacena elementos compuestos por parejas (**Clave**, **Valor**)
- **Clave** y **Valor** pueden ser tipos de datos cualesquiera
- Tiene tres operaciones básicas:
 - **Put**: Dado un nuevo elemento (**Clave**, **Valor**) como parámetro, se añade éste a la tabla. Si ya existía un elemento con la misma Clave, se sustituye su Valor asociado por el especificado en la llamada a Put
 - **Get**: Dada una Clave como parámetro, devuelve el Valor asociado a la misma en la tabla en caso de que exista un elemento (**Clave**, **Valor**)
 - **Delete**: Dada un Clave como parámetro, se borra de la tabla, si existe, el elemento (**Clave**, **Valor**)

Usos de las tablas de símbolos (I)

- Lista de usuarios en Mini-Chat:
(Clave => EP, Valor => Nickname)
- Lista de últimos mensajes vistos en Peer-Chat:
(Clave => EP, Valor => Número de secuencia)
- Listado de teléfonos:
(Clave => Nombre, Valor => Nº de teléfono)
- Diccionario:
(Clave => Palabra, Valor => Definición)
- DNS:
(Clave => Nombre de máquina, Valor => Dirección IP)

Usos de las tablas de símbolos (II)

- DNS inverso:
(Clave => Dirección IP, Valor => Nombre de máquina)
- Valoración bursátil:
(Clave => Valor bursátil, Valor => Cotización)
- Intercambio de ficheros P2P:
(Clave => Fichero, Valor => Máquina)
- Índice inverso de un libro:
(Clave => Vocablo, Value => Lista de números de página)
- Catálogo de buscador Web:
(Clave => Palabra, Value => Sitios web)

Especificación de la tabla de símbolos

```
with Ada.Strings.Unbounded;
package Maps is
    type Map is limited private;

    procedure Get (M      : Map;
                  Key    : in ASU.Unbounded_String;
                  Value  : out ASU.Unbounded_String;
                  Success : out Boolean);
    procedure Put (M      : in out Map;
                  Key    : ASU.Unbounded_String;
                  Value  : ASU.Unbounded_String);
    procedure Delete (M     : in out Map;
                      Key   : in Asu.Unbounded_String;
                      Success : out Boolean);
    function Map_Length (M : Map) return Natural;

    --
    -- Cursor Interface for iterating over Map elements
    --
    ...

private
    ...
end Maps;
```

Implementaciones de tablas de símbolos

- Mediante un Array no ordenado
- Mediante una Lista enlazada no ordenada
- Mediante un Array ordenado con búsqueda binaria
- Mediante una Lista enlazada ordenada
- Mediante un Árbol de búsqueda binaria (ABB)

Contenidos

- ① Tablas de Símbolos
- ② **Implementación de TS mediante un array no ordenado**
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

Implementación de TS mediante un array no ordenado

- Put, Get y Delete requieren una búsqueda lineal en el Array: en el peor caso hay que recorrer todos los elementos
- El Array tiene un tamaño máximo fijado de antemano

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ **Implementación de TS mediante una lista enlazada no ordenada**
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

Tipos de datos

```
package Maps is
    package ASU renames Ada.Strings.Unbounded;

    type Map is limited private;

    procedure Get (M      : Map;
                  Key    : in ASU.Unbounded_String;
                  Value  : out ASU.Unbounded_String;
                  Success : out Boolean);

private
    type Cell;
    type Cell_A is access Cell;
    type Cell is record
        Key   : ASU.Unbounded_String := ASU.Null_Unbounded_String;
        Value : ASU.Unbounded_String := ASU.Null_Unbounded_String;
        Next  : Cell_A;
    end record;
    type Map is record
        P_First : Cell_A;
        Length  : Natural := 0;
    end record;
end Maps;
```

Comparación con la implementación mediante un Array no ordenado

- La lista enlazada puede crecer / contraerse
- La búsqueda de un elemento (**Get**) no mejora respecto a la implementación con un Array no ordenado: hay que **buscar** linealmente la **Clave**
- La inserción de un elemento (**Put**) tampoco mejora, pues requiere **buscar** la **Clave**, ya que si existe hay que sustituir el valor almacenado por el nuevo
- El borrado de un elemento (**Delete**) tampoco mejora, pues, de nuevo, hay que **buscar** la **Clave**

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

1 Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
ASU.To_Unbounded_String("69.63.189.16"));

2 Maps.Get (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
Value, Success);

3 Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
ASU.To_Unbounded_String("66.249.92.104"));

4 Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
ASU.To_Unbounded_String("212.128.240.25"));

5 Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
ASU.To_Unbounded_String("69.63.189.11"));

6 Maps.Delete (A_Map, ASU.To_Unbounded_String("google.com"), Success);

7 Maps.Delete (A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),  
          ASU.To_Unbounded_String("69.63.189.16"));
```

A_Map.P_First



A_Map.Length

0

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length

0

Success

?

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length

0

Success

?

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length

0

Success

False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length

Success

0

False

Cell

Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success

0

False

Cell

Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success

0

False

cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length Success

0

False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success

0

False

Cell
Key
Value
Next

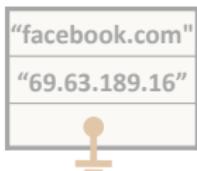
1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length

0

Success

False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First

P_Aux

```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length Success
0 False

Cell

Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

M.Length

0

Success

False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

M.Length

1

Success

False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First
P_Aux



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
1 False

Cell
Key
Value
Next

1

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.16"));
```

M.P_First

P_Aux




```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

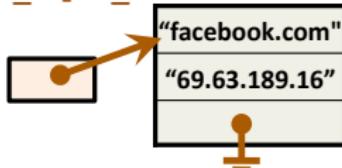
M.Length Success
1 False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),  
          Value, Success);
```

A_Map.P_First



A_Map.Length

1

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



	"facebook.com"
	"69.63.189.16"



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
  P_Aux : Cell_A;
begin
  Value := ASU.Null_Unbounded_String;

  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null Loop
    if P_Aux.Key = Key then
      Value := P_Aux.Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;
end Get;
```

M.Length

Value

1

Success

?

Cell

Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



"facebook.com"
"69.63.189.16"



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length

Value

1

Success

?

Cell

Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



"facebook.com"
"69.63.189.16"



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
  Value := ASU.Null_Unbounded_String;

  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null Loop
    if P_Aux.Key = Key then
      Value := P_Aux.Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;
end Get;
```

M.Length

Value

1

Null_Unbounded_String

Success

?

Cell

Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



```
procedure Get (M      : in out Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length

Value

1

Null Unbounded String

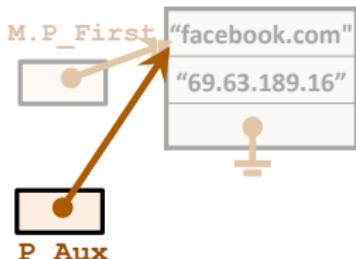
Success

?

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



```
procedure Get (M      : in out Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length Value

1

Null Unbounded String

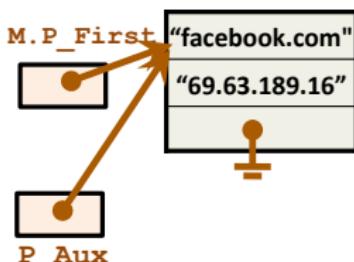
Success

False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;
    P_Aux := M.P.First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

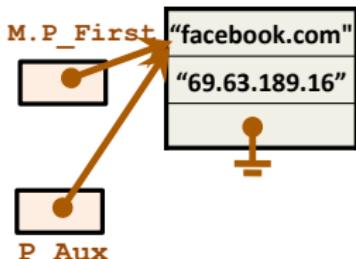
M.Length	Value
1	Null_Unbounded_String

Success
False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
  Value := ASU.Null_Unbounded_String;
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null Loop
    if P_Aux.Key = Key then
      Value := P_Aux.Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;
end Get;
```

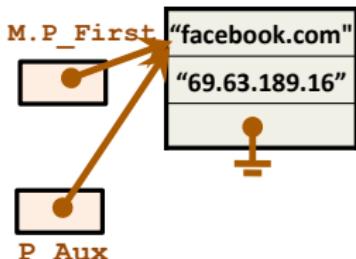
M.Length	Value
1	Null_Unbounded_String

Success
False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

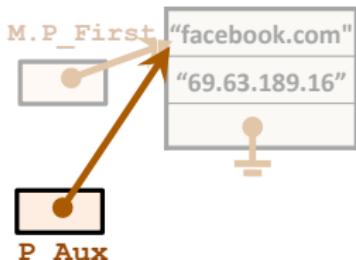
M.Length	Value
1	Null_Unbounded_String

Success
False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



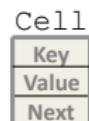
```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P.First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length

Value	Success
Null Unbounded String	False

1



2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

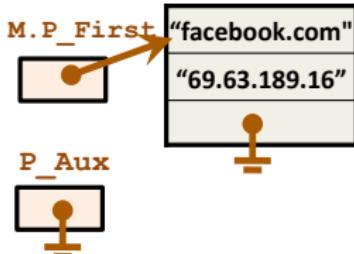
M.Length

M.Length	Value	Success
1	Null Unbounded String	False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length	Value
----------	-------

1

Null_Unbounded_String

Success

False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length Value

1

Null_Unbounded_String

Success

False

Cell
Key
Value
Next

2

```
Maps.Get(A_Map, ASU.To_Unbounded_String("www.urjc.es"),
          Value, Success);
```

M.P_First



```
procedure Get (M      : in out Map;
              Key    : in  ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success: out Boolean) is
    P_Aux : Cell_A;
begin
    Value := ASU.Null_Unbounded_String;

    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null Loop
        if P_Aux.Key = Key then
            Value := P_Aux.Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;
end Get;
```

M.Length

Value

1

Null_Unbounded_String

Success

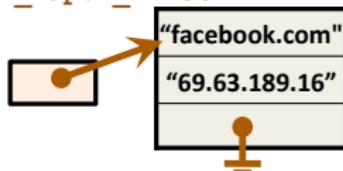
False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),  
           ASU.To_Unbounded_String("66.249.92.104"));
```

A_Map.P_First



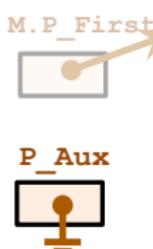
A_Map.Length

1

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
1 ?

Cell
Key
Value
Next

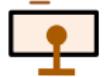
3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length

1

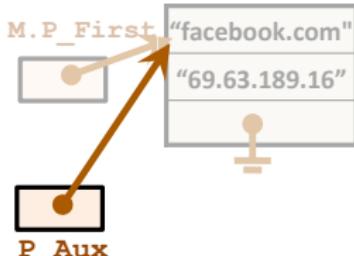
Success

?

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

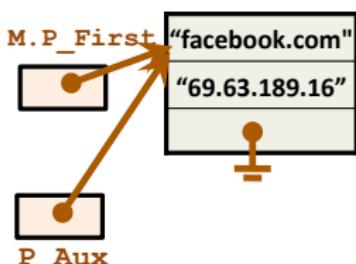
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
1	False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

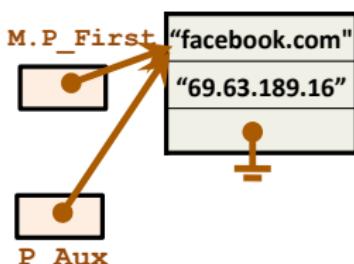
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
1 False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



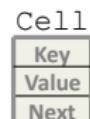
```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

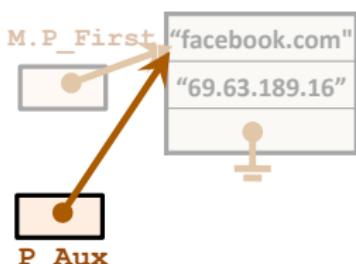
```

M.Length	Success
1	False



3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

M.Length	Success
1	False

Cell
Key
Value
Next

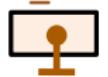
3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length

1

Success

False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First
P_Aux



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
1 False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First
P_Aux



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
1 False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First
P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

M.Length Success
1 False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

M.Length



Success

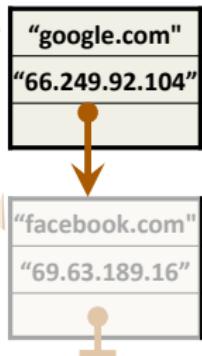


Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First



```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

M.Length Success

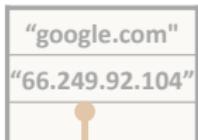
1

False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```

M.P_First

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

M.Length

1

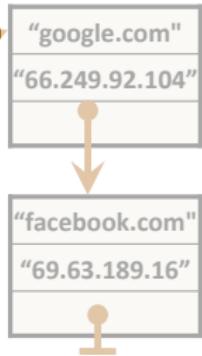
Success

False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



M.Length	Success
1	False

```

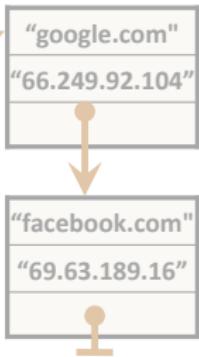
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

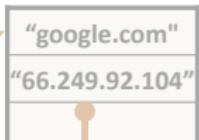
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
1	False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

M.Length

2

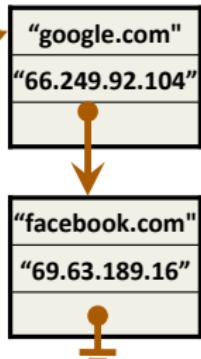
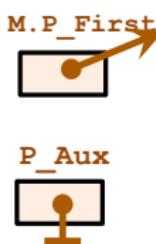
Success

False

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

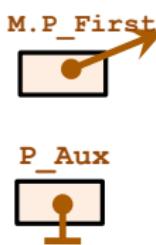
M.Length Success

2	False
---	-------

Cell
Key
Value
Next

3

```
Maps.Put (A_Map, ASU.To_Unbounded_String("google.com"),
           ASU.To_Unbounded_String("66.249.92.104"));
```



M.Length Success
2 False

```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

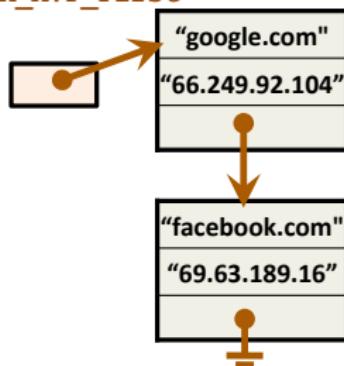
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),  
           ASU.To_Unbounded_String("212.128.240.25"));
```

A_M.P_First



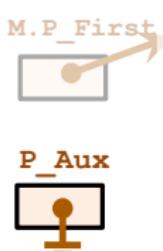
A_Map.Length

2

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length Success
2 ?

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



P_Aux



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

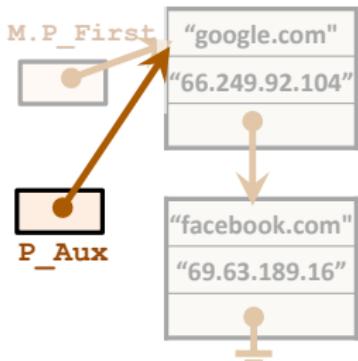
M.Length Success

2	?
---	---

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

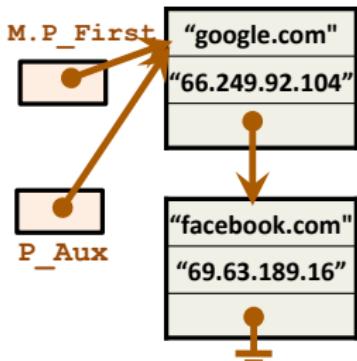
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

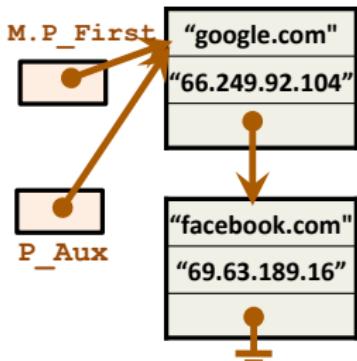
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

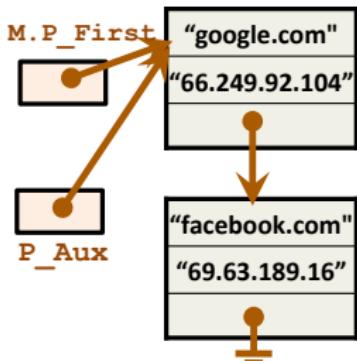
M.Length
2

Success
False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

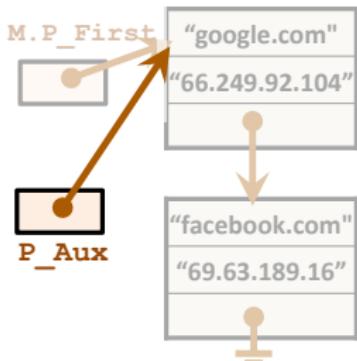
M.Length
2

Success
False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



M.Length	Success
2	False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

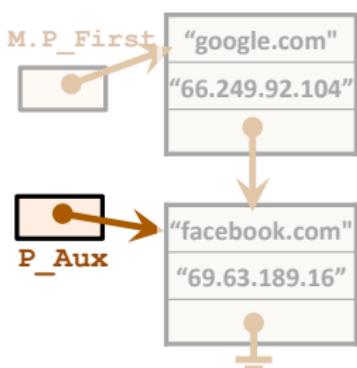
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

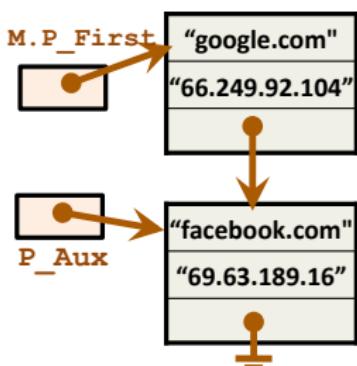
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

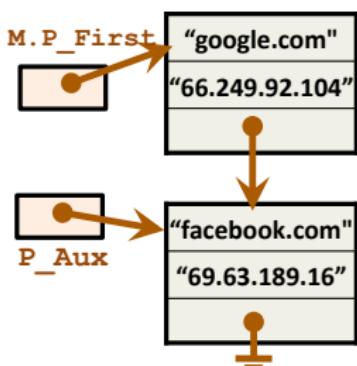
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

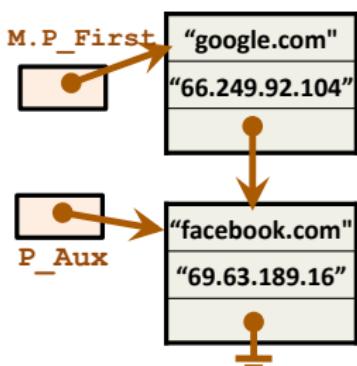
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

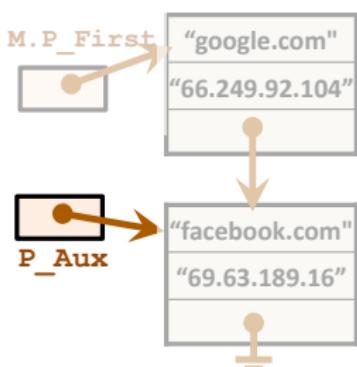
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



M.Length
2

Success
False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



P_Aux

```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

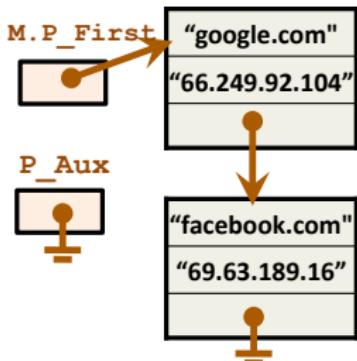
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



M.Length	Success
2	False

```

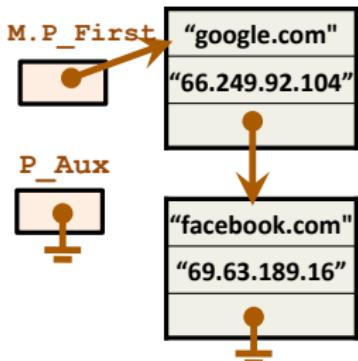
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



M.Length Success
2 False

```

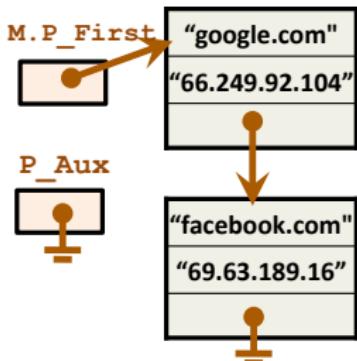
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

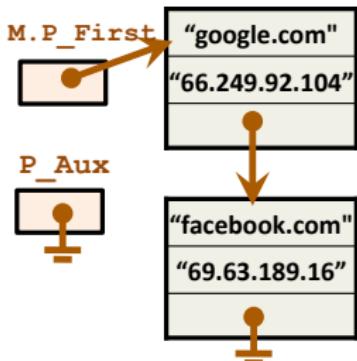
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```



```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

M.Length	Success
2	False

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



"www.urjc.es"

"212.128.240.25"

"google.com"

"66.249.92.104"

"facebook.com"

"69.63.189.16"

M.Length

2

Success

False

```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



	"www.urjc.es"
	"212.128.240.25"

	"google.com"
	"66.249.92.104"

	"facebook.com"
	"69.63.189.16"

M.Length

2

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

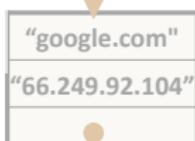
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



M.Length

2

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

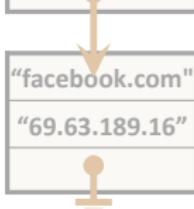
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



M.Length

2

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

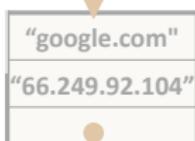
4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



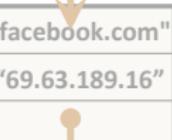
P_Aux



M.Length



3



False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



"www.urjc.es"

"212.128.240.25"



"google.com"
"66.249.92.104"

"facebook.com"
"69.63.189.16"

M.Length

3

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

4

```
Maps.Put (A_Map, ASU.To_Unbounded_String("www.urjc.es"),
           ASU.To_Unbounded_String("212.128.240.25"));
```

M.P_First



"www.urjc.es"

"212.128.240.25"



"google.com"

"66.249.92.104"

"facebook.com"

"69.63.189.16"

M.Length

3

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

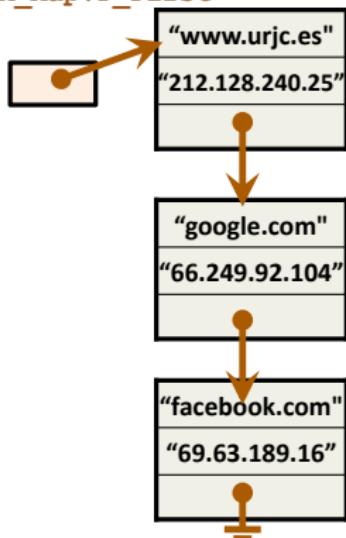
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),  
ASU.To_Unbounded_String("69.63.189.11"));
```

A_Map.P_First



A_Map.Length

3

Cell
Key
Value
Next

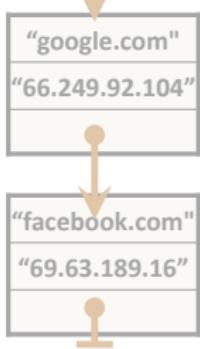
5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



P_Aux

P_Aux points to the previous node of the target key ("facebook.com").



M.Length

3

Success

?

```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

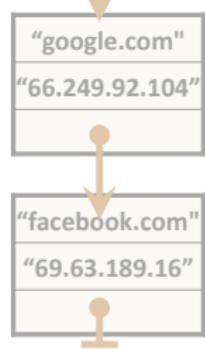
5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```



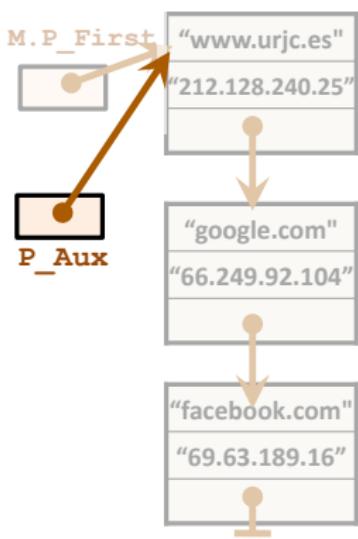
3

?

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



3 False

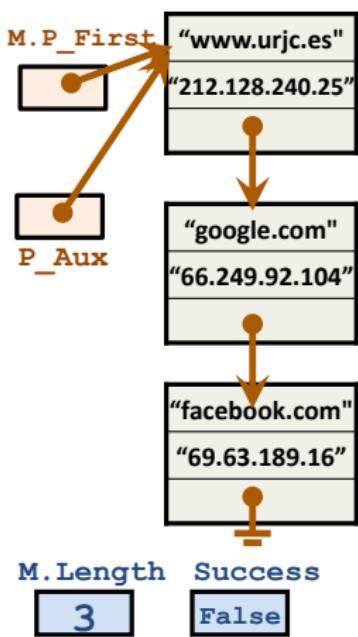
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

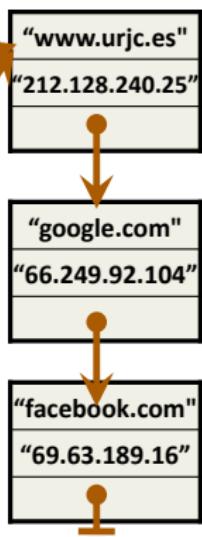
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```

M.P_First
P_Aux



M.Length

3

Success

False

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```

M.P_First
P_Aux

M.Length

3

Success

False

```

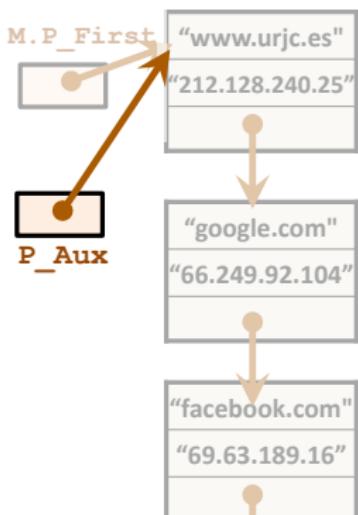
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length

3

Success

False

```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

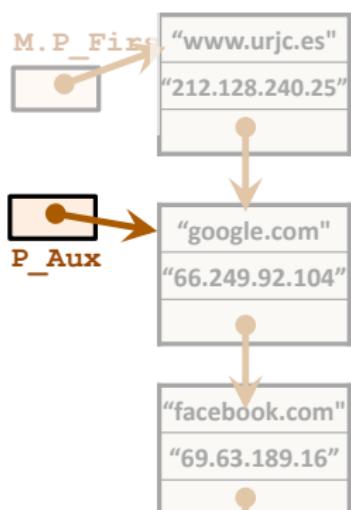
    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;

```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



3 False

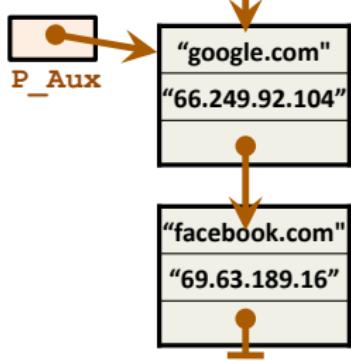
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



3

False

```

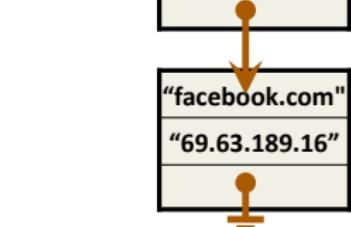
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



Success
False

```

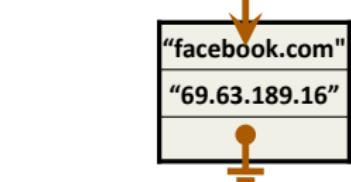
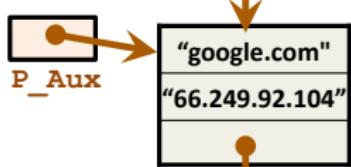
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length
3

Success
False

```

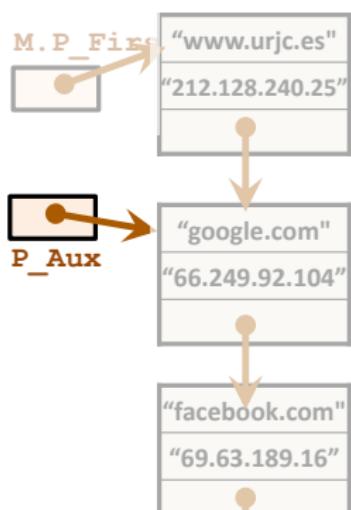
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length

3

Success

False

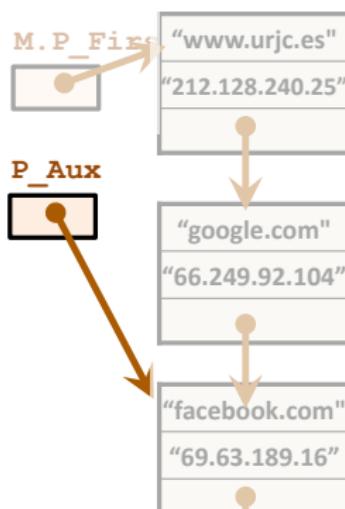
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



3 False

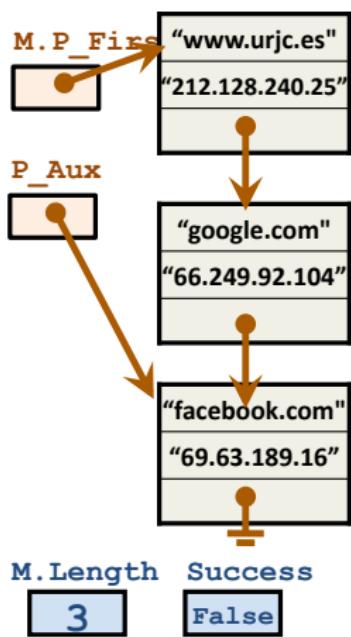
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



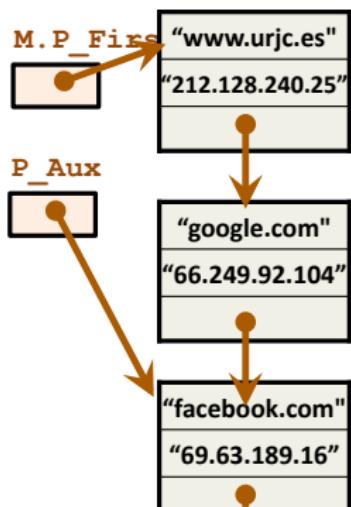
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



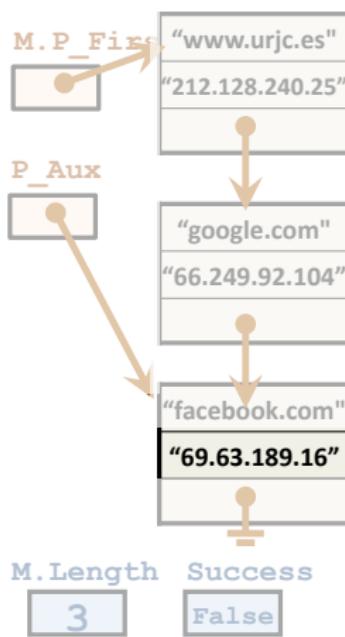
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



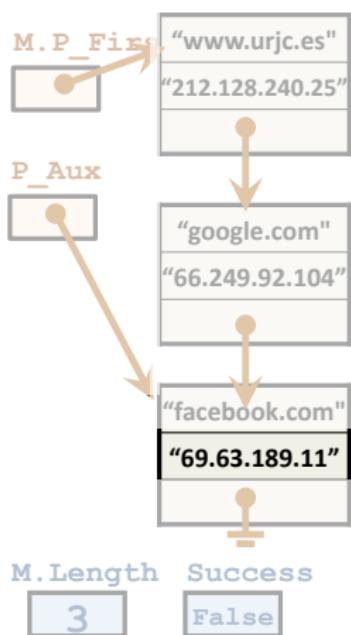
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



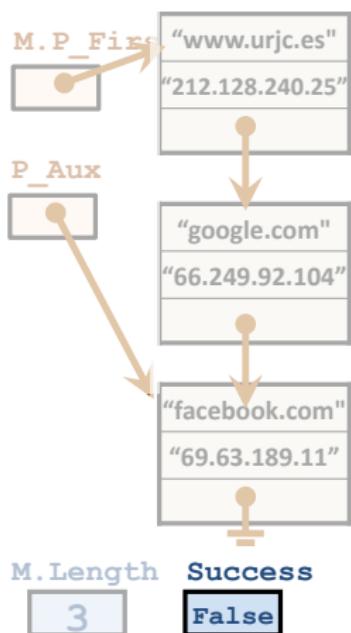
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

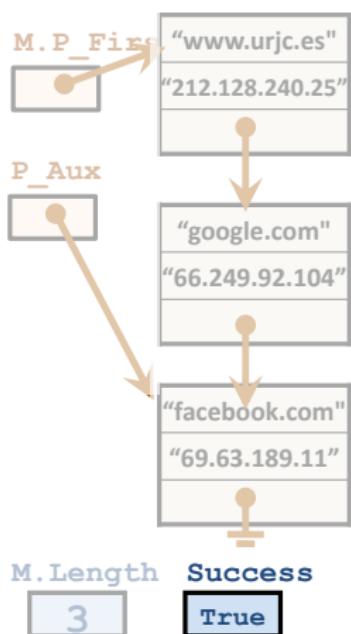
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



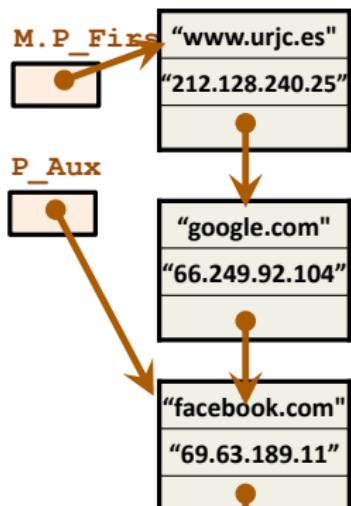
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



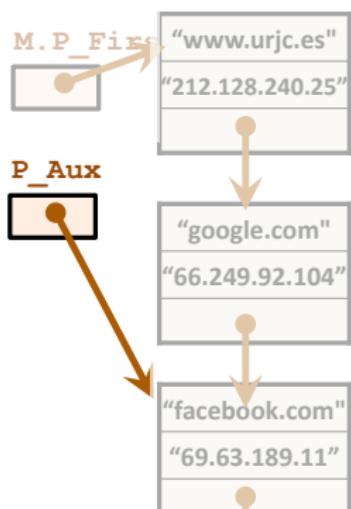
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length Success

3	True
---	------

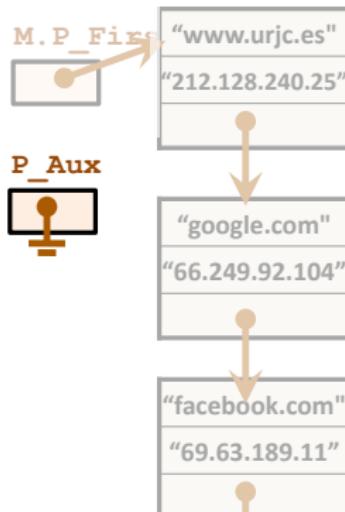
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



3 True

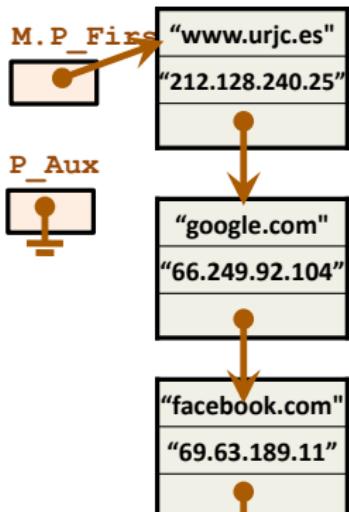
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



Success
True

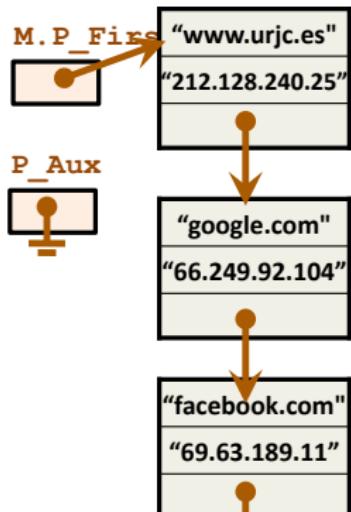
```
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



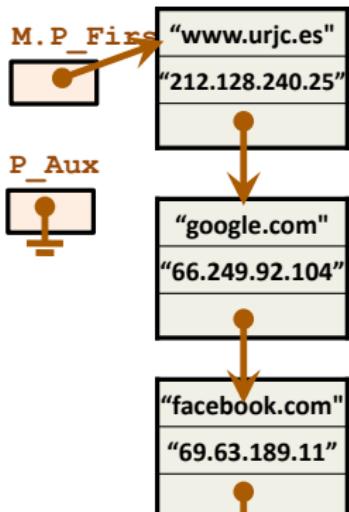
```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



```

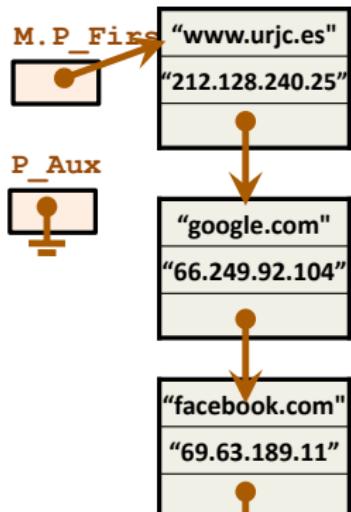
procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;
```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length

3

Success

True

```

procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
  P_Aux : Cell_A;
  Success : Boolean;
begin
  P_Aux := M.P_First;
  Success := False;
  while not Success and P_Aux /= null loop
    if P_Aux.Key = Key then
      P_Aux.Value := Value;
      Success := True;
    end if;
    P_Aux := P_Aux.Next;
  end loop;

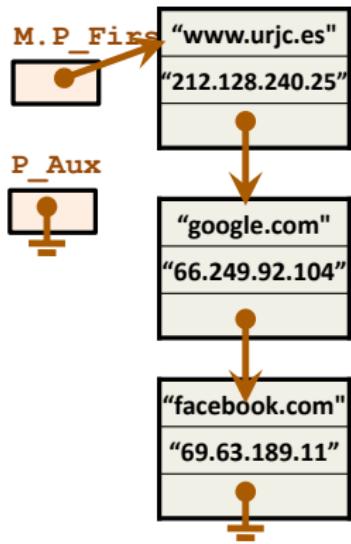
  if not Success then
    M.P_First:=new Cell'(Key, Value, M.P_First);
    M.Length := M.Length + 1;
  end if;
end Put;

```

Cell
Key
Value
Next

5

```
Maps.Put (A_Map, ASU.To_Unbounded_String("facebook.com"),
           ASU.To_Unbounded_String("69.63.189.11"));
```



M.Length Success
3 True

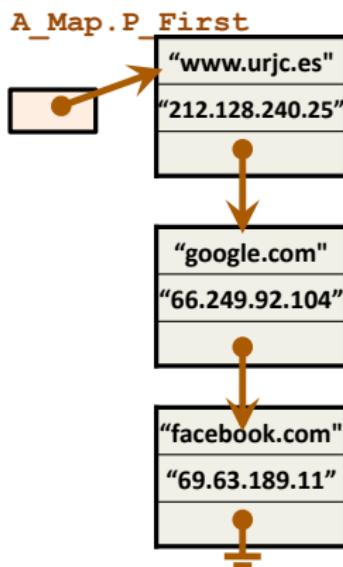
```

procedure Put (M      : in out Map;
              Key   : ASU.Unbounded_String;
              Value : ASU.Unbounded_String) is
    P_Aux : Cell_A;
    Success : Boolean;
begin
    P_Aux := M.P_First;
    Success := False;
    while not Success and P_Aux /= null loop
        if P_Aux.Key = Key then
            P_Aux.Value := Value;
            Success := True;
        end if;
        P_Aux := P_Aux.Next;
    end loop;

    if not Success then
        M.P_First:=new Cell'(Key, Value, M.P_First);
        M.Length := M.Length + 1;
    end if;
end Put;
```

6

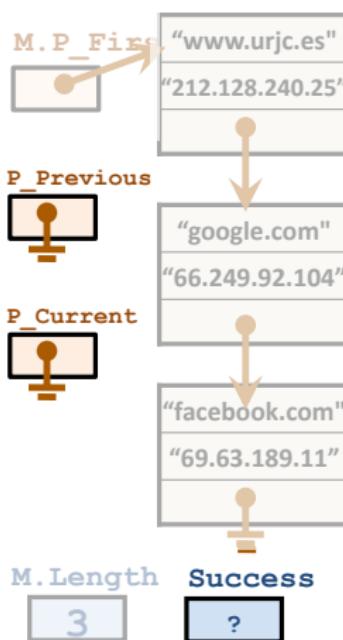
```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```



A_Map.Length

3

6



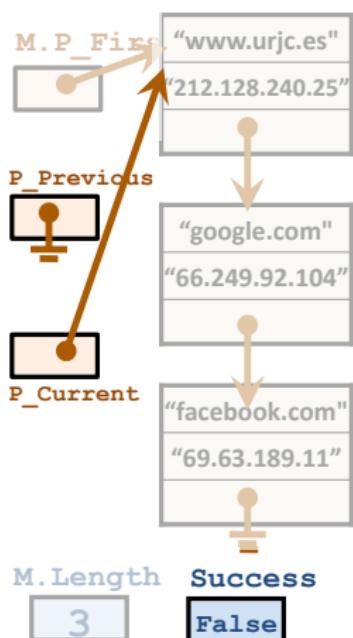
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success : out Boolean) is
    P_Current : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6

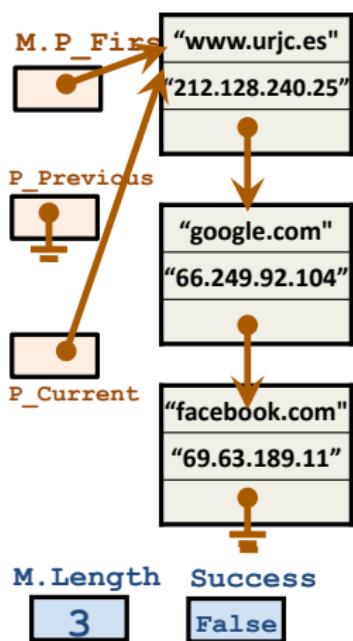


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("google.com"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

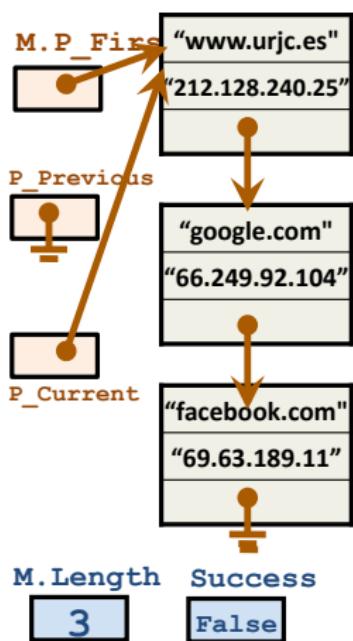
6



```
Maps.Delete(A_Map, ASU.To_Unbounded_String("google.com"), Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



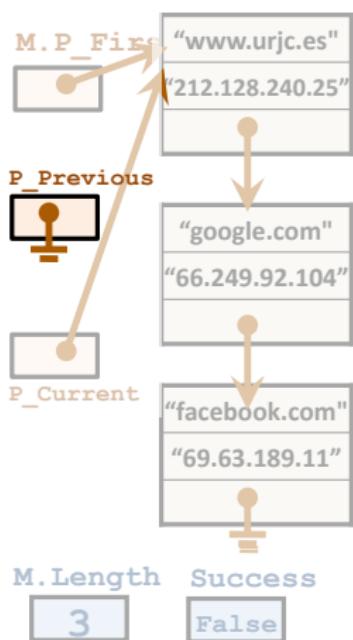
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6

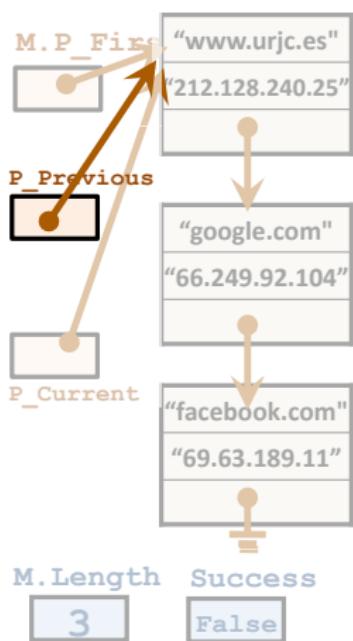


```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6

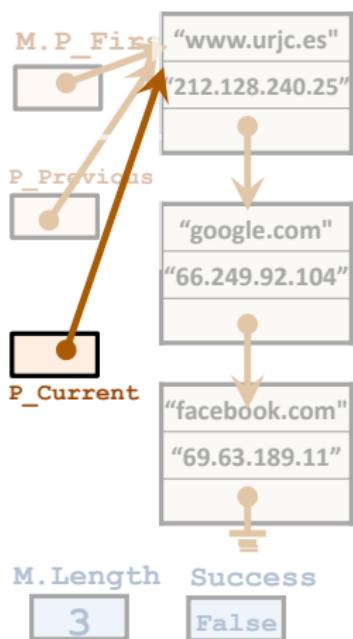


```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6

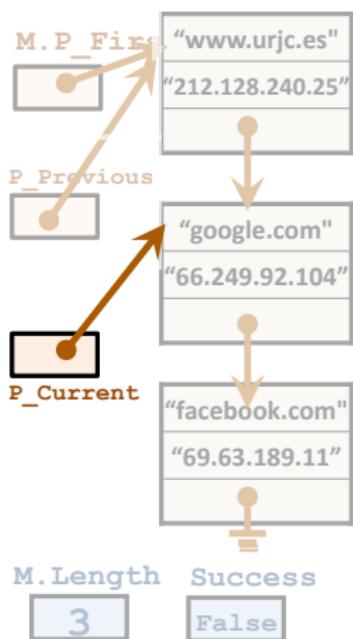


```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

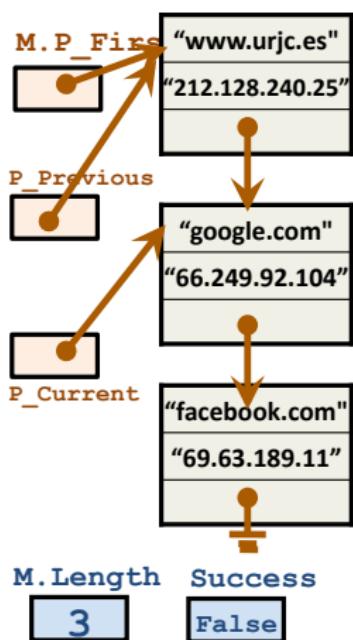
6



```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6

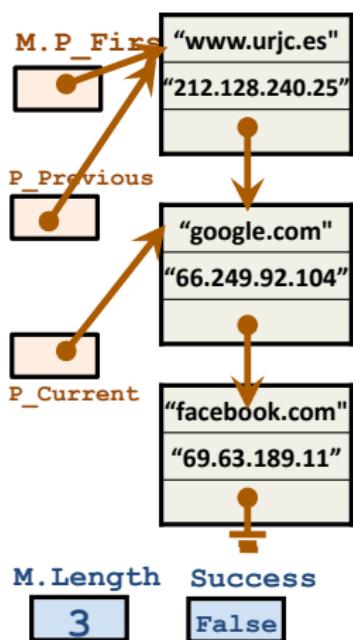


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("google.com"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



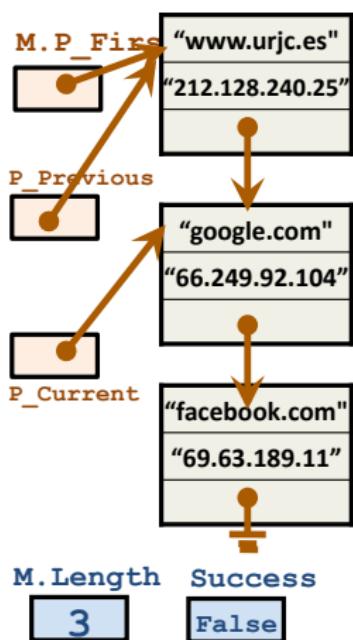
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6



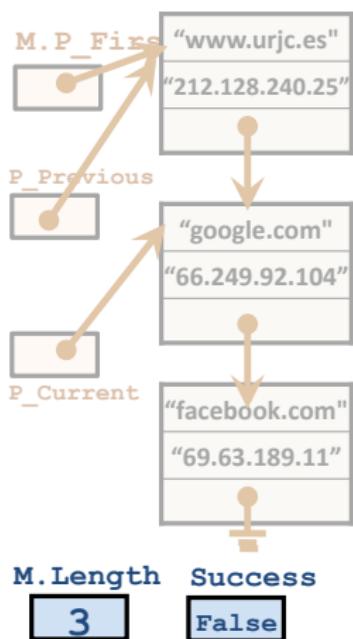
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6

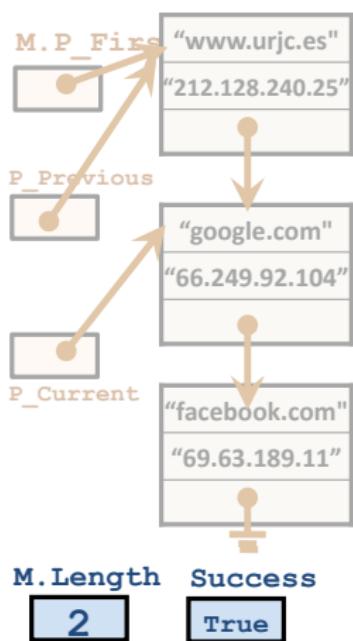


```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



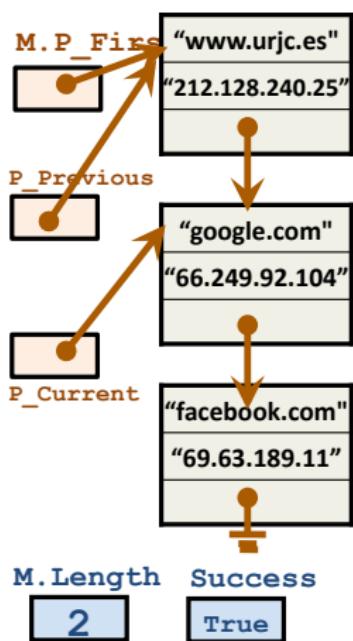
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

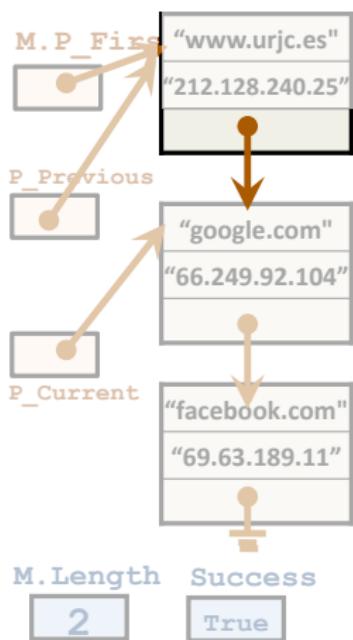
6



```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



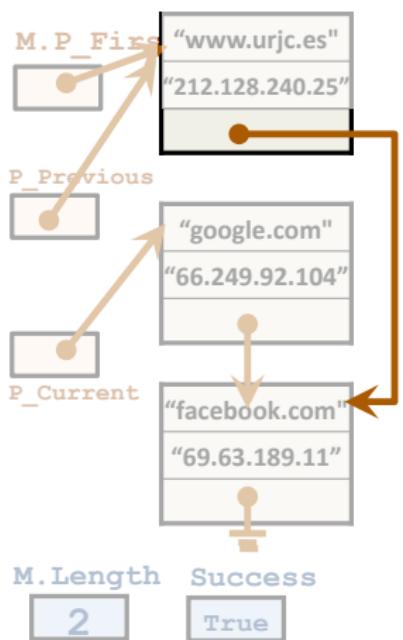
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

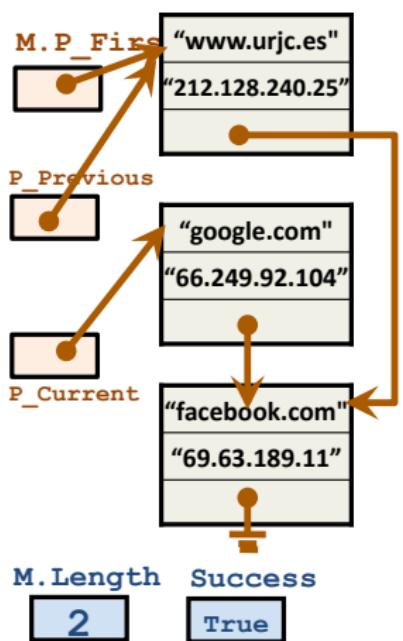
6



```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



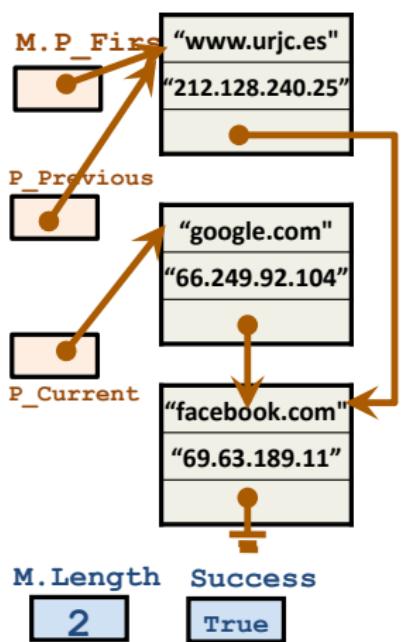
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6



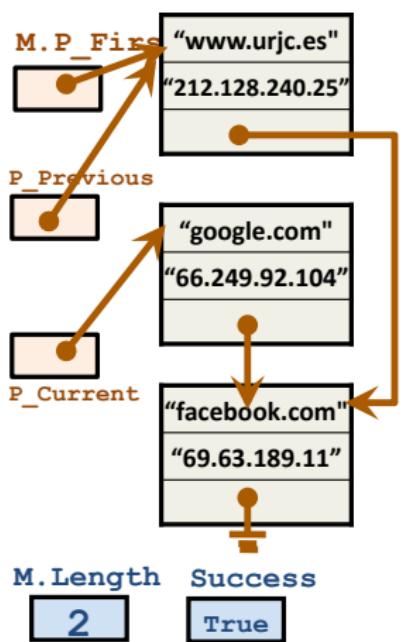
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6



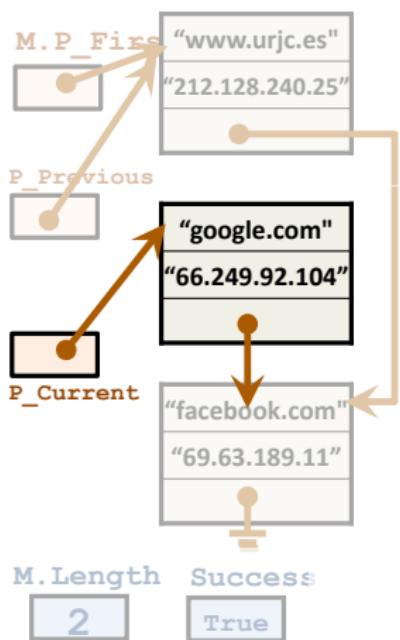
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

6



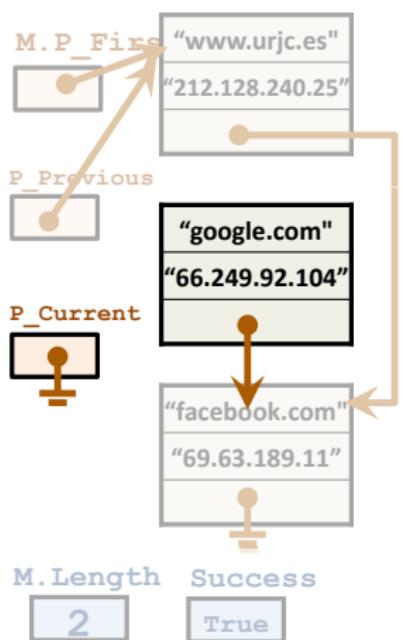
```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

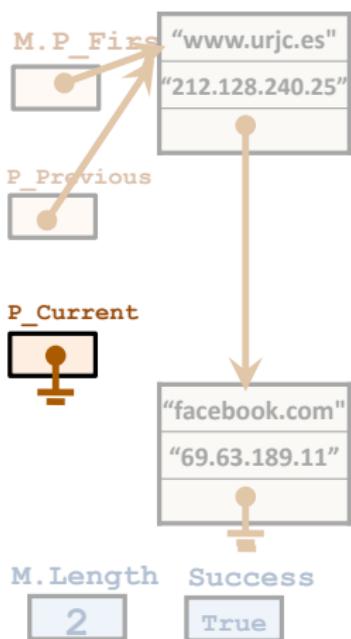
6



```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6



```

Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

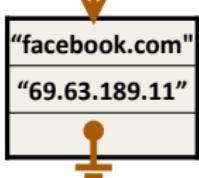
```

6

```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```



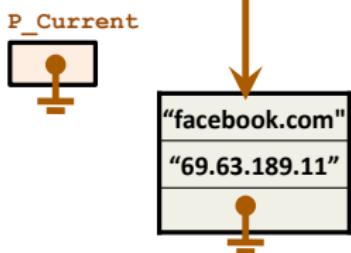
M.Length Success
2 True

6

```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```



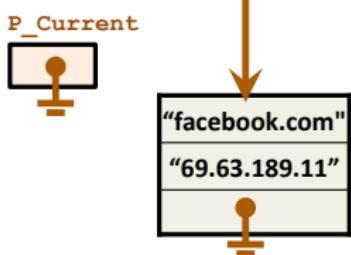
M.Length Success
2 True

6

```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

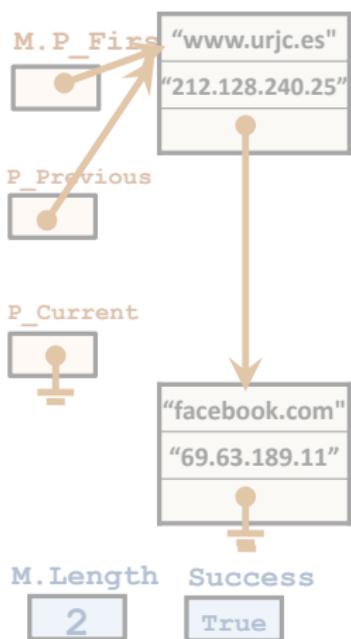
```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```



M.Length Success
2 True

6



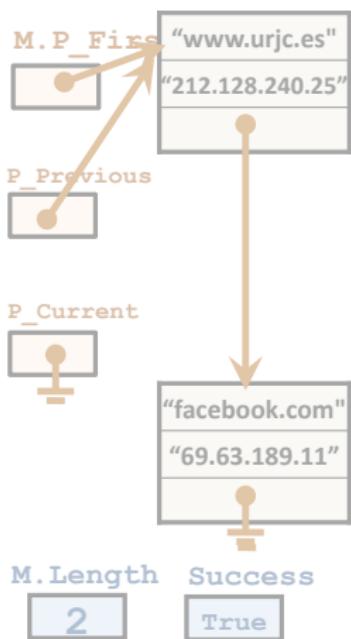
```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

6

```
Maps.Delete(A_Map,ASU.To_Unbounded_String("google.com"),Success);
```



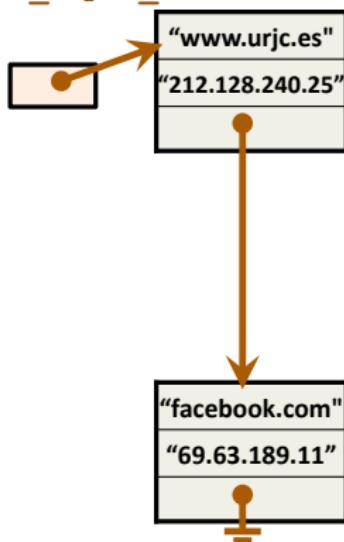
```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

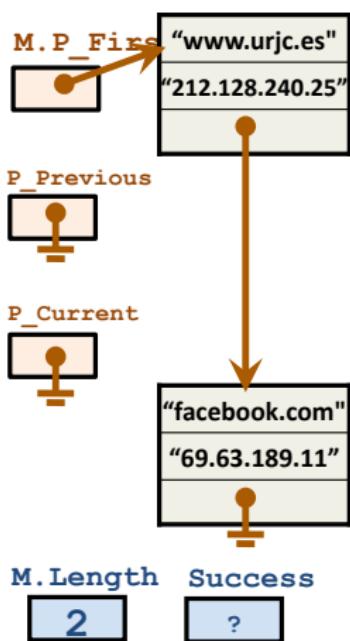
A_Map.P_First



A_Map.Length

2

7

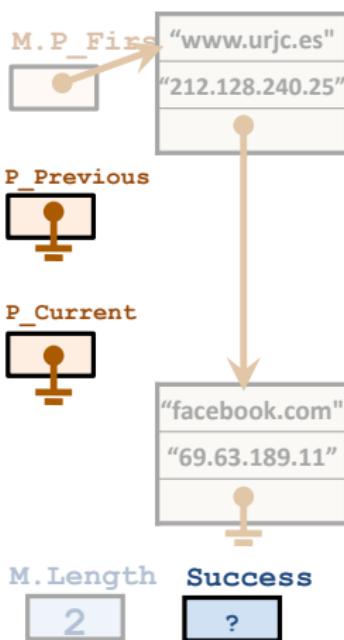


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

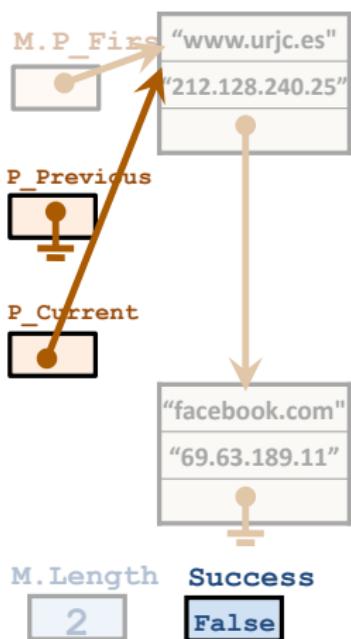


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

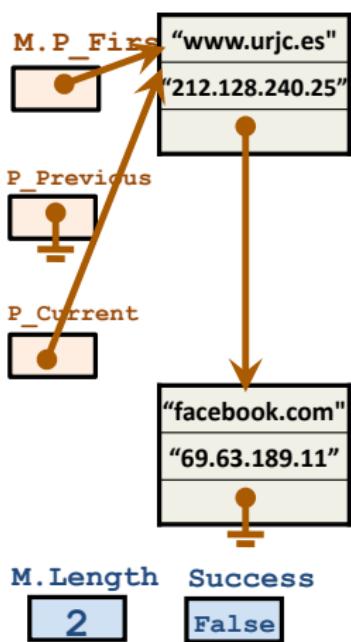


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7



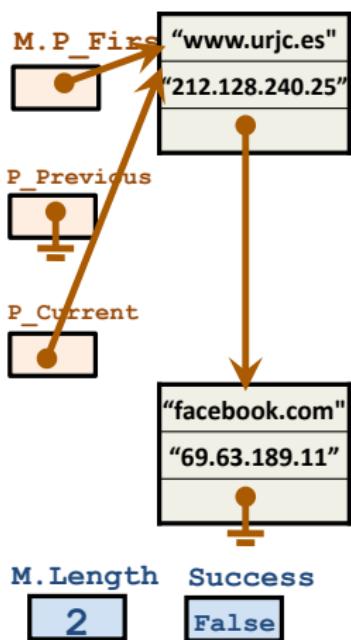
```

Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

7

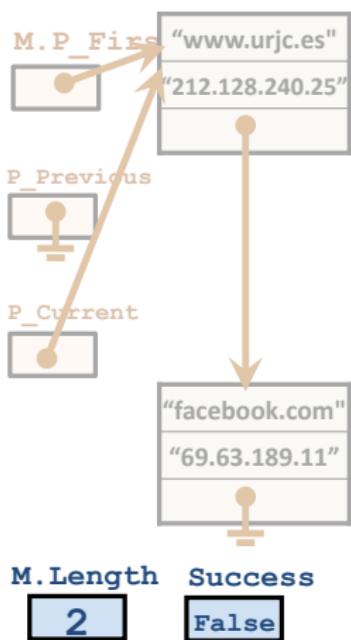


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key    : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

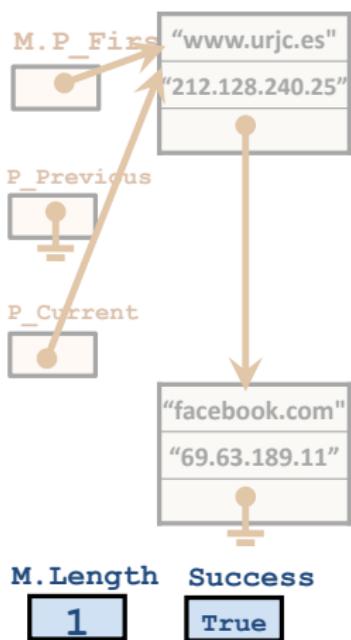


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

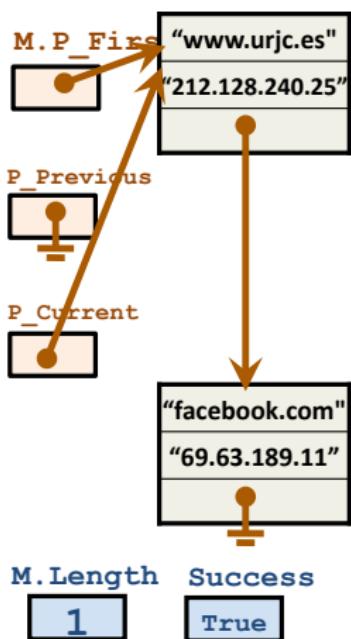


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

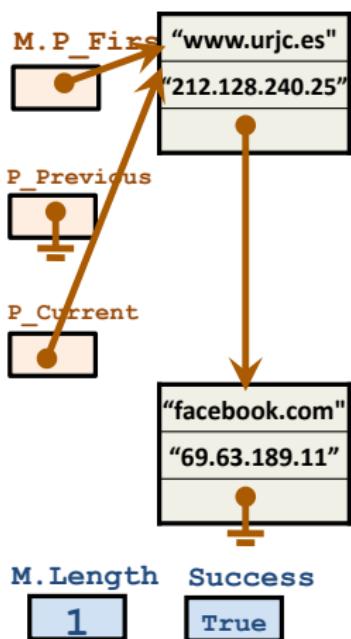


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

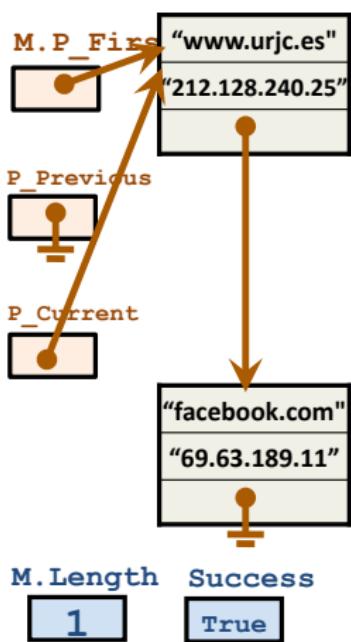


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

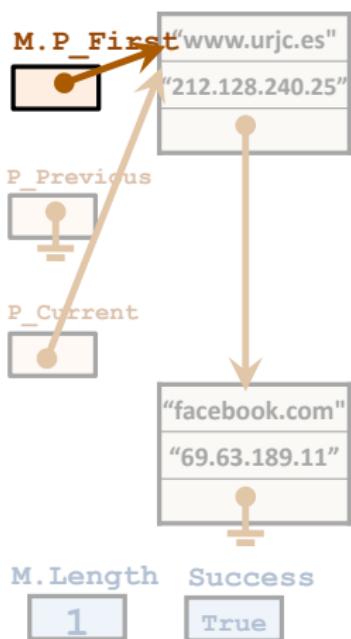


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

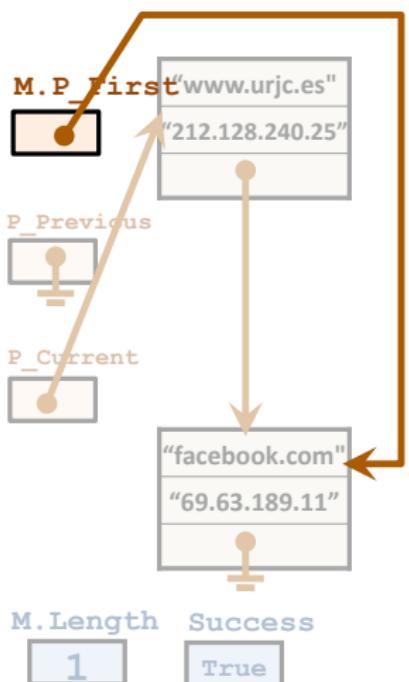


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

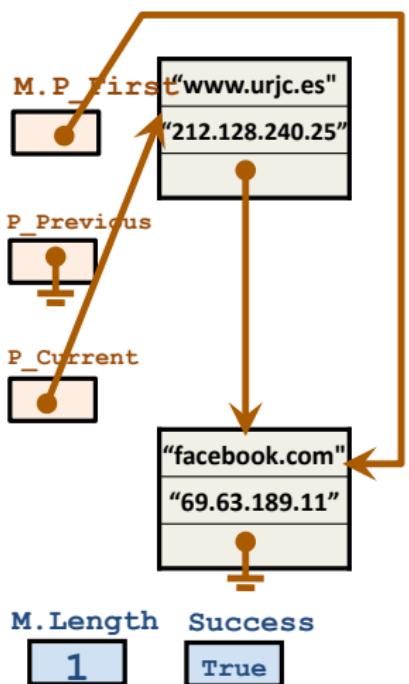


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

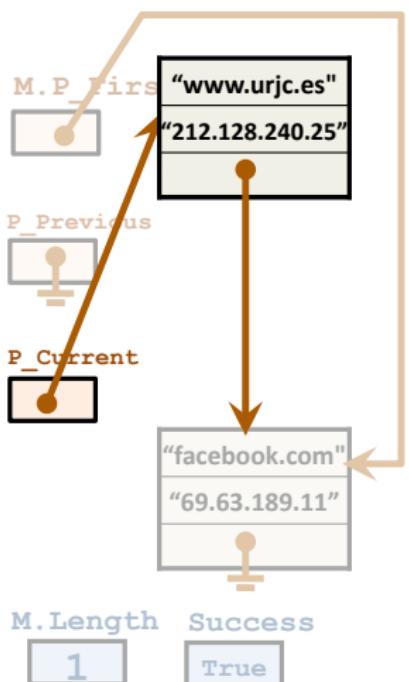


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

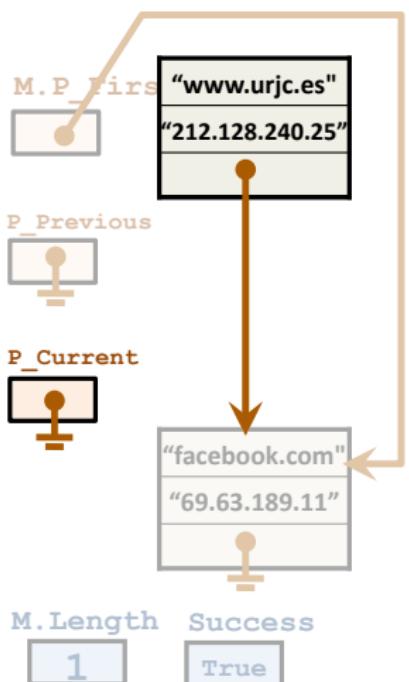


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

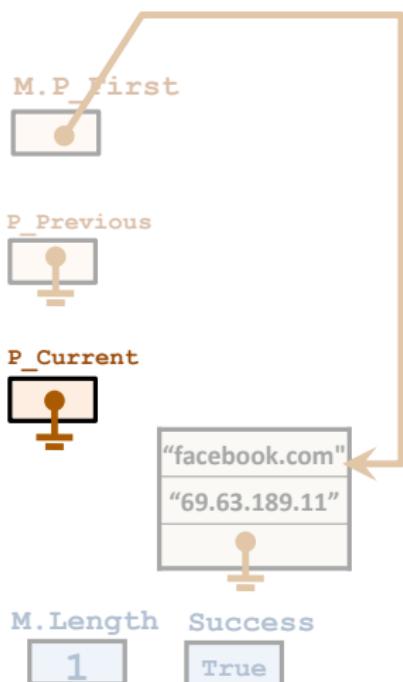


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7

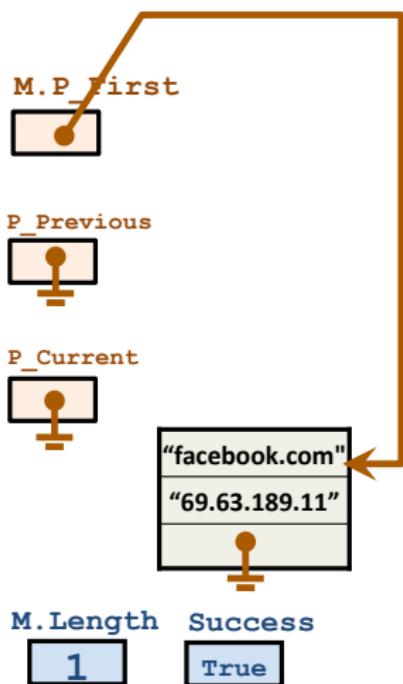


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7



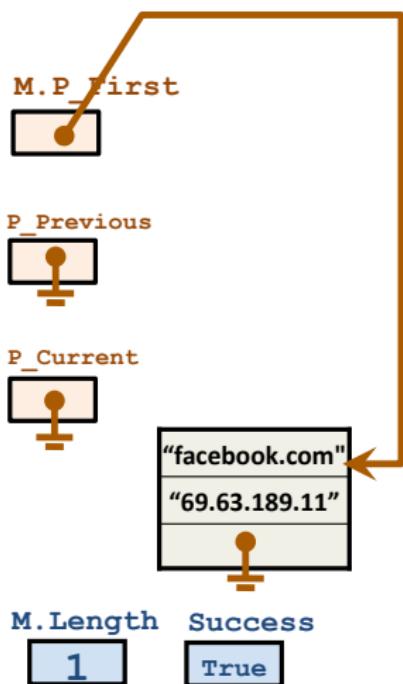
```

Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

7



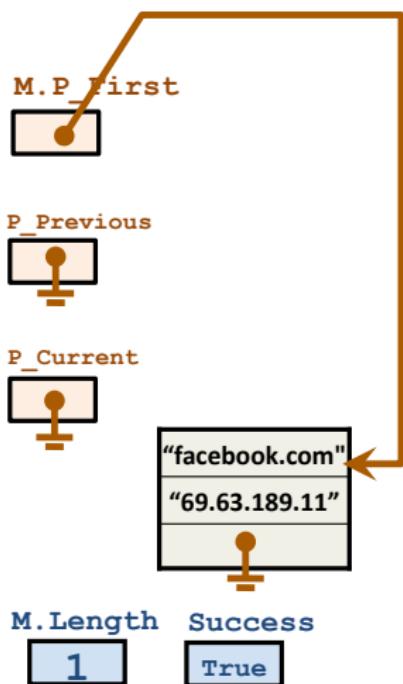
```

Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);

procedure Delete (M      : in out Map;
                  Key   : in Asu.Unbounded_String;
                  Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

7

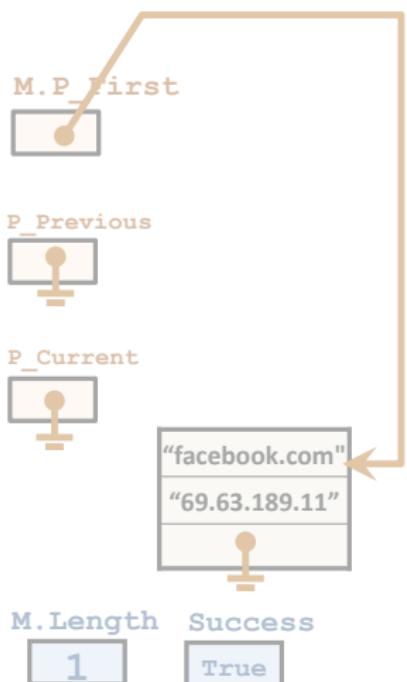


```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

7

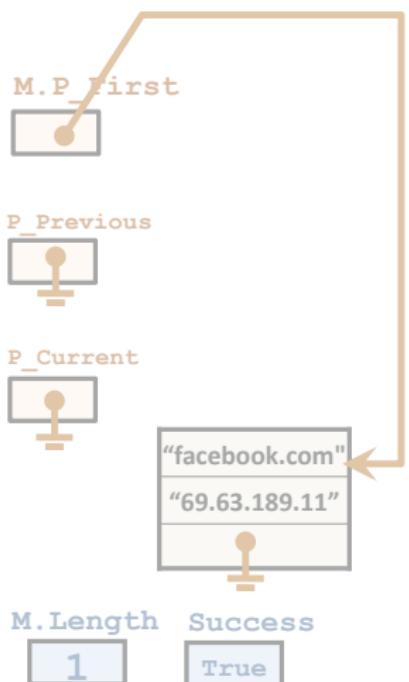


```
Maps.Delete(A_Map, ASU.To_Unbounded_String("www.urjc.es"), Success);
```

```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;
```

7



```

procedure Delete (M      : in out Map;
                 Key   : in Asu.Unbounded_String;
                 Success: out Boolean) is
    P_Current  : Cell_A;
    P_Previous : Cell_A;
begin
    Success := False;
    P_Previous := null;
    P_Current  := M.P_First;
    while not Success and P_Current /= null loop
        if P_Current.Key = Key then
            Success := True;
            M.Length := M.Length - 1;
            if P_Previous /= null then
                P_Previous.Next := P_Current.Next;
            end if;
            if M.P_First = P_Current then
                M.P_First := M.P_First.Next;
            end if;
            -- Liberar si no hay Garbage Collector
            P_Current:=null;
        else
            P_Previous := P_Current;
            P_Current := P_Current.Next;
        end if;
    end loop;
end Delete;

```

Contenidos

- 1 Tablas de Símbolos
- 2 Implementación de TS mediante un array no ordenado
- 3 Implementación de TS mediante una lista enlazada no ordenada
- 4 Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- 5 Iteración sobre todos los elementos de una colección
- 6 Implementación de TS mediante un Array ordenado
- 7 Implementación de TS mediante una lista enlazada ordenada
- 8 Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- 9 Ejemplo de ejecución: Get en un ABB
- 10 Ejemplo de ejecución: Put en un ABB vacío
- 11 Ejemplo de ejecución: Put en un ABB
- 12 Borrado de un nodo en un ABB

Especificación de iteradores para la tabla de símbolos

```
with Ada.Strings.Unbounded;
package Maps is
...
-- Cursor Interface for iterating over Map elements
--
type Cursor is limited private;
function First (M: Map) return Cursor;
procedure Next (C: in out Cursor);
function Has_Element (C: Cursor) return Boolean;
type Element_Type is record
    Key: ASU.Unbounded_String;
    Value: ASU.Unbounded_String;
end record;
No_Element: exception;
-- Raises No_Element if Has_Element(C) = False;
function Element (C: Cursor) return Element_Type;
private
... // Suponiendo una implementación mediante lista enlazada
type Cursor is record
    M          : Map;
    Element_A : Cell_A;
end record;
```

Ejemplo de uso del iterador

```
with Maps;
procedure Map_Test is

procedure Print_Map (M : Maps.Map) is
C: Maps.Cursor
begin
Ada.Text_IO.Put_Line ("Map");
Ada.Text_IO.Put_Line ("====");

C := Maps.First(M);
while Maps.Has_Element(C) loop
Ada.Text_IO.Put_Line (ASU.To_String(Maps.Element(C).Key) &
                      " " &
                      ASU.To_String(Maps.Element(C).Value));
Maps.Next(C);
end loop;
end Print_Map;
...

begin
...
end Map_Test;
```

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ **Implementación de TS mediante un Array ordenado**
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

Tabla de símbolos implementada mediante un Array ordenado

- Se utiliza también un Array, por lo que tenemos un tamaño máximo fijado de antemano
- Los elementos de la tabla se mantienen ordenados y contiguos en el array
- La ventaja ahora es que la búsqueda de un elemento NO requiere recorrer todos los elementos del Array hasta encontrar el que se busca. En su lugar, se realiza una **búsqueda binaria**.
- La inserción de un nuevo elemento se realiza usando la búsqueda binaria para encontrar la posición que le corresponde al elemento que se inserta
 - Si la clave del elemento a insertar no está ya en el Array, hay que mover todos los elementos que le suceden una posición hacia adelante para hacer hueco para el que se inserta
 - Esta operación es costosa porque implica no sólo recorrer todos los elementos mayores sino también copiar cada uno de ellos.

Búsqueda binaria en un Array ordenado

- La búsqueda binaria es menos costosa que la búsqueda lineal en un Array ordenada ya que, comparado con la implementación que usa un Array no ordenado, requiere menos accesos al Array para localizar el elemento
- Se compara el elemento a buscar con el elemento que está en el medio del Array:
 - si es menor, se busca con el mismo procedimiento en el subArray a la izquierda
 - en caso contrario, se busca en el subArray a la derecha

Contenidos

- 1 Tablas de Símbolos
- 2 Implementación de TS mediante un array no ordenado
- 3 Implementación de TS mediante una lista enlazada no ordenada
- 4 Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- 5 Iteración sobre todos los elementos de una colección
- 6 Implementación de TS mediante un Array ordenado
- 7 Implementación de TS mediante una lista enlazada ordenada**
- 8 Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- 9 Ejemplo de ejecución: Get en un ABB
- 10 Ejemplo de ejecución: Put en un ABB vacío
- 11 Ejemplo de ejecución: Put en un ABB
- 12 Borrado de un nodo en un ABB

Implementación de TS mediante una lista enlazada ordenada

- Al igual que con un Array ordenado, mantenemos ordenados los elementos según su clave
- Ventaja: la operación de inserción no requiere mover los elementos que le suceden
- Inconveniente: la operación de búsqueda sin embargo no se puede implementar tan eficientemente como en el Array ordenado: no podemos ir al elemento que está en la mitad de la lista por lo que la búsqueda ha de ser lineal y no binaria

Contenidos

- 1 Tablas de Símbolos
- 2 Implementación de TS mediante un array no ordenado
- 3 Implementación de TS mediante una lista enlazada no ordenada
- 4 Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- 5 Iteración sobre todos los elementos de una colección
- 6 Implementación de TS mediante un Array ordenado
- 7 Implementación de TS mediante una lista enlazada ordenada
- 8 **Implementación de TS mediante un árbol de búsqueda binaria (ABB)**
- 9 Ejemplo de ejecución: Get en un ABB
- 10 Ejemplo de ejecución: Put en un ABB vacío
- 11 Ejemplo de ejecución: Put en un ABB
- 12 Borrado de un nodo en un ABB

Tabla de símbolos implementada mediante un ABB

- En lugar de utilizar un Array o una Lista enlazada utilizaremos ahora un **árbol de búsqueda binaria (ABB)** para implementar la misma estructura de datos: una tabla de símbolos
- Recordatorio:
 - La **tabla de símbolos** es una estructura de datos que almacena elementos compuestos por parejas (**Clave, Valor**)
 - **Clave** y **Valor** pueden ser tipos de datos cualesquiera
 - Tiene tres operaciones básicas:
 - **Put**: Dado un nuevo elemento (**Clave, Valor**) como parámetro, se añade éste a la tabla. Si ya existía un elemento con la misma Clave, se sustituye su Valor asociado por el especificado en la llamada a Put
 - **Get**: Dada una Clave como parámetro, devuelve el Valor asociado a la misma en la tabla en caso de que exista un elemento (**Clave, Valor**)
 - **Delete**: Dada un Clave como parámetro, se borra de la tabla, si existe, el elemento (**Clave, Valor**)

Especificación de la tabla de símbolos

La especificación no cambia, salvo por la parte privada:

```
with Ada.Strings.Unbounded;
package Maps is
    package ASU renames Ada.Strings.Unbounded;

        type Map is limited private;
        procedure Get (M          : Map;
                      Key       : in  ASU.Unbounded_String;
                      Value     : out ASU.Unbounded_String;
                      Success   : out Boolean);
        ...
private
    type Tree_Node;
    type Map is access Tree_Node;
    type Tree_Node is record
        Key   : ASU.Unbounded_String := ASU.Null_Unbounded_String;
        Value : ASU.Unbounded_String := ASU.Null_Unbounded_String;
        Left  : Map;
        Right : Map;
    end record;
end Maps;
```

Árbol de búsqueda binaria (ABB)

Principal característica de un ABB

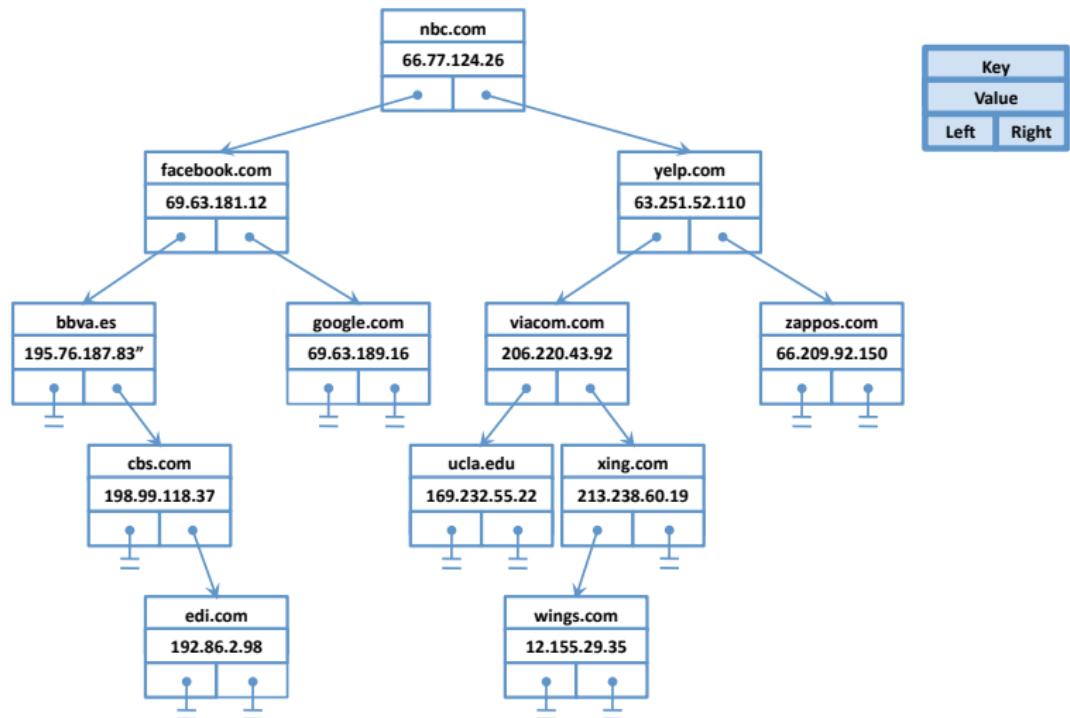
- La información que está almacenada en el árbol está **ordenada**
- Gracias a ello la búsqueda de un elemento en la tabla implementada con un ABB a partir de su clave NO requiere recorrer todos los nodos: **búsqueda binaria**
- Cada nodo de un árbol de búsqueda binaria almacena una clave con su valor asociado: (**Key, Value**)
- Los nodos del árbol están **ordenados** por su clave ⇒
 - las claves tienen que ser de tipos que tengan definidos los operadores de comparación $<$, $>$, $=$, $/$ =
- El valor de cada nodo puede ser de cualquier tipo
 - Incluyendo tipos compuestos (arrays, listas,...)

Definición recursiva del árbol de búsqueda binaria (ABB)

Un árbol de búsqueda binaria (ABB):

- o está vacío
- o está formado por:
 - Una pareja (**Key, Value**)
 - Un **ABB izquierdo con claves menores que Key** (subárbol izquierdo)
 - Un **ABB derecho con claves mayores que Key** (subárbol derecho)

Definición recursiva del ABB: ejemplo



Key	
Value	
Left	
Left	Right

Definición recursiva del árbol de búsqueda binaria (ABB): código

```
type Tree_Node;
type Map is access Tree_Node;
type Tree_Node is record
    Key   : ASU.Unbounded_String := ASU.Null_Unbounded_String;
    Value : ASU.Unbounded_String := ASU.Null_Unbounded_String;
    Left  : Map;
    Right : Map;
end record;
```

Definición recursiva del árbol de búsqueda binaria (ABB)

- Para identificar el árbol utilizamos la dirección en la que está su **nodo raíz**
- Llamamos **subárbol izquierdo** de un nodo i al árbol cuya raíz está apuntada por $i.Left$
 - Llamamos **subárbol derecho** de un nodo i al árbol cuya raíz está apuntada por $i.Right$
- Un nodo j es **hijo** de un nodo **padre** i si j es la raíz de uno de los dos subárboles de i
- La definición recursiva del árbol permite definir operaciones recursivas de manera elegante.
 - También una lista enlazada se puede definir recursivamente (o vacía, o formada por un primer nodo y una lista con el resto de los elementos)
 - Se pueden también definir las operaciones de la lista enlazada recursivamente

Búsqueda de un nodo

Para buscar un nodo con determinada clave *Key* en un árbol del que conocemos su nodo raíz (*Root*):

- Si el árbol está vacío \Rightarrow no existe el elemento
- Si el árbol no está vacío \Rightarrow
 - Si $Key = Root.Key \Rightarrow$ el nodo raíz es el nodo buscado
 - Si $Key < Root.Key \Rightarrow$ buscar el nodo en el **subárbol izquierdo**
 - Si $Key > Root.Key \Rightarrow$ buscar el nodo en el **subárbol derecho**

Búsqueda de un nodo: código

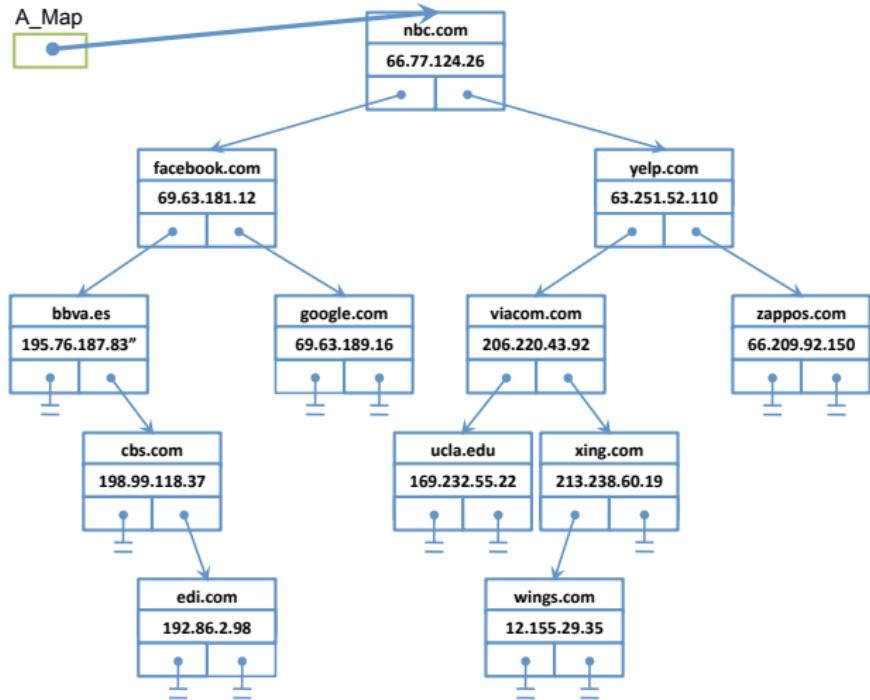
En el primer parámetro, **M**, se le pasa al procedimiento **Get** un puntero al nodo raíz del árbol:

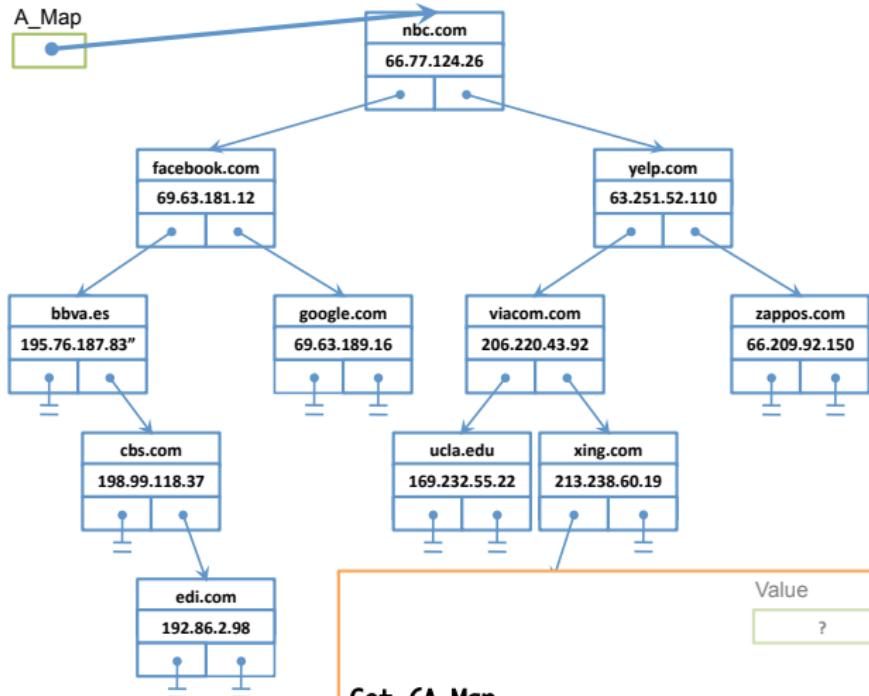
```
procedure Get (M : Map;
               Key      : in ASU.Unbounded_String;
               Value    : out ASU.Unbounded_String;
               Success  : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

  If M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;
```

Contenidos

- 1 Tablas de Símbolos
- 2 Implementación de TS mediante un array no ordenado
- 3 Implementación de TS mediante una lista enlazada no ordenada
- 4 Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- 5 Iteración sobre todos los elementos de una colección
- 6 Implementación de TS mediante un Array ordenado
- 7 Implementación de TS mediante una lista enlazada ordenada
- 8 Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- 9 Ejemplo de ejecución: Get en un ABB**
- 10 Ejemplo de ejecución: Put en un ABB vacío
- 11 Ejemplo de ejecución: Put en un ABB
- 12 Borrado de un nodo en un ABB

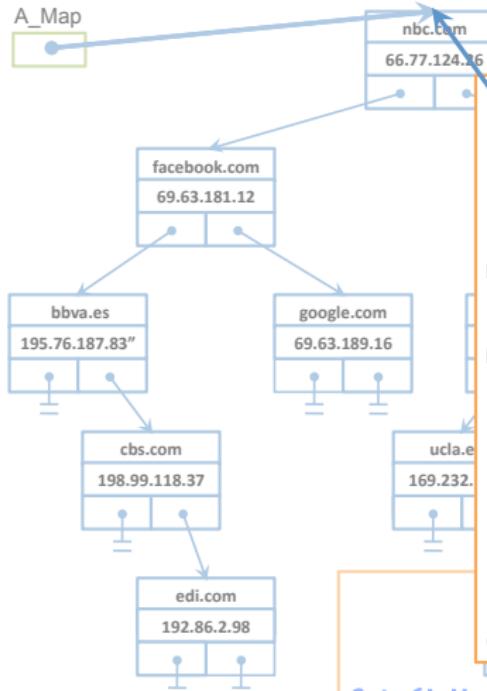




```

Get (A_Map,
ASU.To_Unbounded_String ("google.com"),
Value,
Success);
  
```

Value	Success
?	?



M	Key	Value	Success
blue dot	google.com	?	?

```

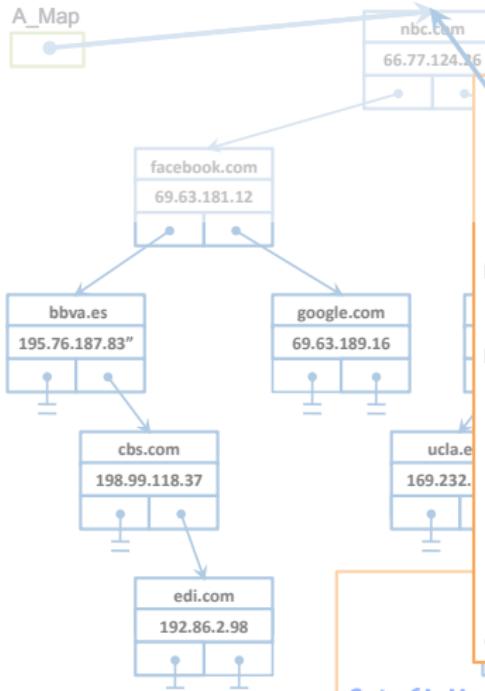
procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
	google.com	Null...String	?

```

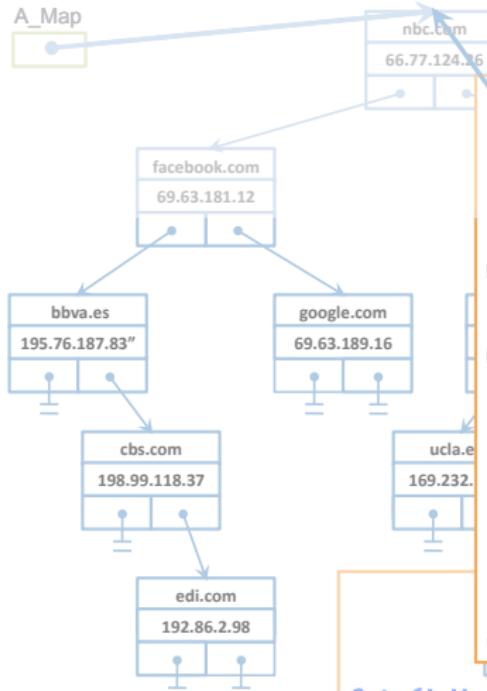
procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
google.com	Null...String	?	

```

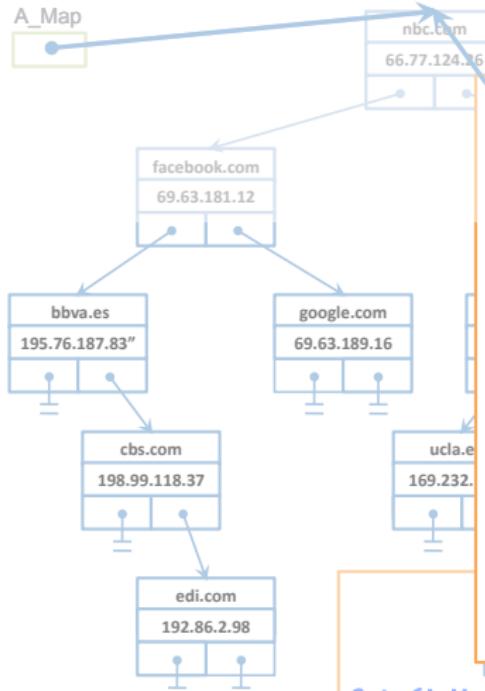
procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
	google.com	Null...String	?

```

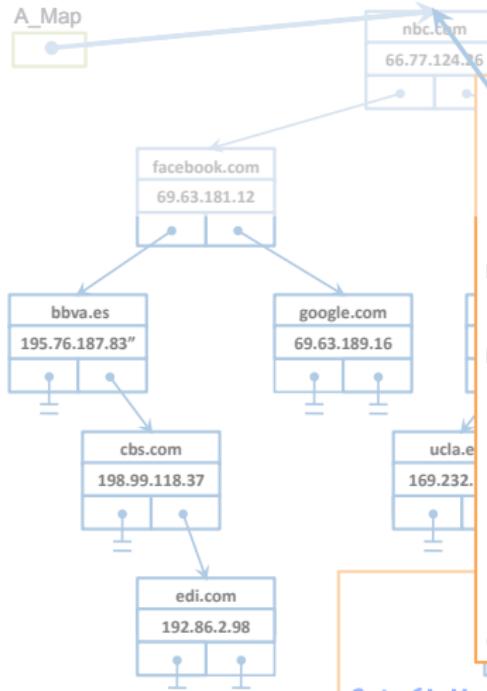
procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
	google.com	Null...String	?

```

procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

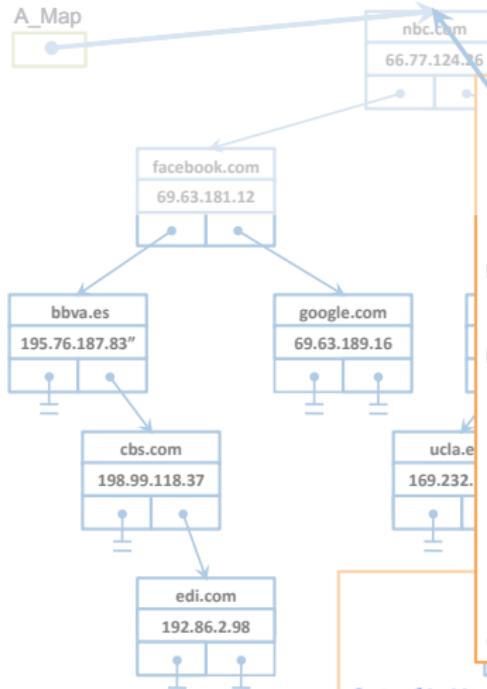
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
google.com	Null_..String	?	

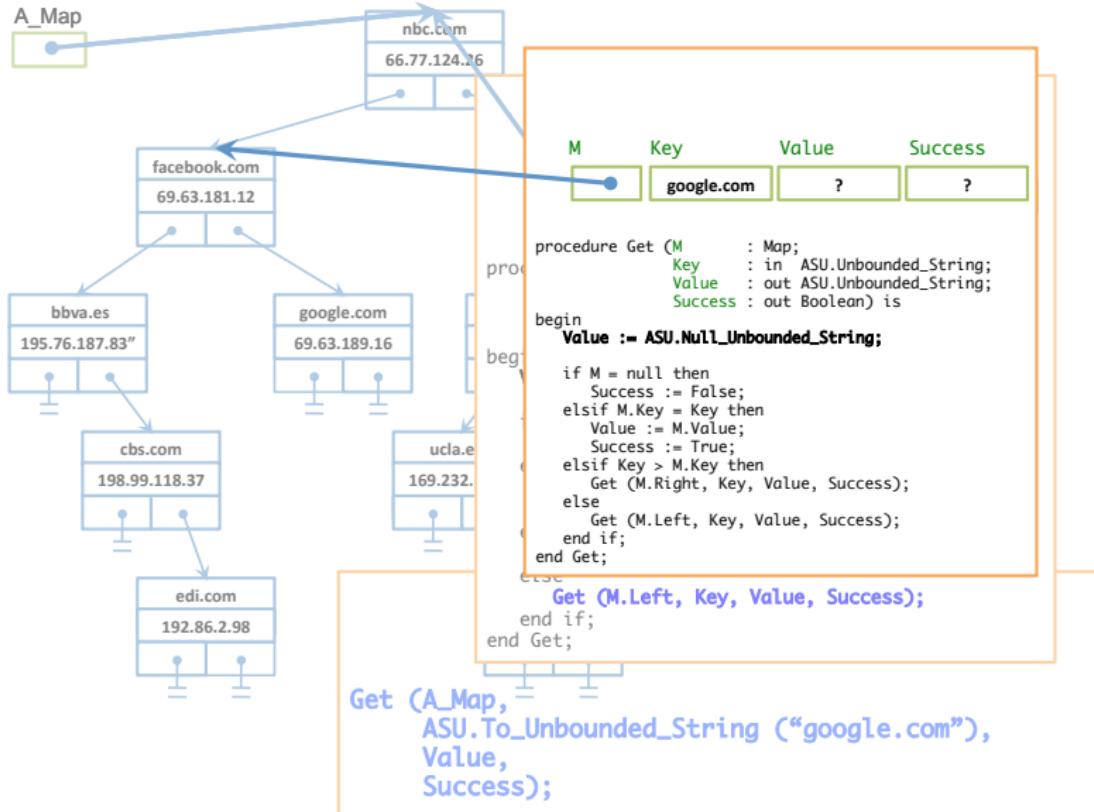
```

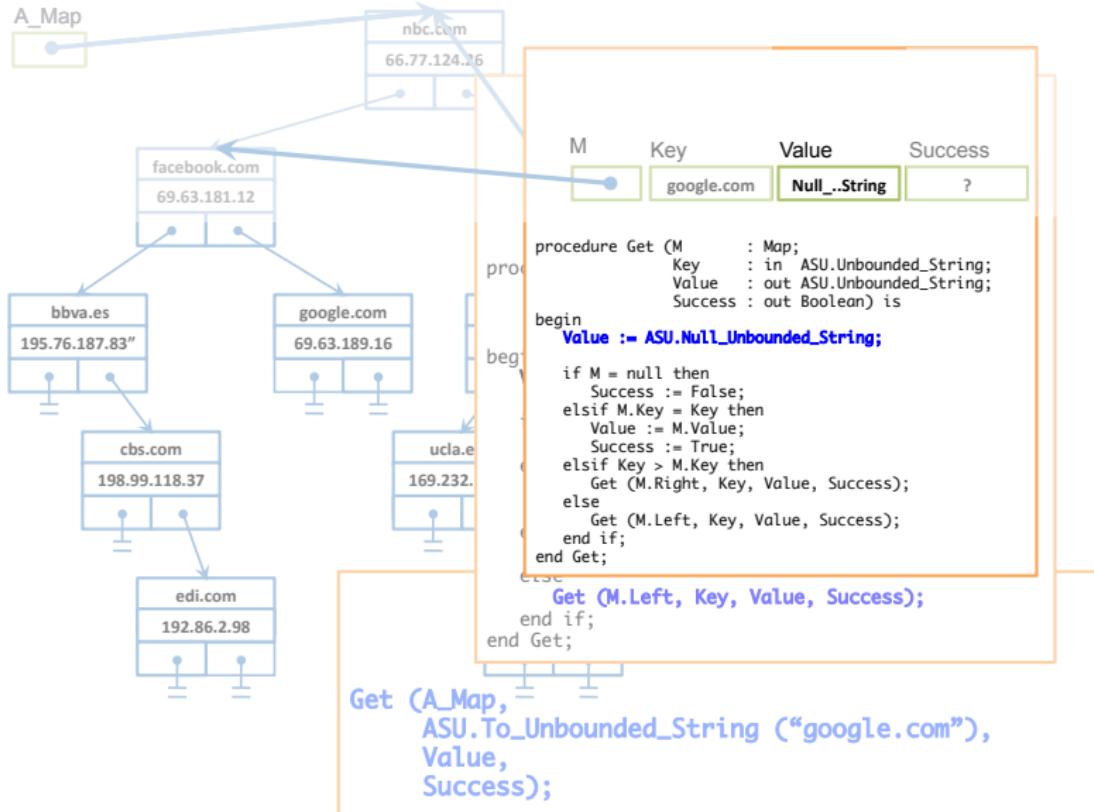
procedure Get (M      : Map;
              Key     : in ASU.Unbounded_String;
              Value   : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

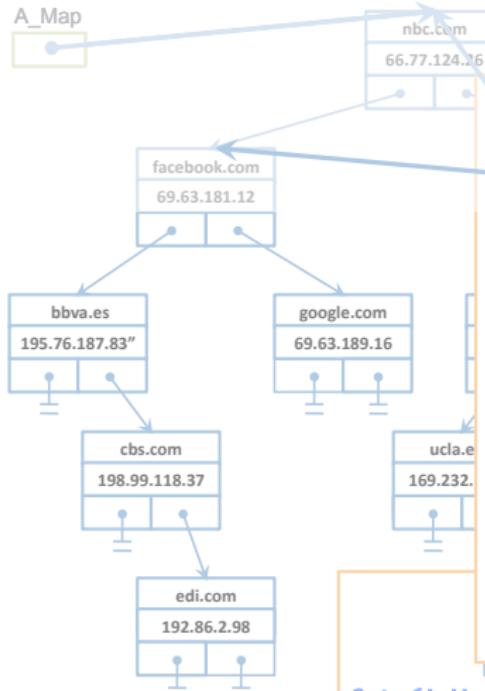
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;
  
```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```





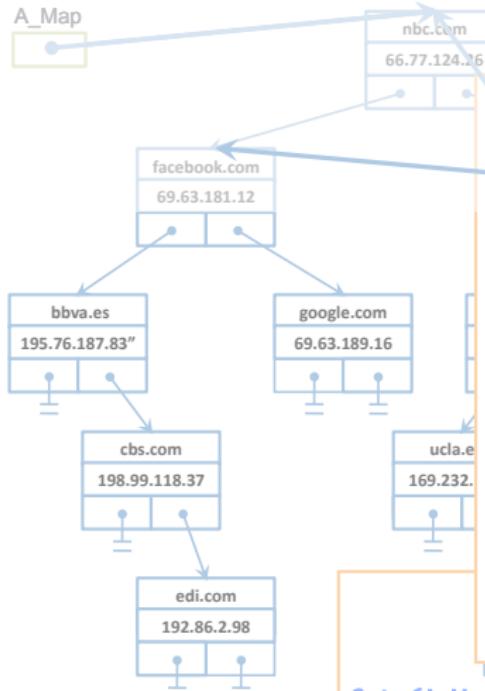


```

procedure Get (M      : Map;
              Key   : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;
  
```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```



M	Key	Value	Success
<code>ASU.To_Unbounded_String ("google.com")</code>		<code>Null_.String</code>	?

```

procedure Get (M      : Map;
              Key   : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is

```

```

begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

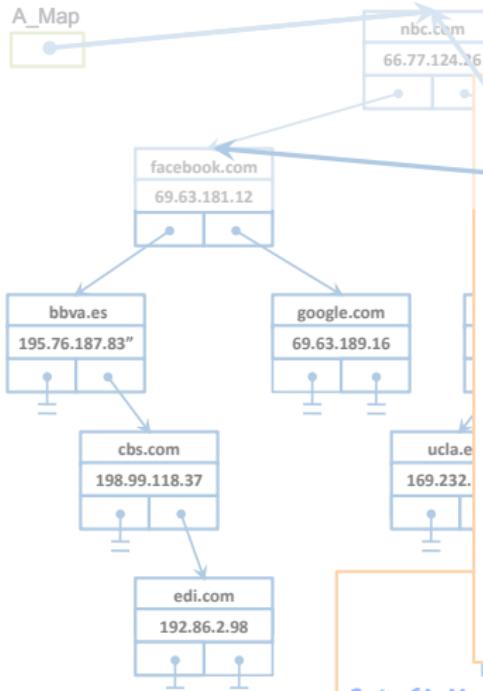
Get (M.Left, Key, Value, Success);
end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
<code>ASU.To_Unbounded_String ("google.com")</code>		<code>Null_.String</code>	?

```

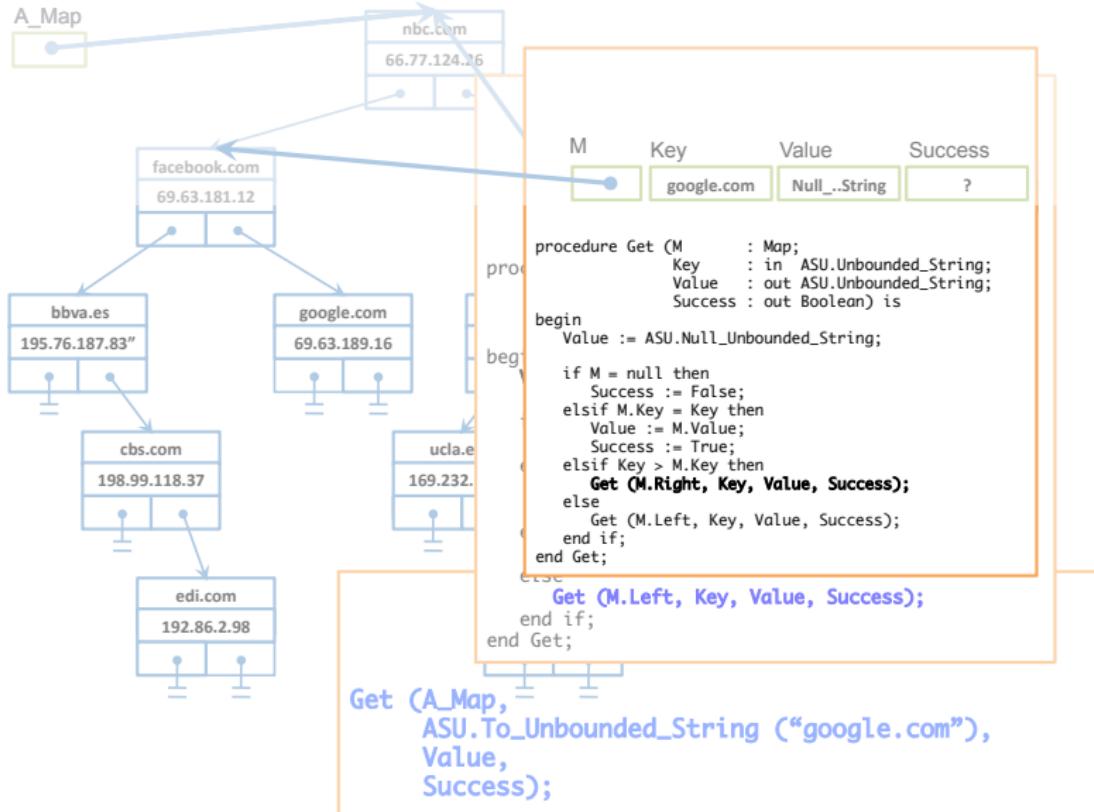
procedure Get (M      : Map;
              Key   : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

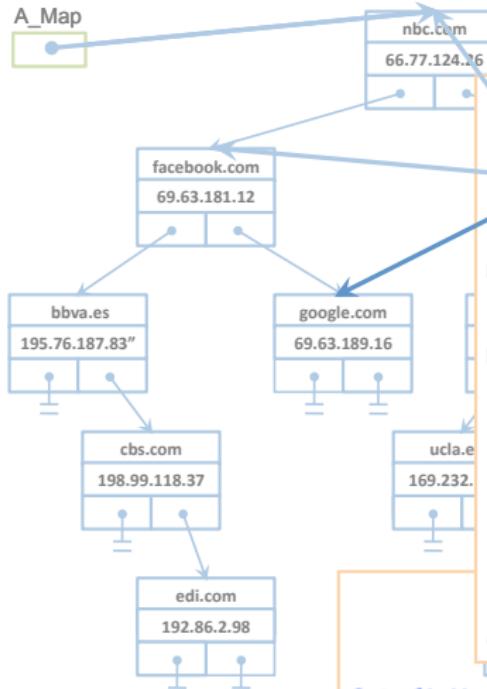
```

```

Get (M,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```





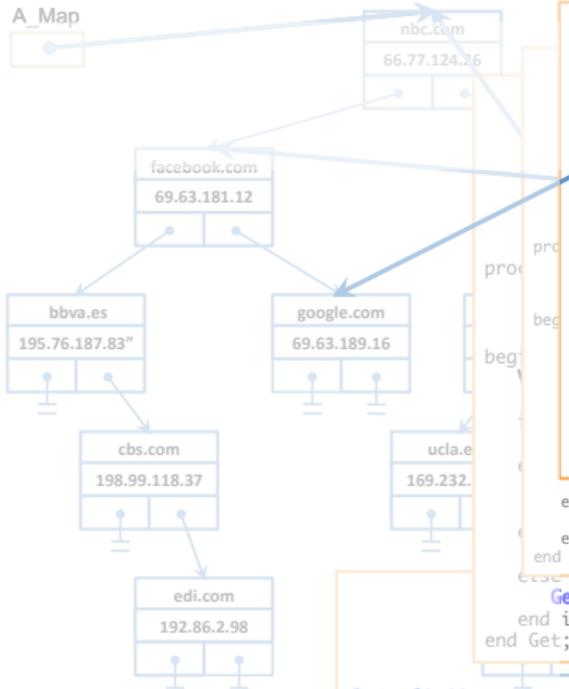
```

procedure Get (M : Map;
              Key : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

Get (M.Right, Key, Value, Success);
else
  Get (M.Left, Key, Value, Success);
end if;
end Get;
  
```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```



```

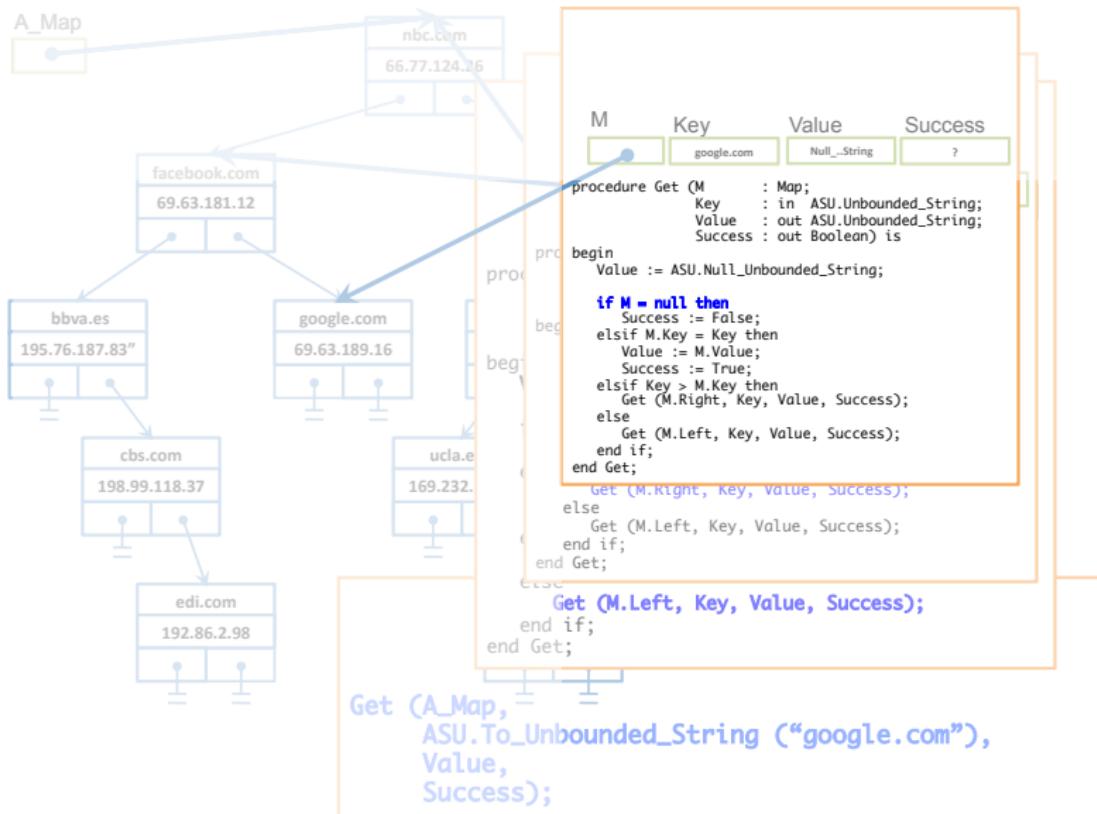
M   Key   Value   Success
---+---+---+---
procedure Get (M      : Map;
               Key    : in ASU.Unbounded_String;
               Value  : out ASU.Unbounded_String;
               Success: out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;
  
```

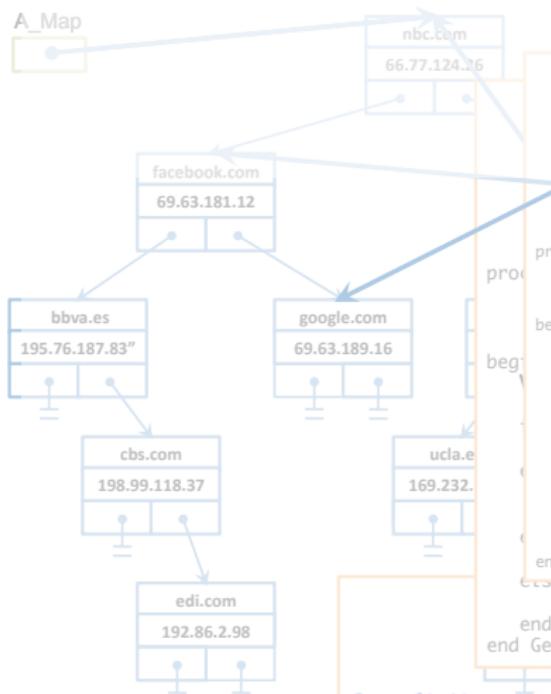
```

Get (M.Right, Key, Value, Success);
end if;
end Get;
  
```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```





```

M   Key   Value   Success
-----|-----|-----|-----
procedure Get (M      : Map;
               Key    : in ASU.Unbounded_String;
               Value  : out ASU.Unbounded_String;
               Success: out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

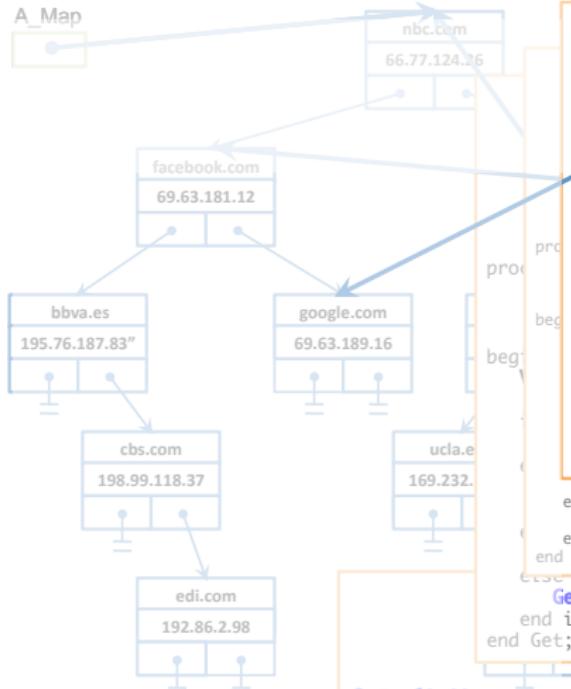
Get (M.Right, Key, Value, Success);
else
  Get (M.Left, Key, Value, Success);
end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



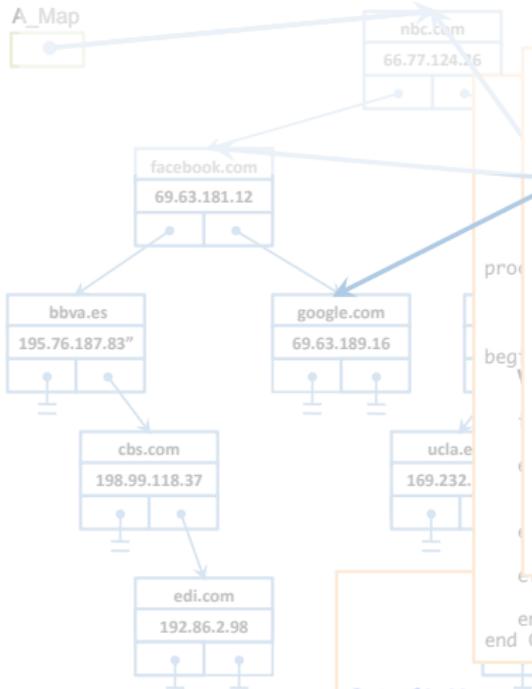
M	Key	Value	Success
ASU.Map	ASU.Unbounded_String ("google.com")	ASU.Null_Unbounded_String	?

```

procedure Get (M      : Map;
              Key   : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;
Get (M.Right, Key, Value, Success);
else
  Get (M.Left, Key, Value, Success);
end if;
end Get;
  
```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```



```

procedure Get (M : Map;
              Key : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

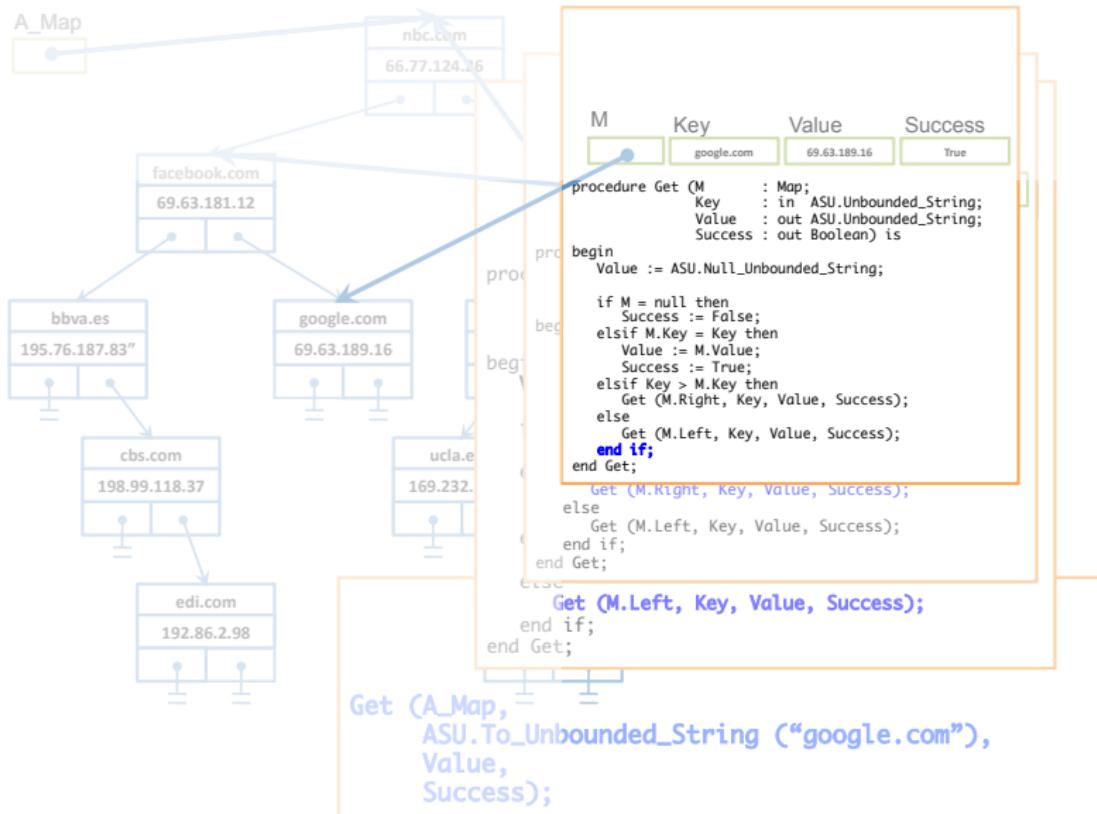
Get (M.Right, Key, Value, Success);
end if;
end Get;

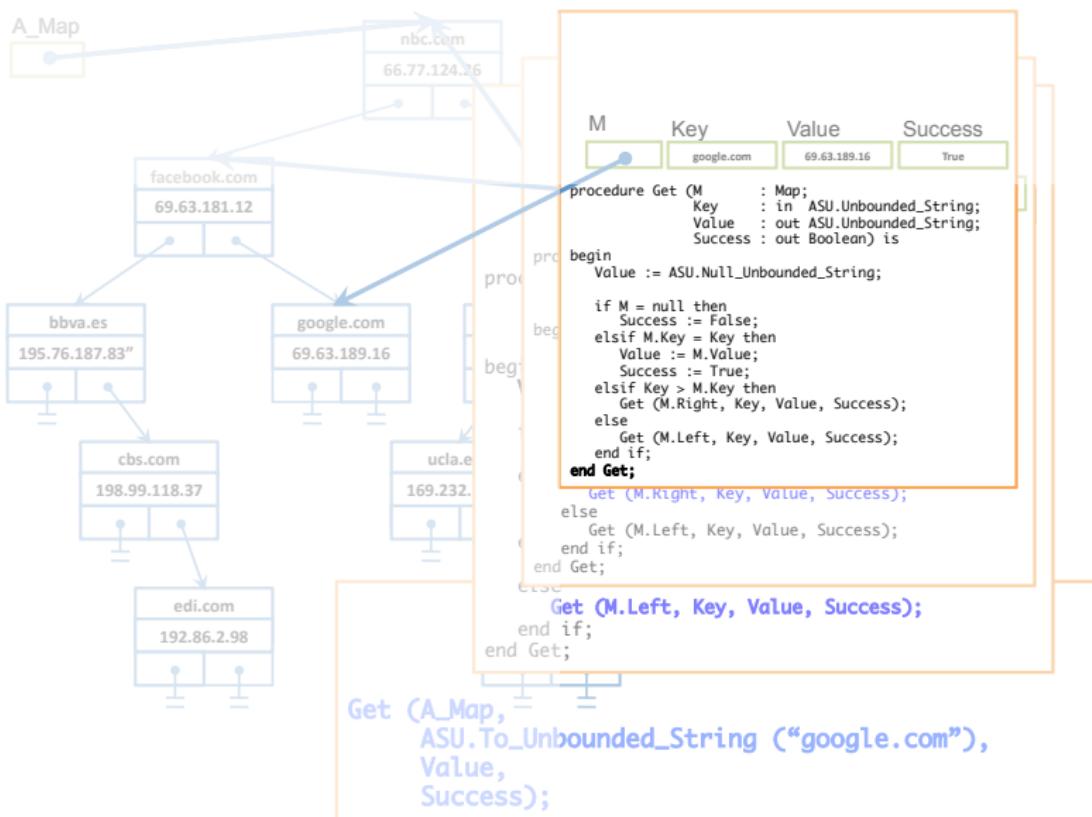
```

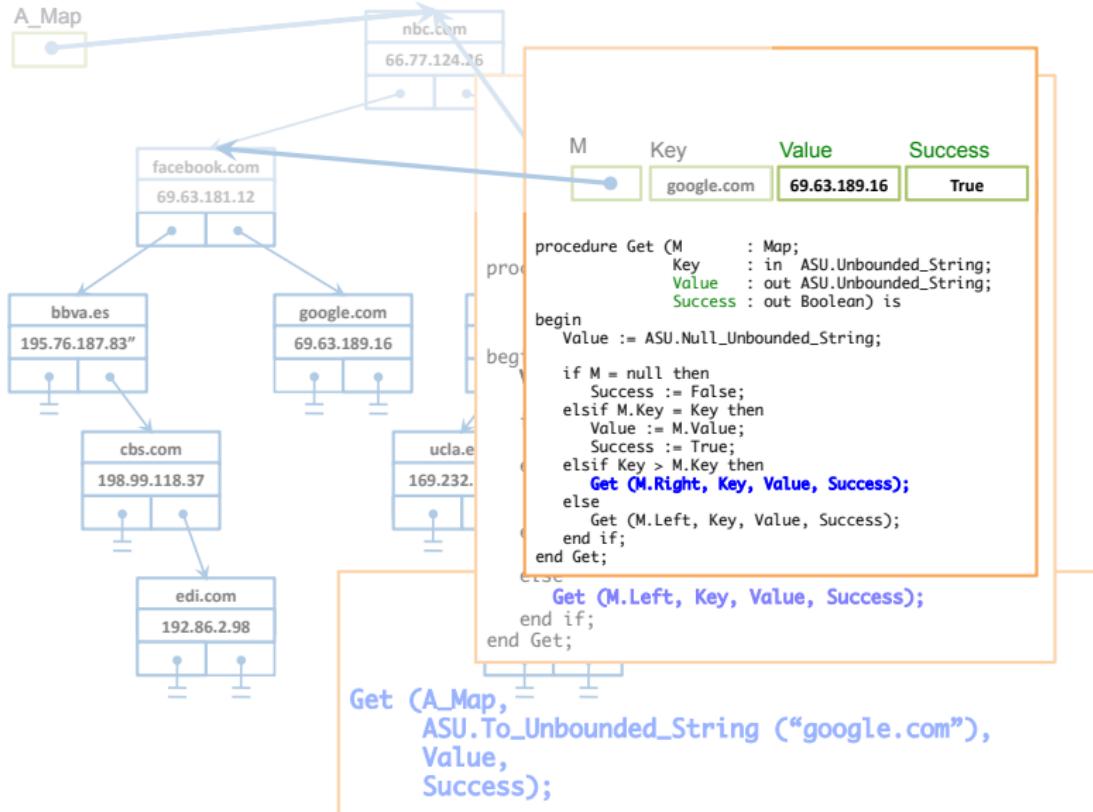
```

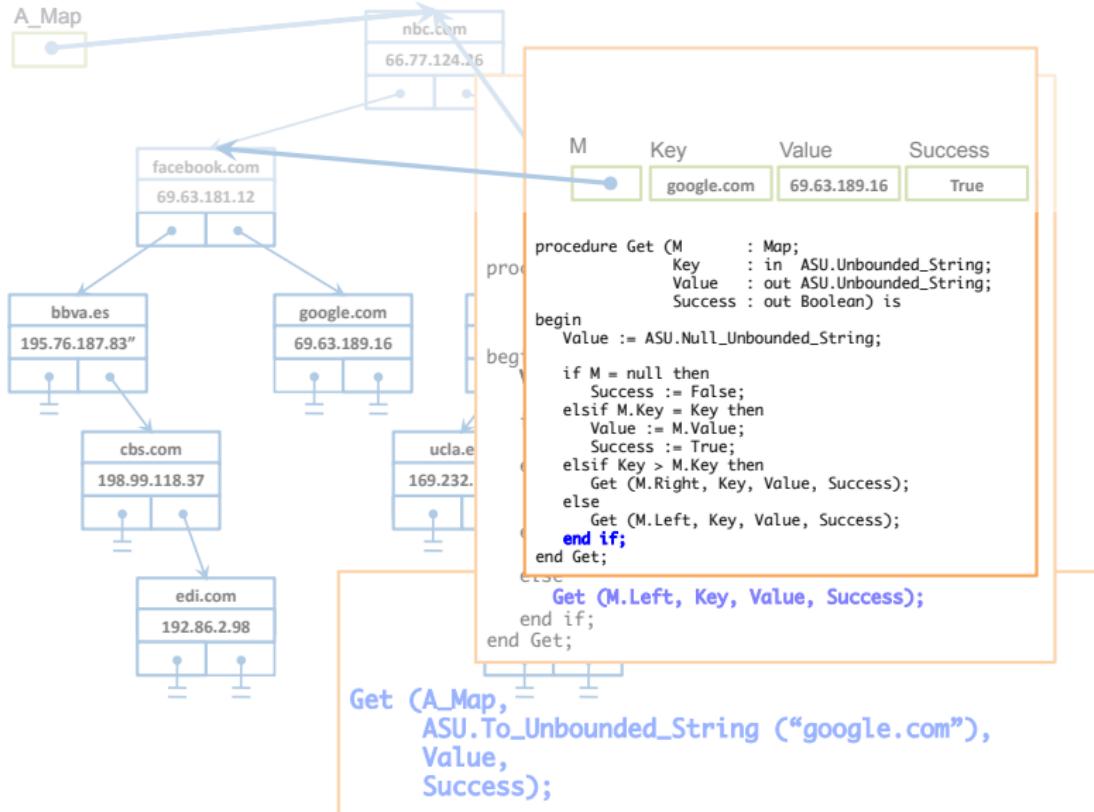
Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

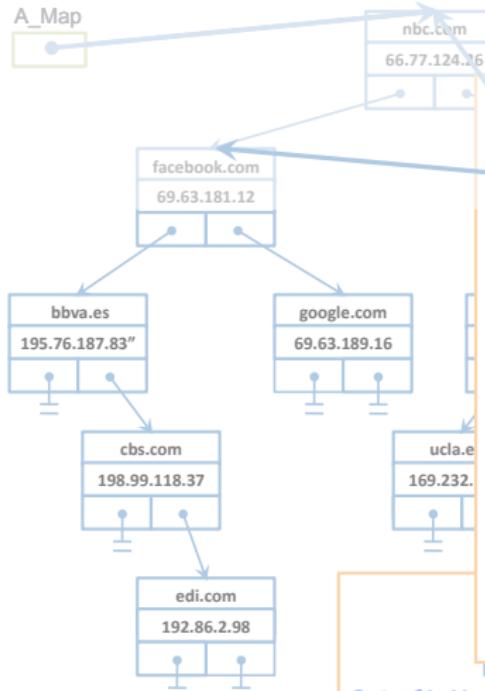
```







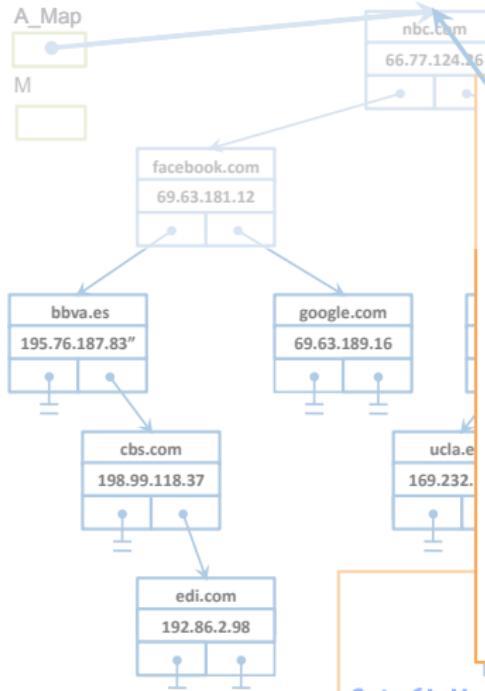




```

procedure Get (M      : Map;
              Key   : in ASU.Unbounded_String;
              Value : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);
  
```



M	Key	Value	Success
	<code>google.com</code>	<code>69.63.189.16</code>	<code>True</code>

```

procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

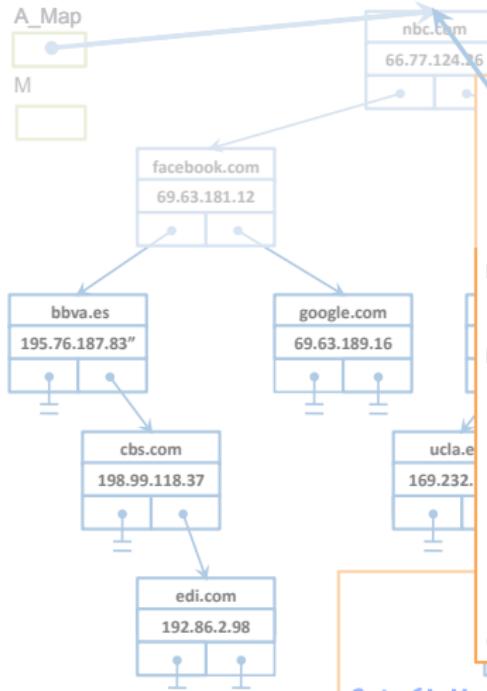
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
	google.com	69.63.189.16	True

```

procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

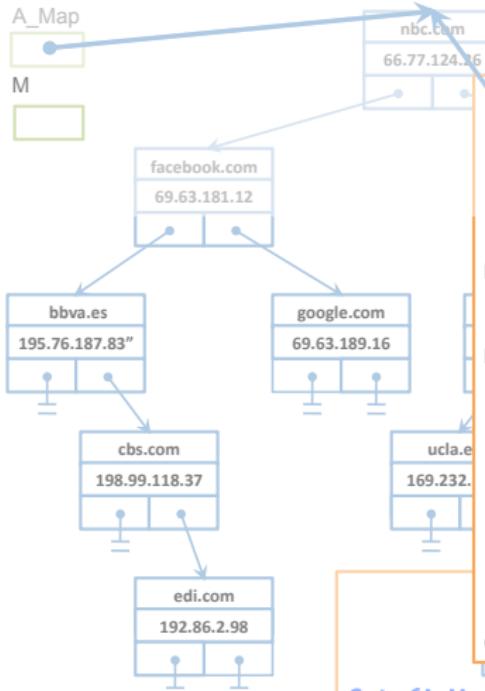
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



M	Key	Value	Success
	<code>google.com</code>	<code>69.63.189.16</code>	<code>True</code>

```

procedure Get (M      : Map;
              Key    : in ASU.Unbounded_String;
              Value  : out ASU.Unbounded_String;
              Success : out Boolean) is
begin
  Value := ASU.Null_Unbounded_String;

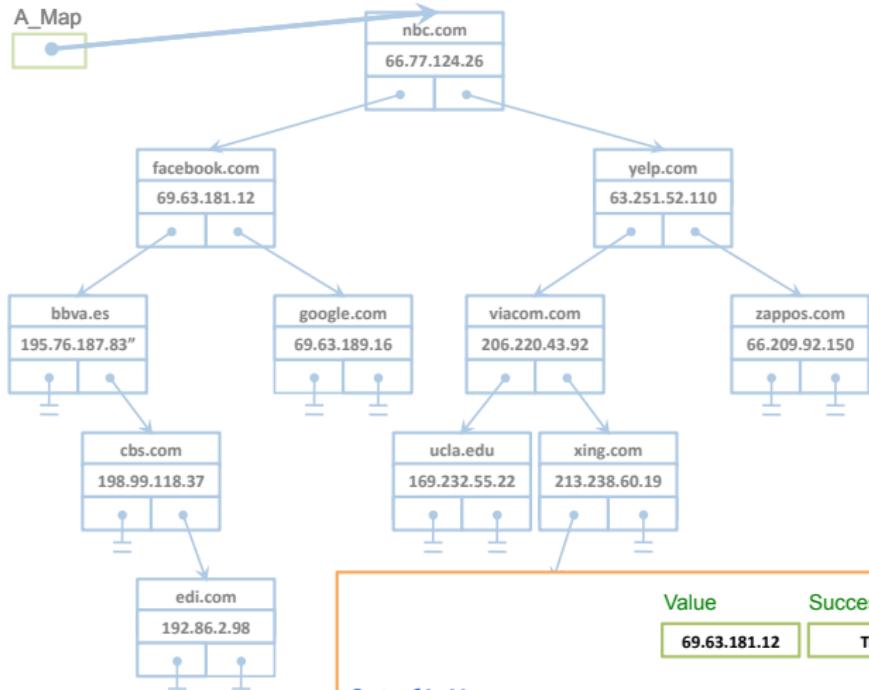
  if M = null then
    Success := False;
  elsif M.Key = Key then
    Value := M.Value;
    Success := True;
  elsif Key > M.Key then
    Get (M.Right, Key, Value, Success);
  else
    Get (M.Left, Key, Value, Success);
  end if;
end Get;

```

```

Get (A_Map,
     ASU.To_Unbounded_String ("google.com"),
     Value,
     Success);

```



```
Get (A_Map,
ASU.To_Unbounded_String ("google.com"),
Value,
Success);
```

Value	Success
69.63.181.12	True

Inserción de un nodo en un árbol

Se recorre el árbol desde la raíz hasta encontrar la posición que corresponda al nodo que se inserta en función de su clave

Para insertar un nuevo nodo con (*Key*, *Value*) en un árbol del que conocemos su nodo raíz (*Root*):

- Si el árbol está vacío \Rightarrow se crea un nuevo nodo con (*Key*, *Value*), que se convierte en la nueva raíz
- Si el árbol no está vacío \Rightarrow
 - Si $Key = Root.Key \Rightarrow Root.Value := Value$
 - Si $Key < Root.Key \Rightarrow$ se asigna al subárbol izquierdo el resultado de insertar el nodo en el subárbol izquierdo
 - Si $Key > Root.Key \Rightarrow$ se asigna al subárbol derecho el resultado de insertar el nodo en el subárbol derecho

Inserción de un nodo en un árbol: código

```
procedure Put (M      : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String) is
begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;
```

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ **Ejemplo de ejecución: Put en un ABB vacío**
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

A_Map



```
Put (A_Map,  
      ASU.To_Unbounded_String ("nbc.com"),  
      ASU.To_Unbounded_String ("66.77.124.26"));
```

A_Map



M	Key	Value
	nbc.com	66.77.124.26

```
procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;
end Put;
```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

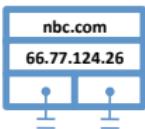
begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

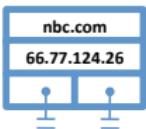
begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

Put



M	Key	Value
	nbc.com	66.77.124.26

```

procedure Put (M : in out Map;
              Key    : ASU.Unbounded_String;
              Value  : ASU.Unbounded_String)

begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

end Put;

```

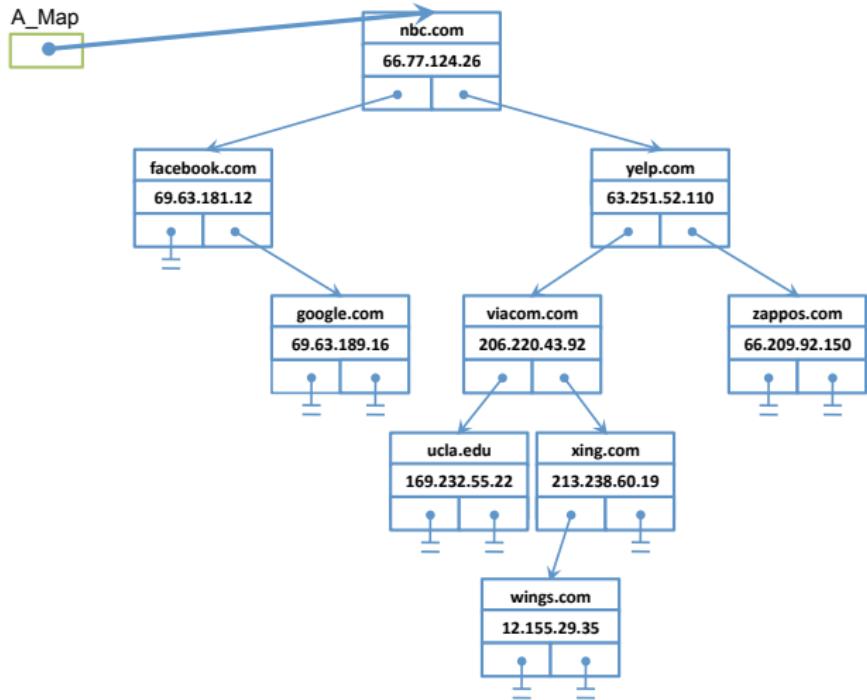
Put

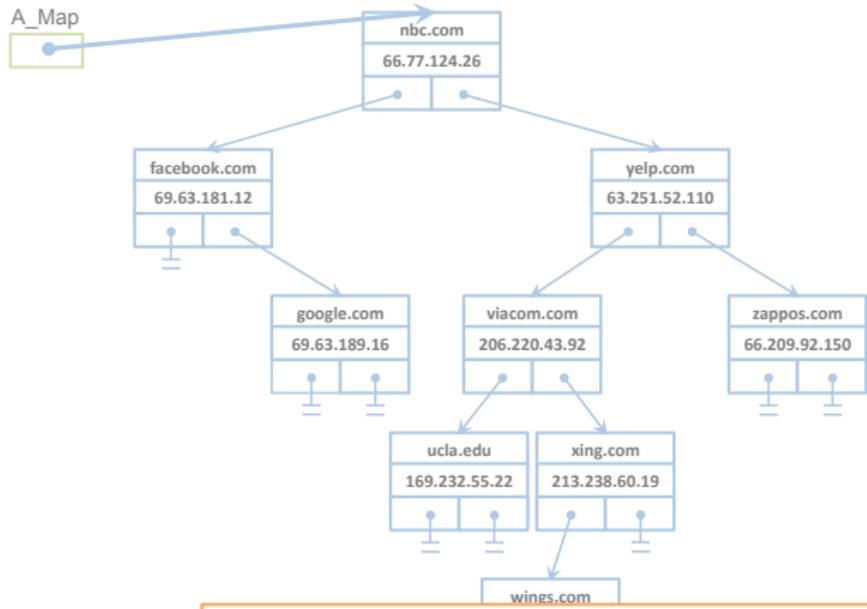


```
Put (A_Map,  
      ASU.To_Unbounded_String ("nbc.com"),  
      ASU.To_Unbounded_String ("66.77.124.26"));
```

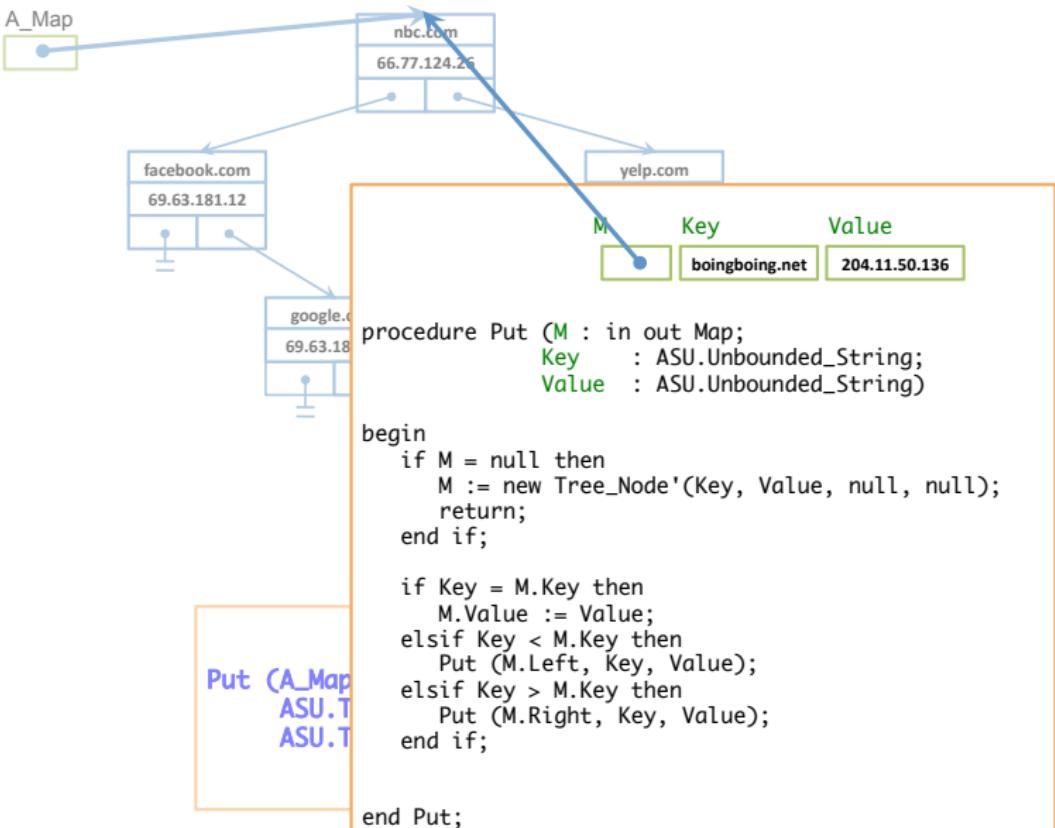
Contenidos

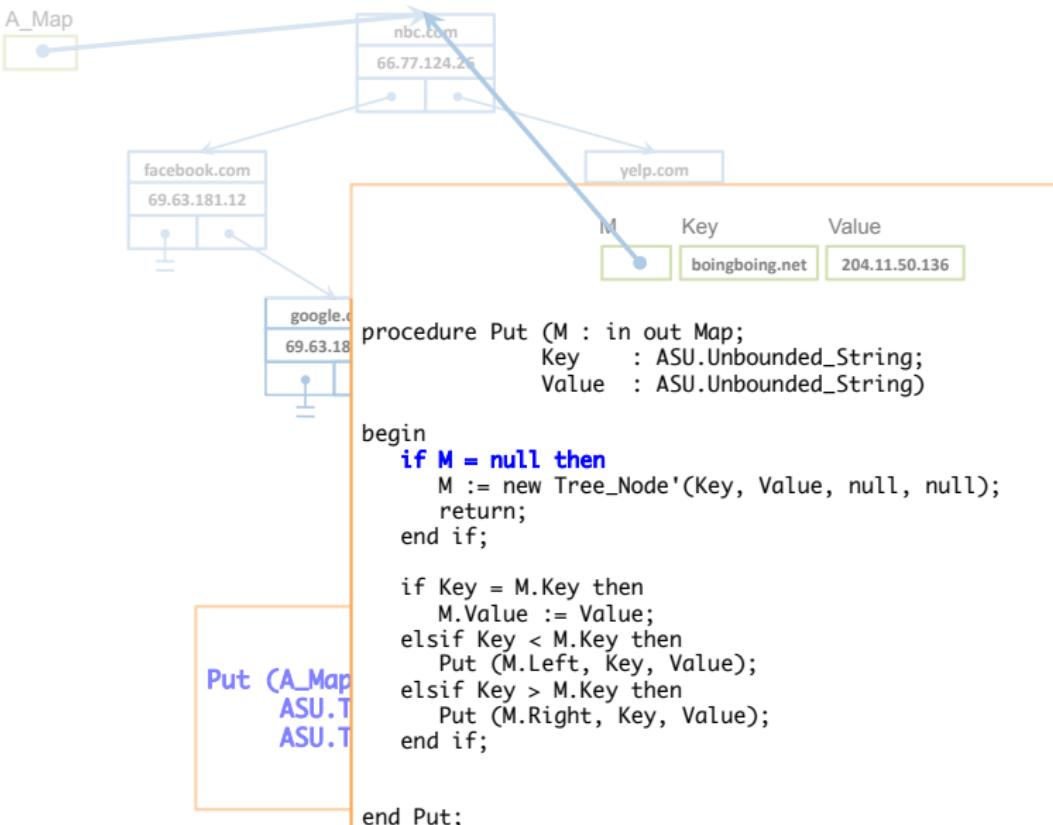
- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ **Ejemplo de ejecución: Put en un ABB**
- ⑫ Borrado de un nodo en un ABB



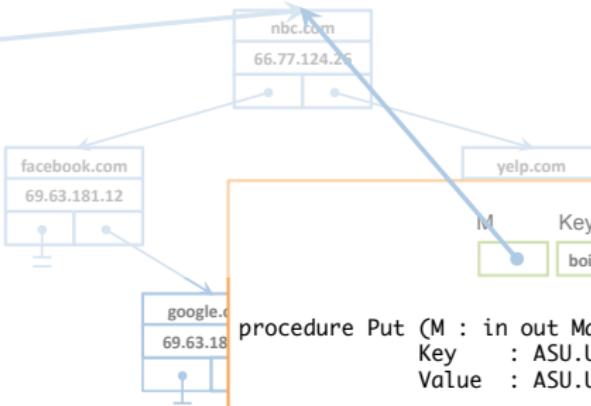


```
Put (A_Map,  
    ASU.To_Unbounded_String ("boingboing.net"),  
    ASU.To_Unbounded_String ("204.11.50.136"));
```





A_Map



M	Key	Value

```

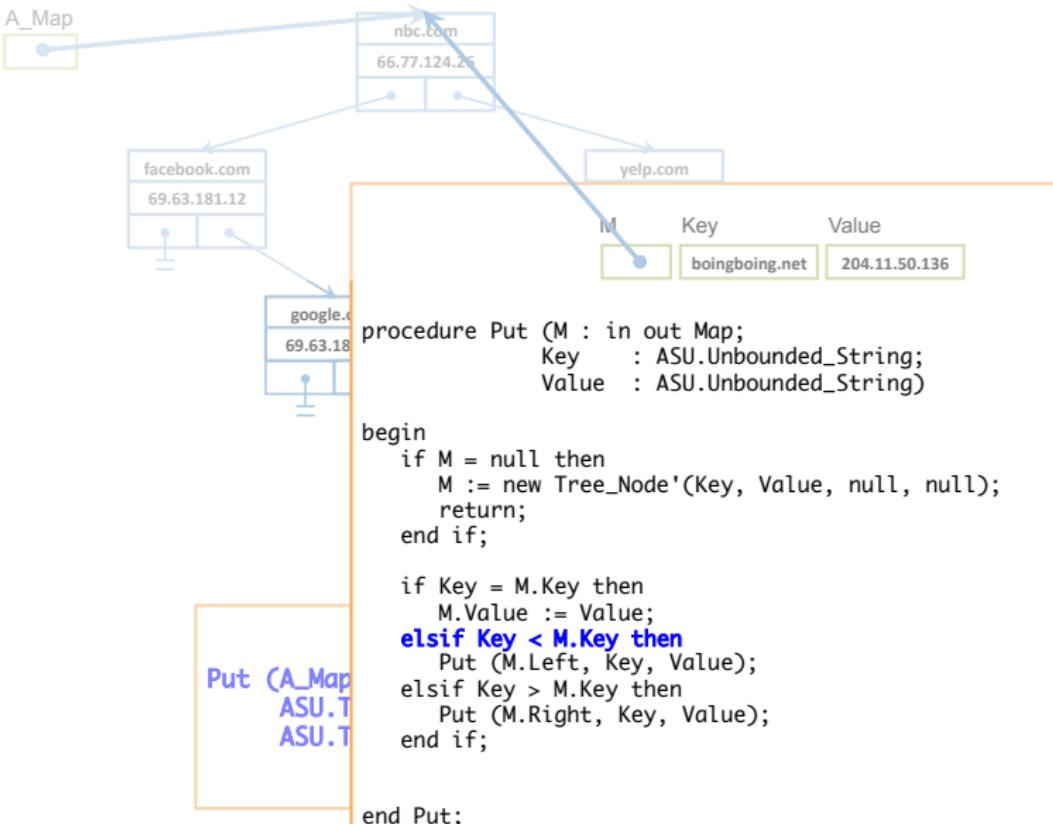
procedure Put (M : in out Map;
               Key      : ASU.Unbounded_String;
               Value    : ASU.Unbounded_String)

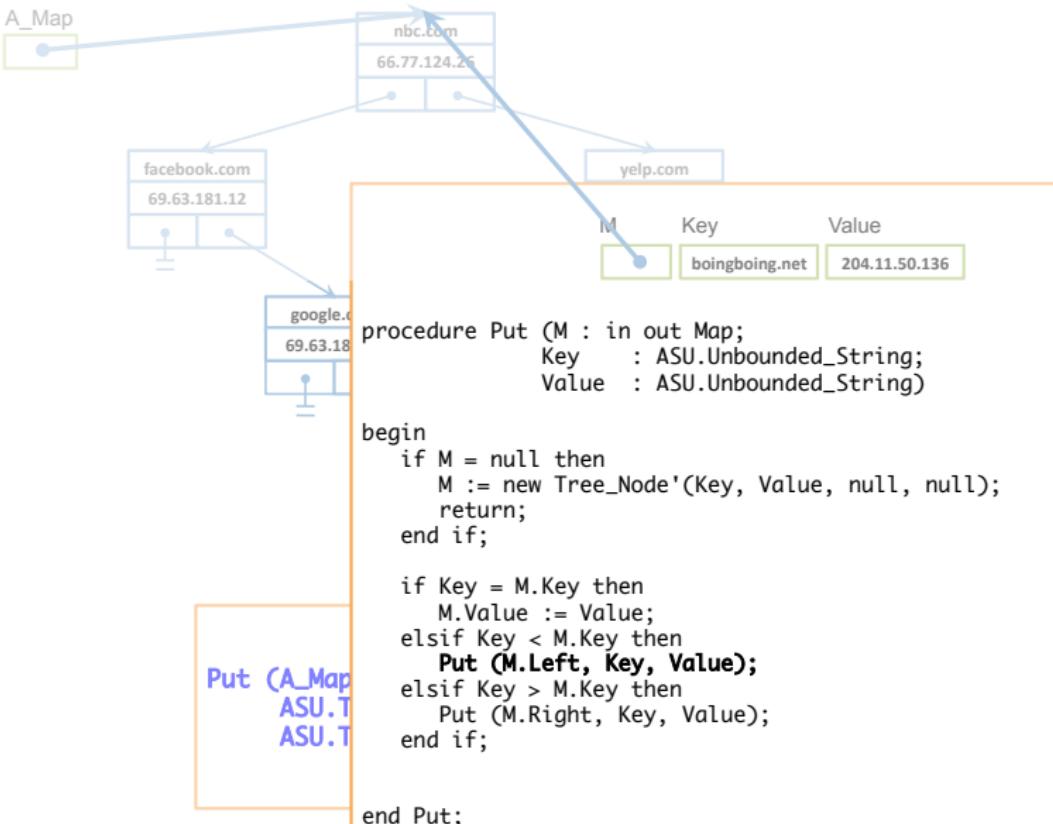
begin
  if M = null then
    M := new Tree_Node'(Key, Value, null, null);
    return;
  end if;

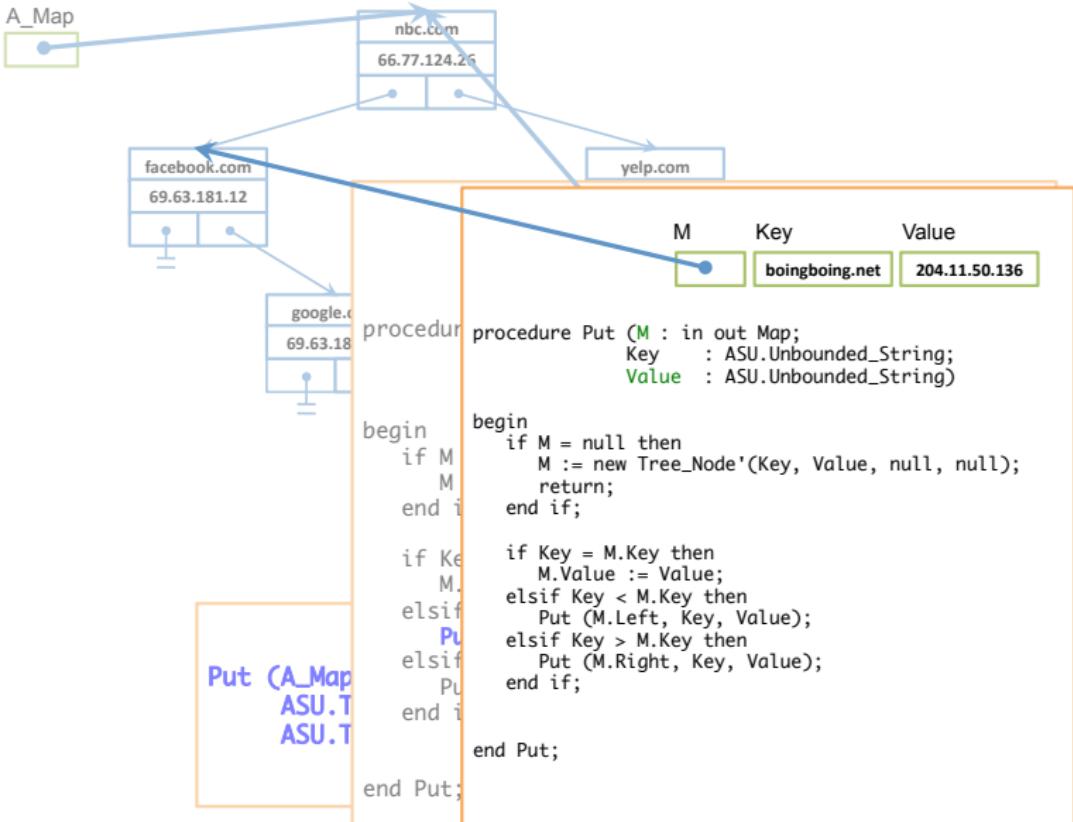
  if Key = M.Key then
    M.Value := Value;
  elsif Key < M.Key then
    Put (M.Left, Key, Value);
  elsif Key > M.Key then
    Put (M.Right, Key, Value);
  end if;

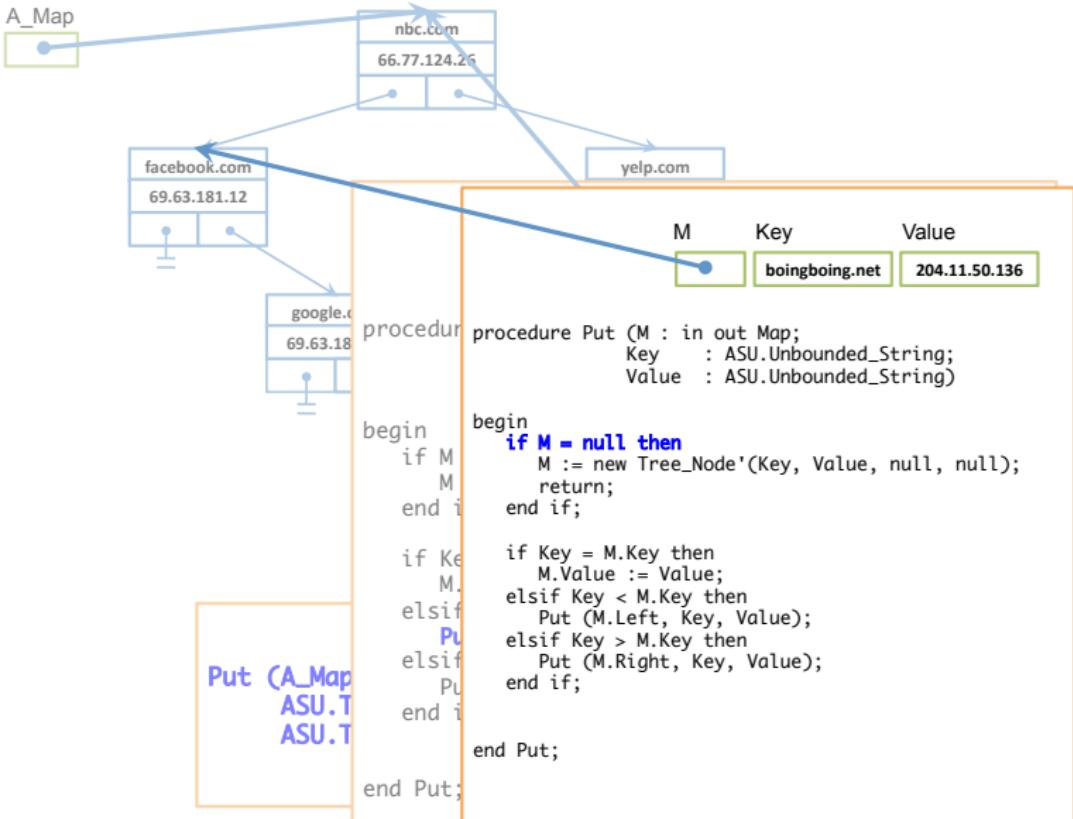
end Put;
  
```

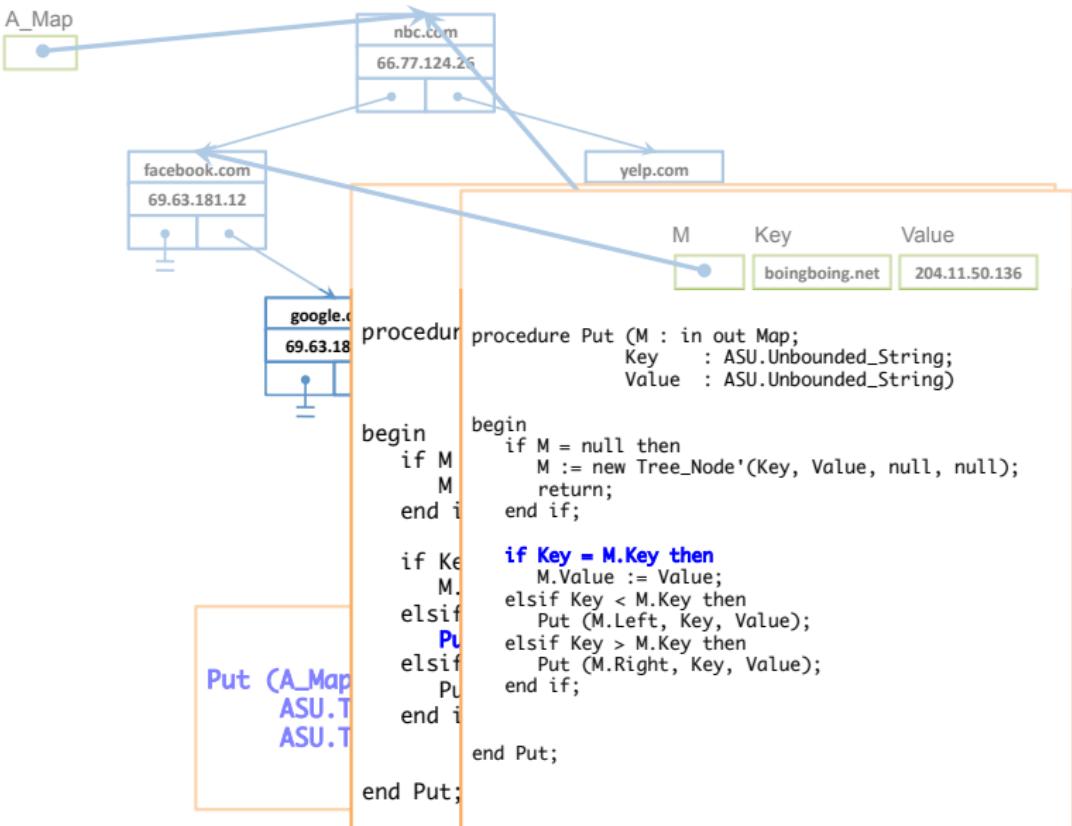
Put (A_Map
ASU.T
ASU.T)

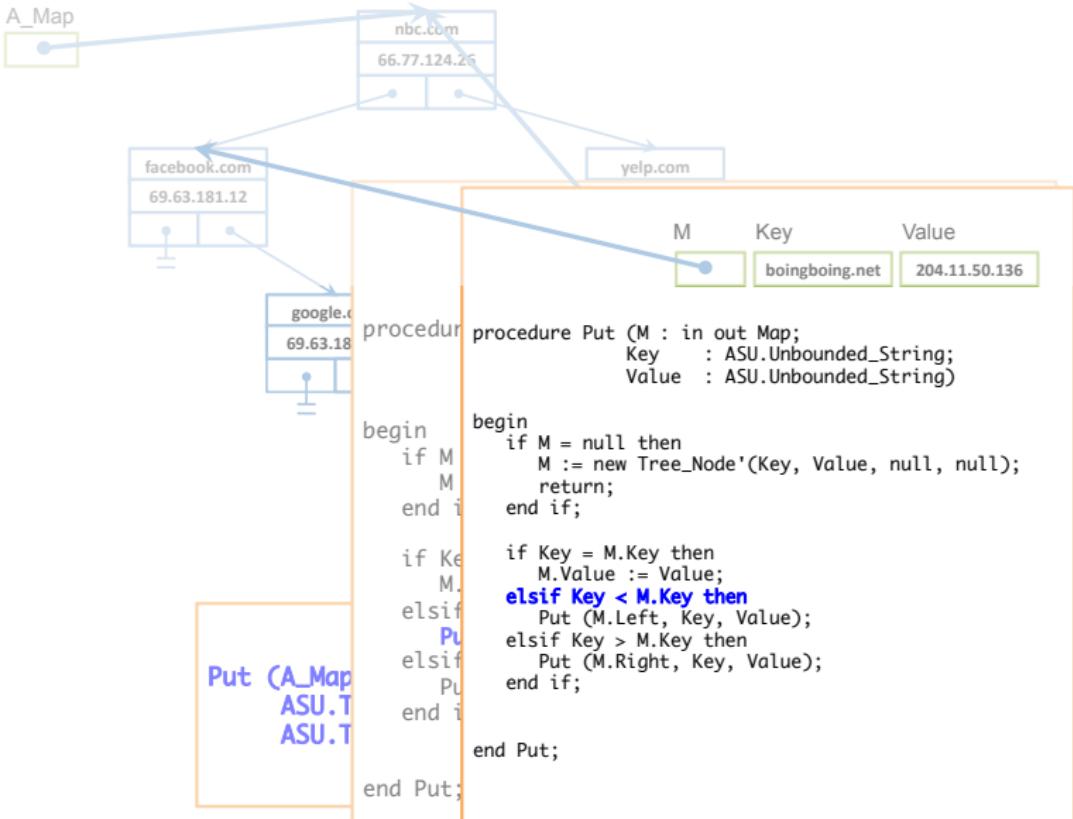


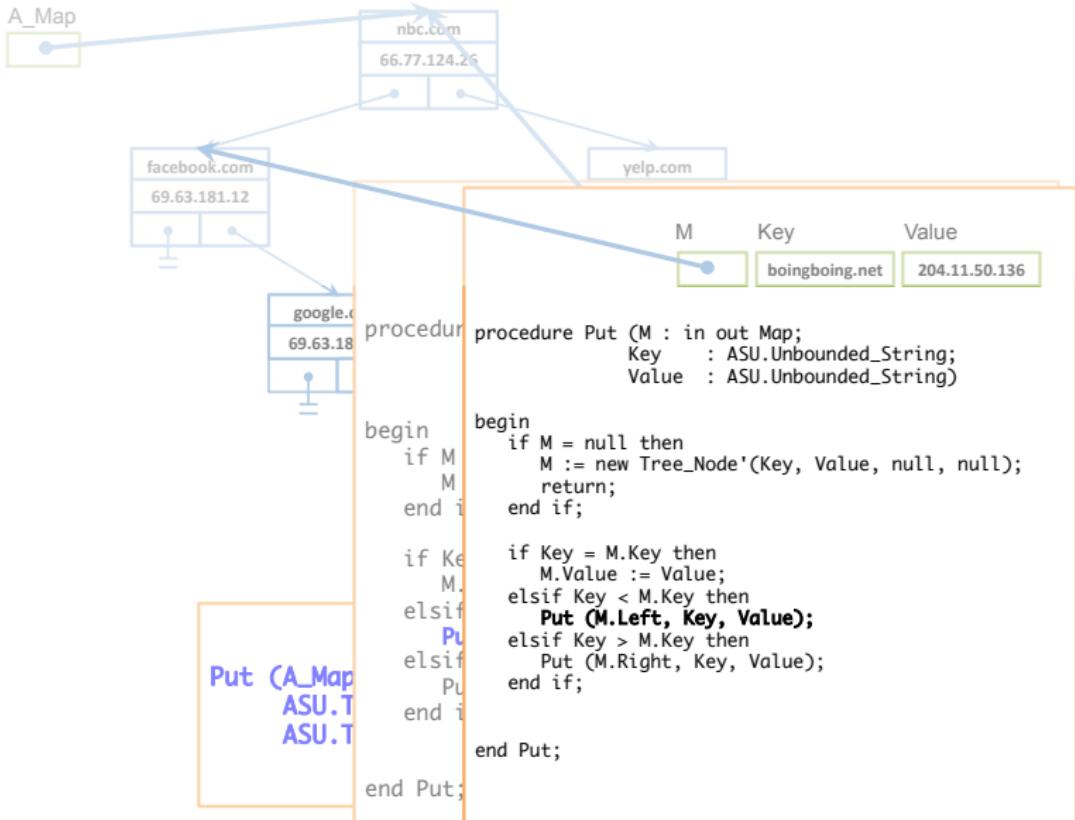


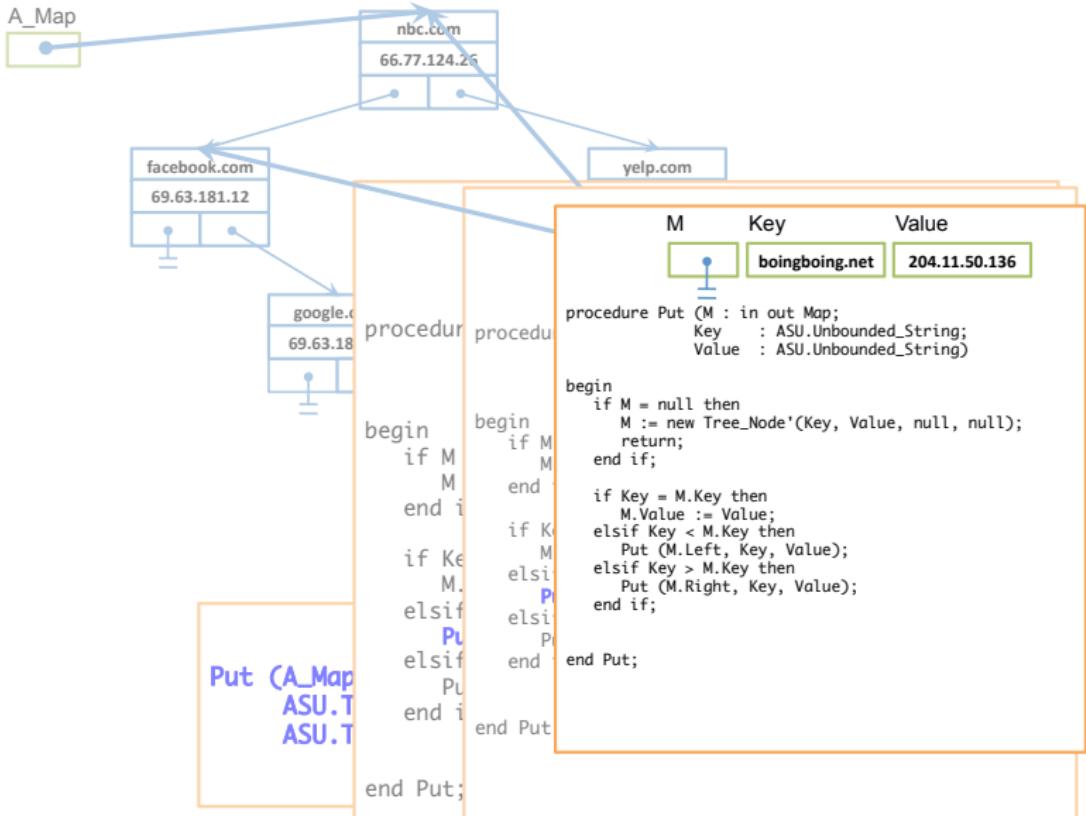


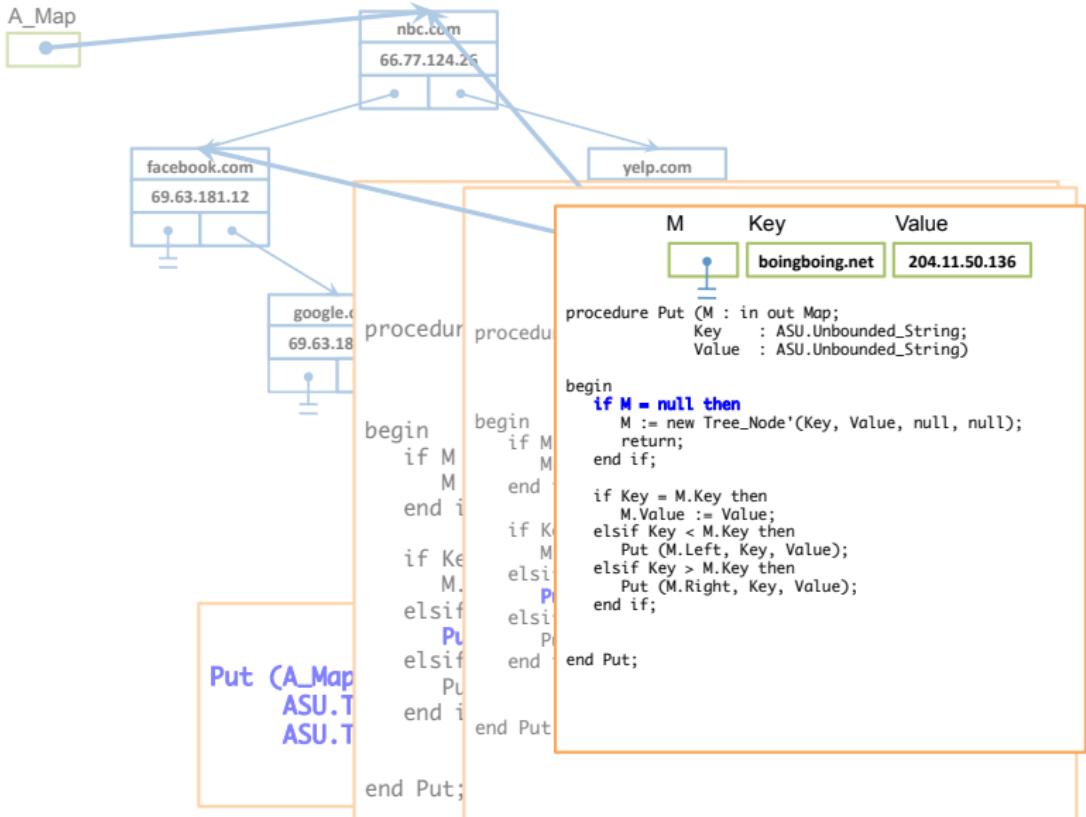


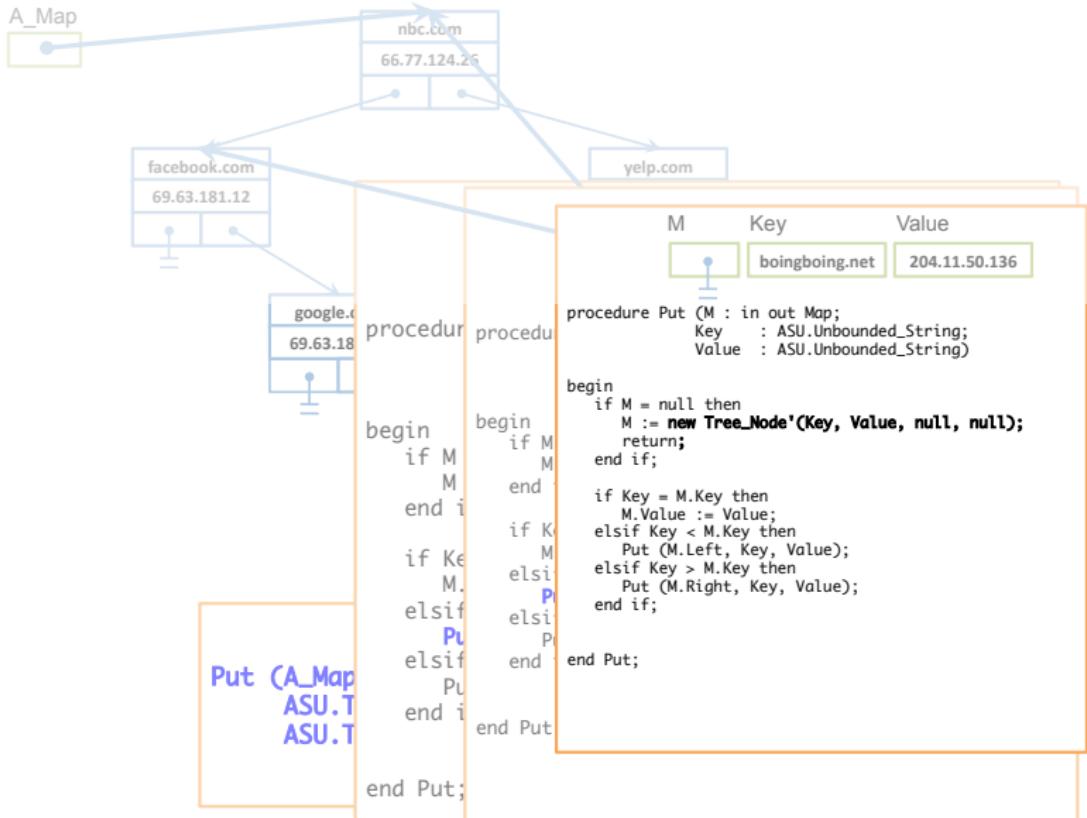


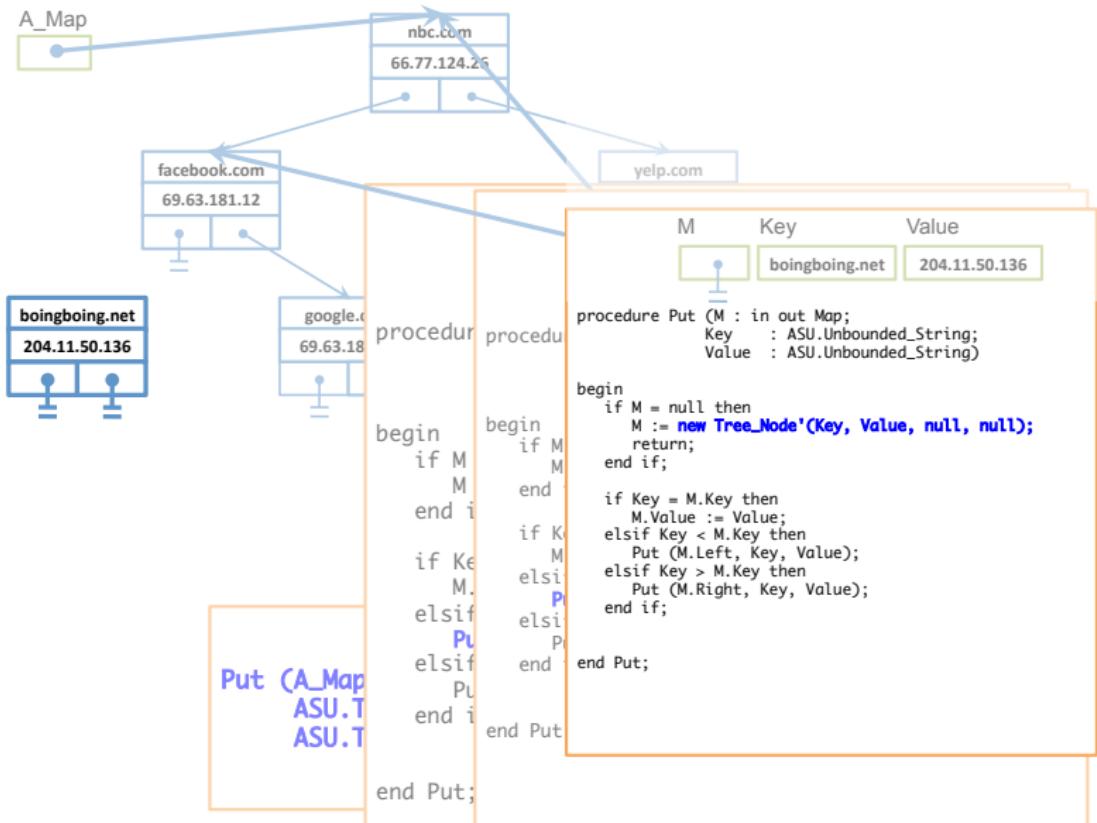


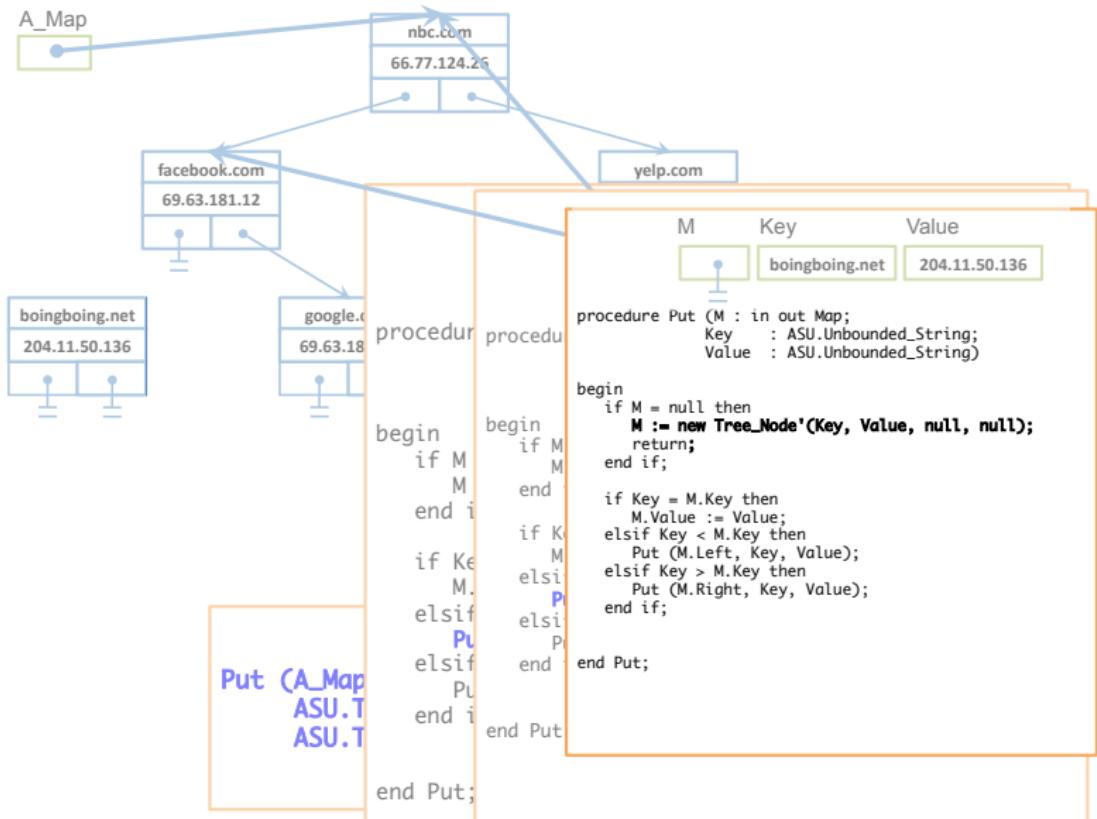


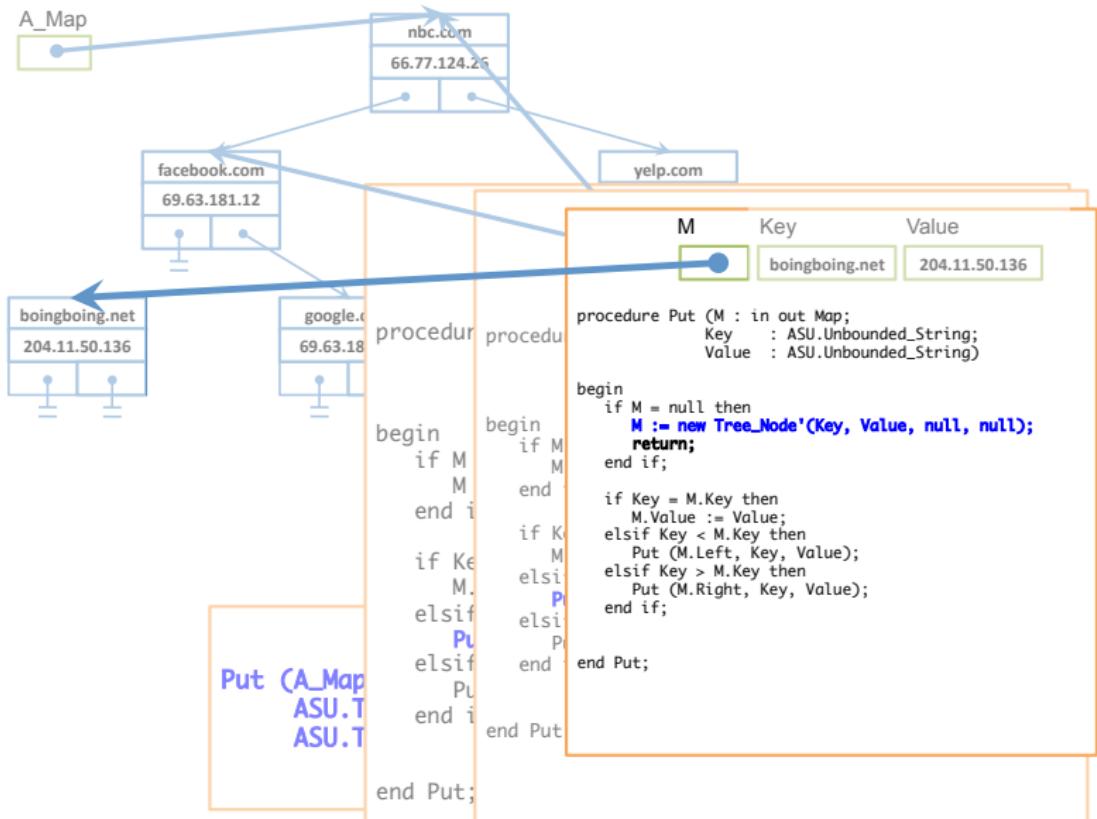


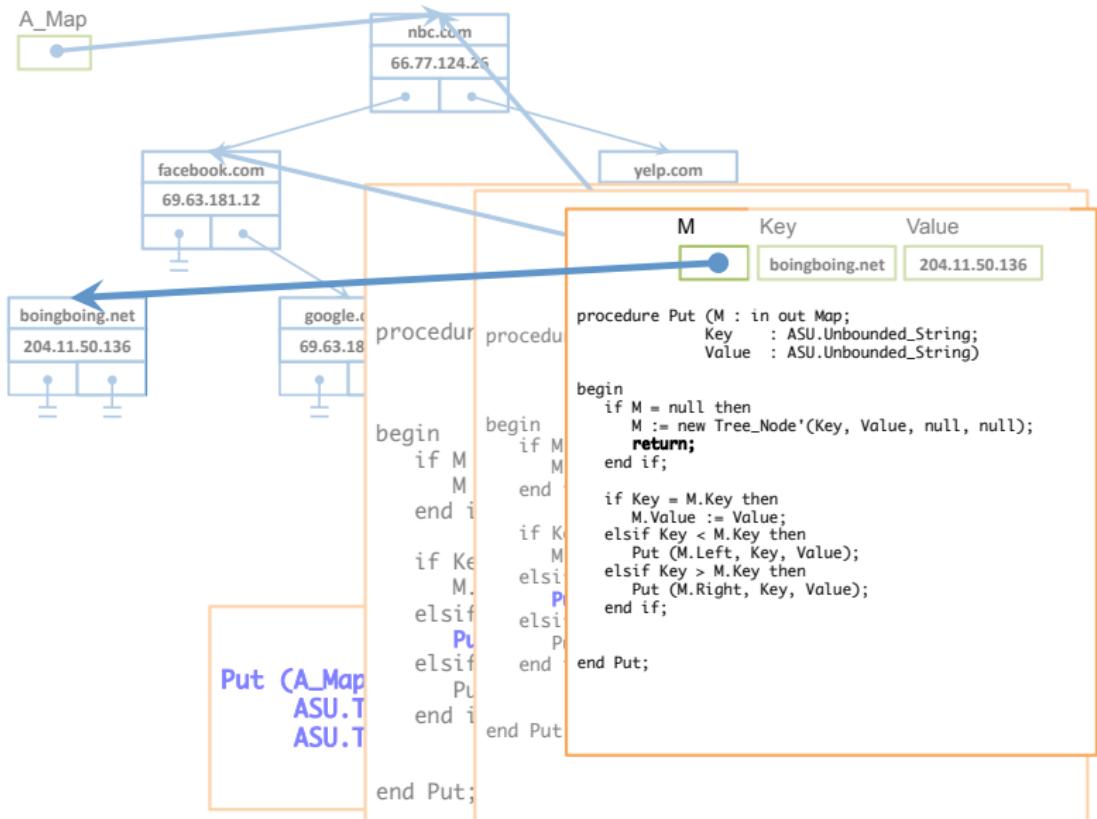


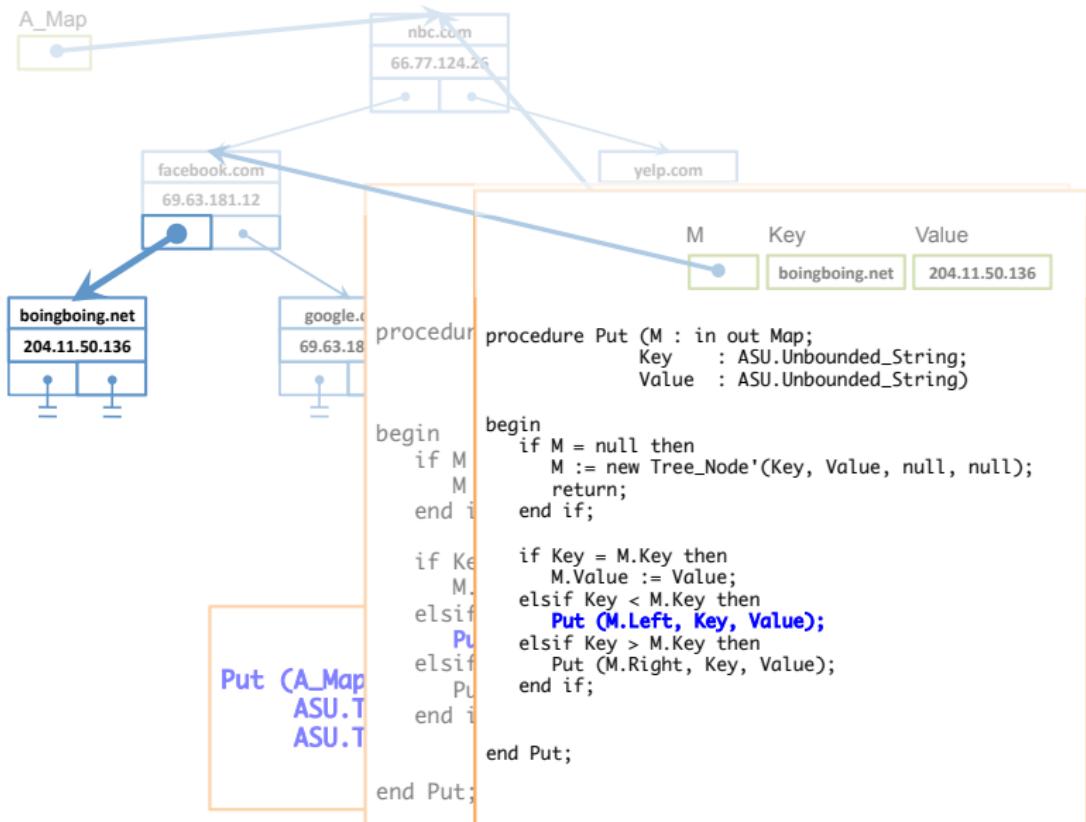


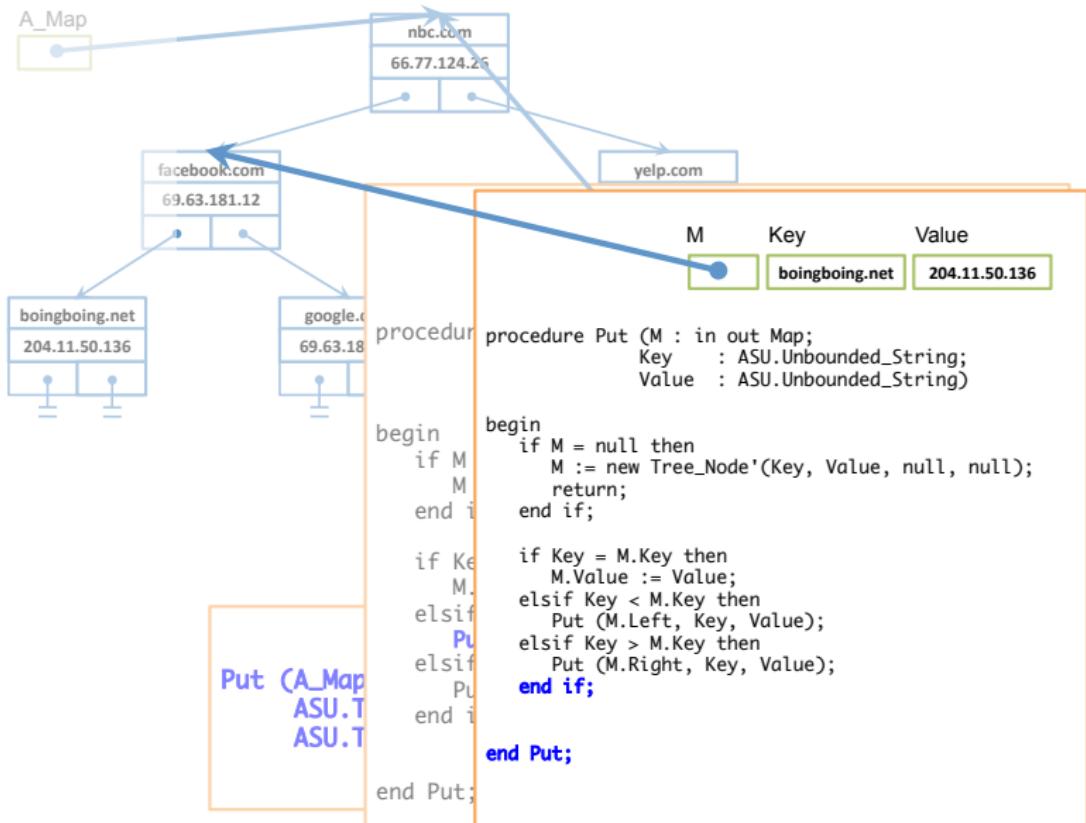


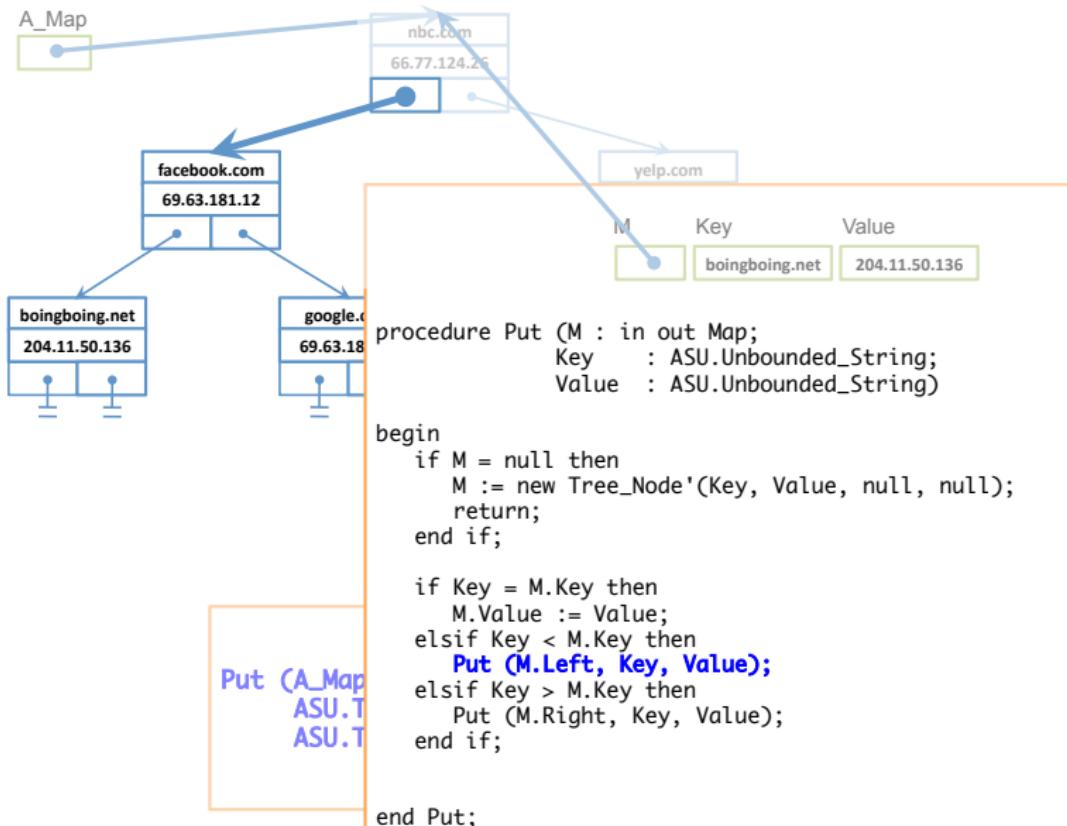


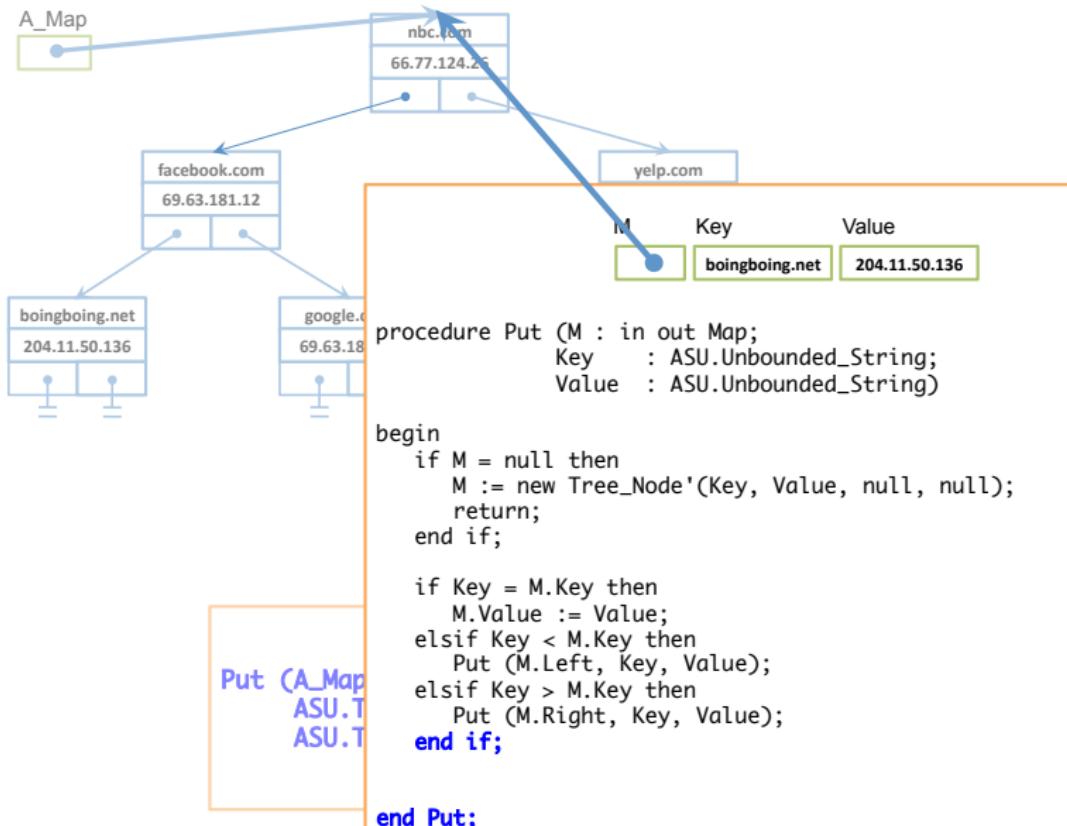


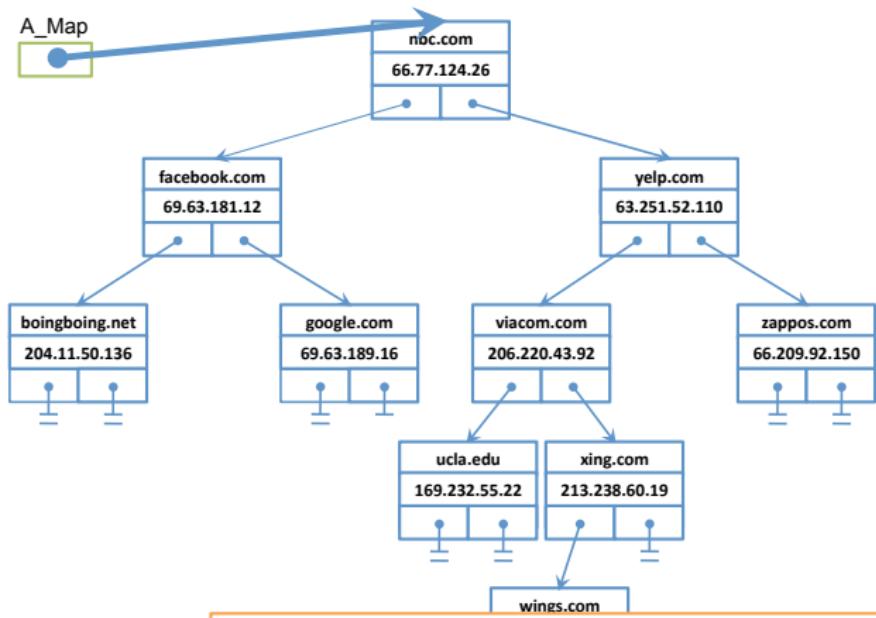












```
Put (A_Map,  
    ASU.To_Unbounded_String ("boingboing.net"),  
    ASU.To_Unbounded_String ("204.11.50.136"));
```

Contenidos

- ① Tablas de Símbolos
- ② Implementación de TS mediante un array no ordenado
- ③ Implementación de TS mediante una lista enlazada no ordenada
- ④ Ejemplo de ejecución (TS mediante lista enlazada no ordenada)
- ⑤ Iteración sobre todos los elementos de una colección
- ⑥ Implementación de TS mediante un Array ordenado
- ⑦ Implementación de TS mediante una lista enlazada ordenada
- ⑧ Implementación de TS mediante un árbol de búsqueda binaria (ABB)
- ⑨ Ejemplo de ejecución: Get en un ABB
- ⑩ Ejemplo de ejecución: Put en un ABB vacío
- ⑪ Ejemplo de ejecución: Put en un ABB
- ⑫ Borrado de un nodo en un ABB

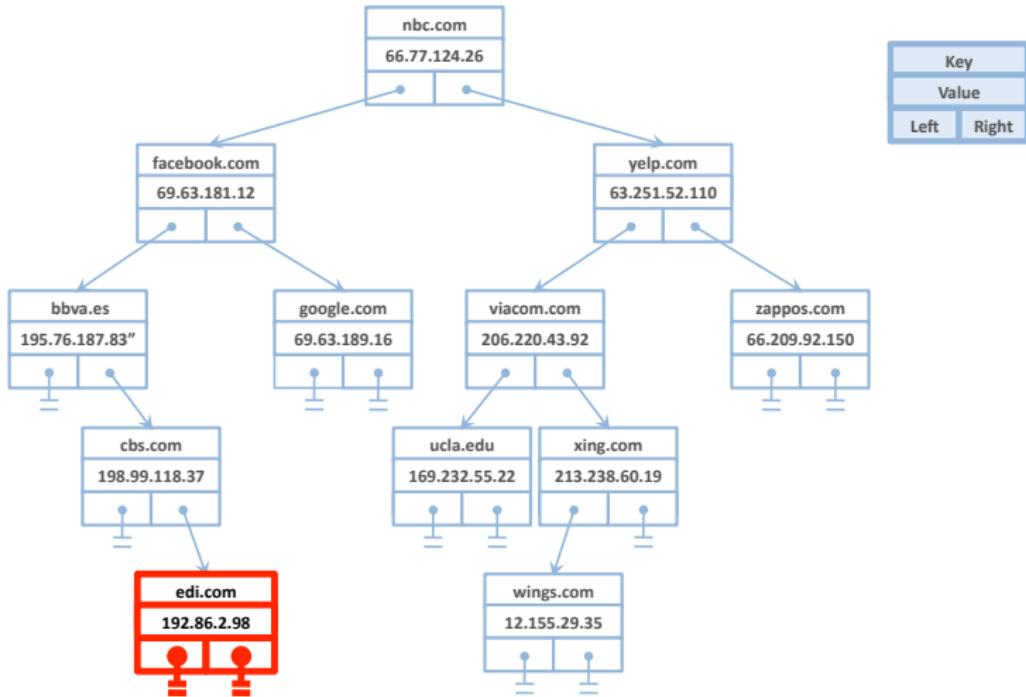
Borrado de un nodo

También requiere buscar el nodo a borrar, pero una vez localizado el nodo a borrar surgen varios casos:

- ① Borrado de un nodo que no tiene hijos
- ② Borrado de un nodo que sólo tiene un subárbol hijo
- ③ Borrado de un nodo con dos subárboles hijos

1. Borrado de un nodo j que no tiene hijos

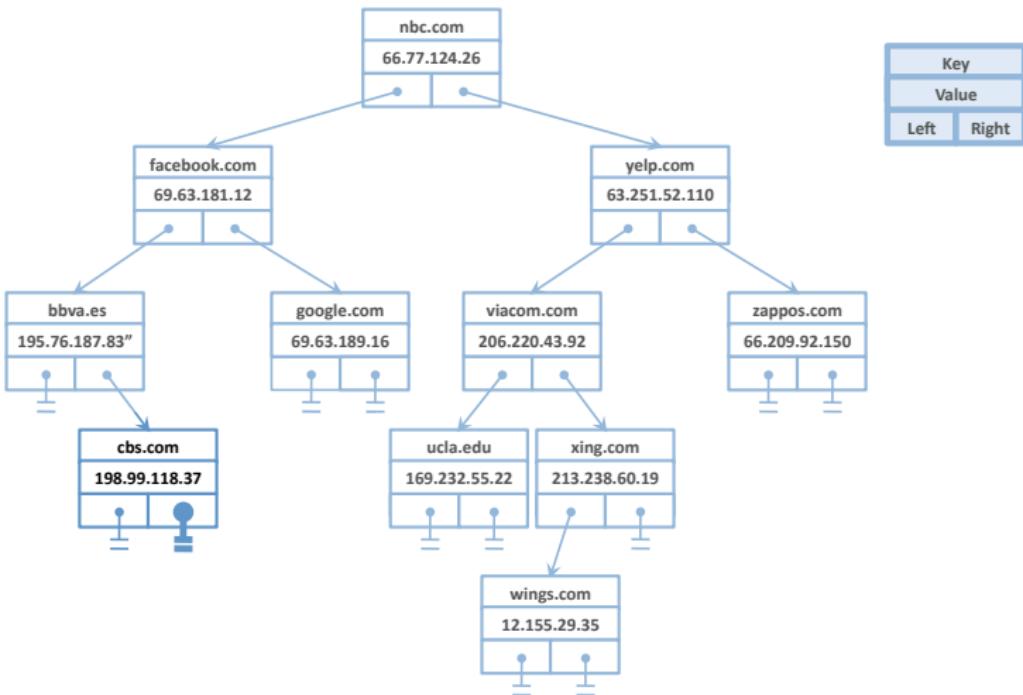
Se asigna null al campo (*Left* o *Right*) que apunta a j en el nodo padre de j



```
Delete (A_Map, ASU.To_Unbounded_String ("edi.com"), Success);
```



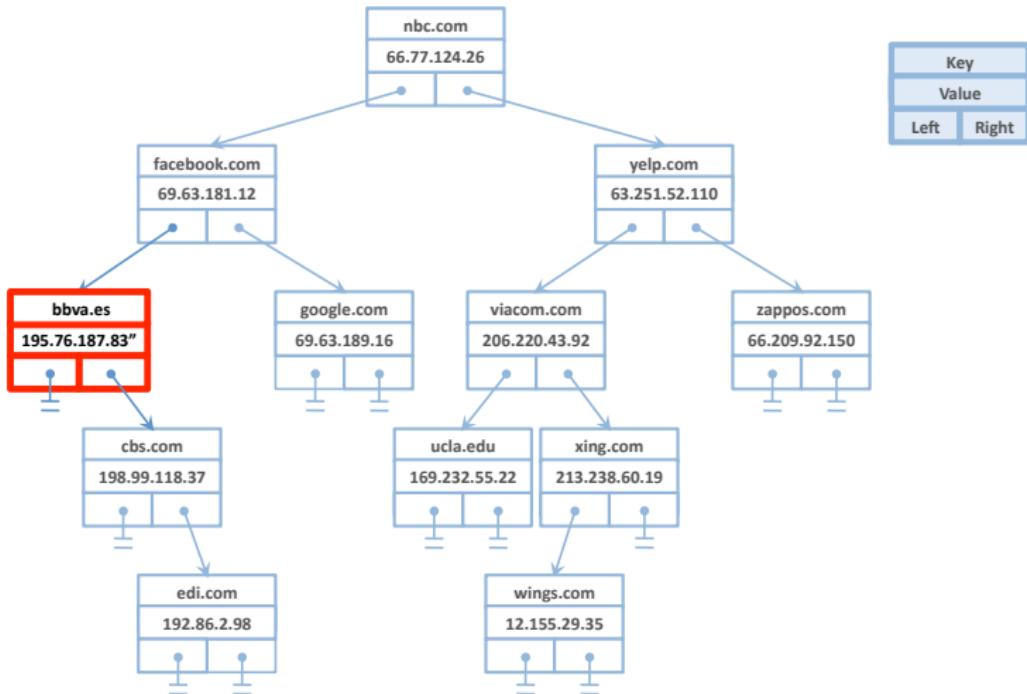
```
Delete (A_Map, ASU.To_Unbounded_String ("edi.com"), Success);
```



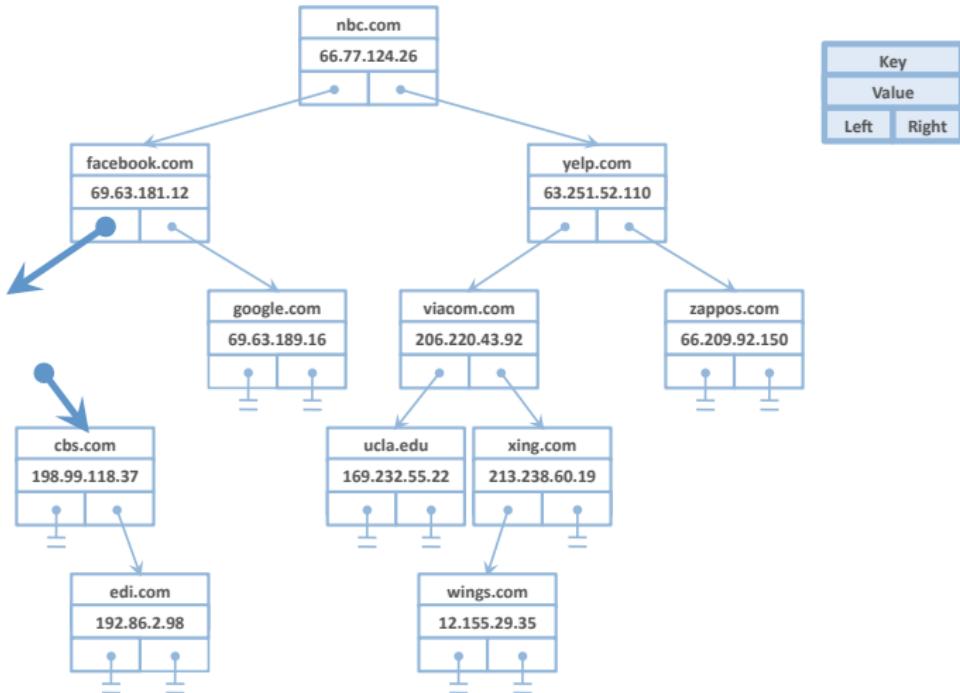
```
Delete (A_Map, ASU.To_Unbounded_String ("edi.com"), Success);
```

2. Borrado de un nodo j que sólo tiene un subárbol hijo

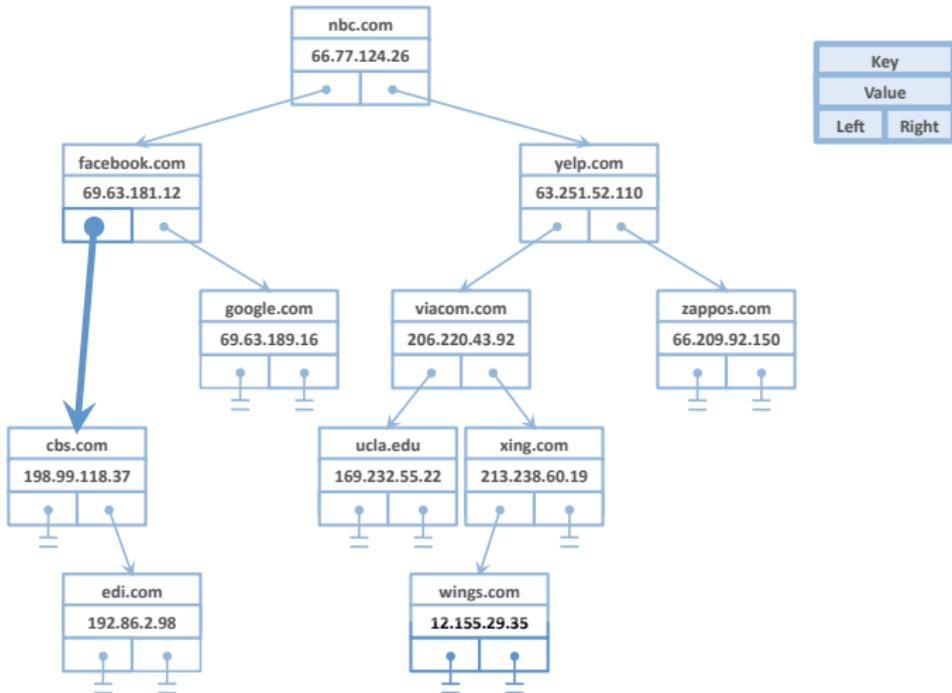
- Si j sólo tiene un subárbol derecho, se asigna $j.Right$ al campo ($Left$ o $Right$) que apunta a j en el nodo padre de j
- Si j sólo tiene un subárbol izquierdo, se asigna $j.Left$ al campo ($Left$ o $Right$) que apunta a j en el nodo padre de j



```
Delete (A_Map, ASU.To_Unbounded_String ("bbva.es"), Success);
```



```
Delete (A_Map, ASU.To_Unbounded_String ("bbva.es"), Success);
```

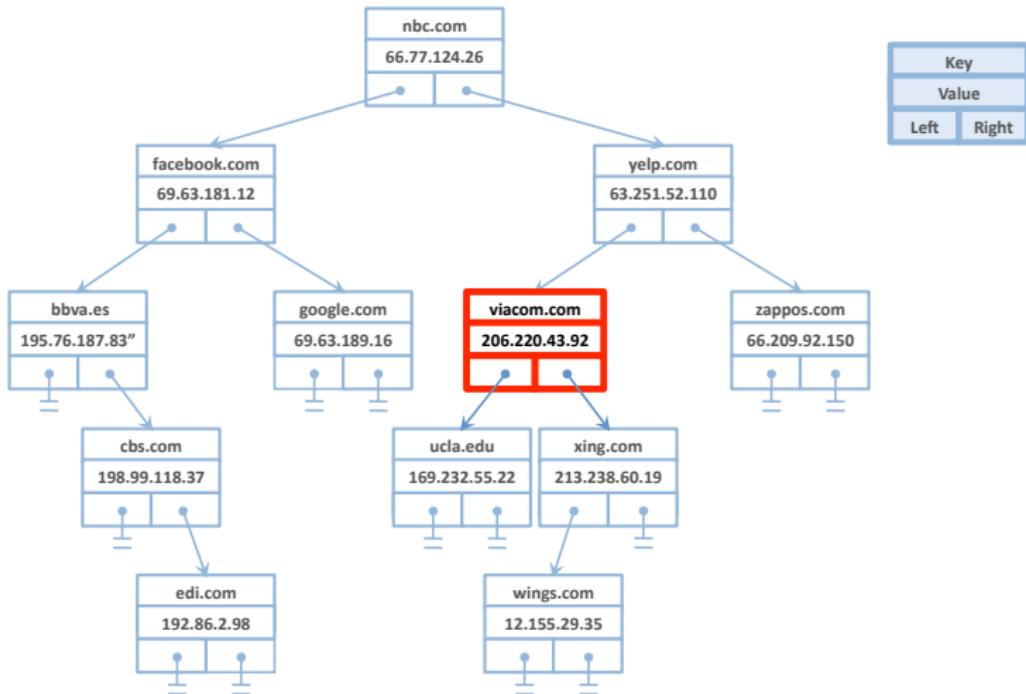


```
Delete (A_Map, ASU.To_Unbounded_String ("bbva.es"), Success);
```

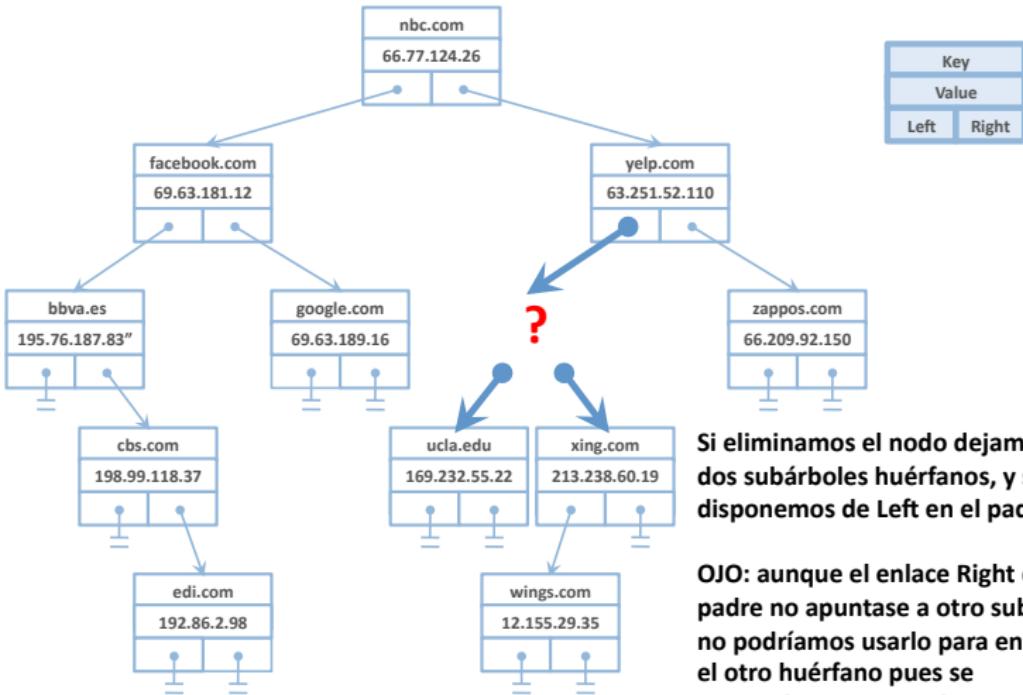
3. Borrado de un nodo j con dos subárboles hijos

Problema:

Hay un sólo enlace apuntando al nodo j que se borra en su nodo padre, pero j apunta a dos subárboles hijos.



```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```



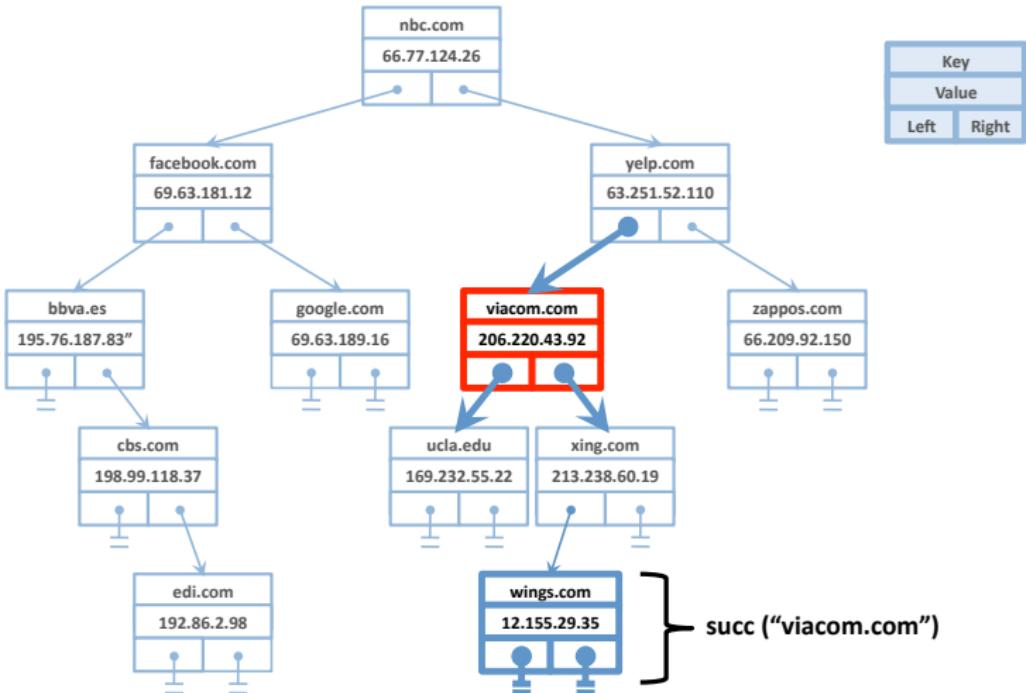
```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```

3. Borrado de un nodo j con dos subárboles hijos

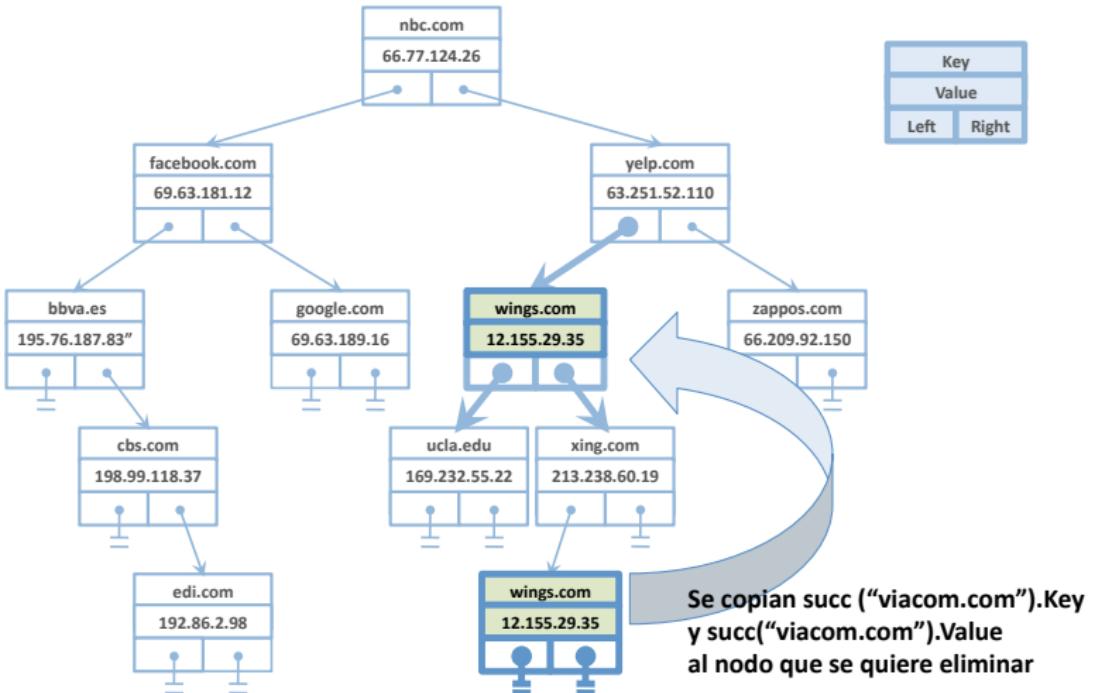
Solución: sustituir el nodo j que se quiere borrar por su **sucesor** y borrar el sucesor

- ① Se sustituye $j.(Key, Value)$ por $\text{succ}(j).(Key, Value)$, siendo $\text{succ}(j)$ el nodo cuya clave sucede a la de j en el árbol:
 - $j.Key < \text{succ}(j).Key$ y
 - no existe un nodo m tal que $j.Key < m.Key < \text{succ}(j).Key$
- ② Se borra $\text{succ}(j)$
 - El nodo $\text{succ}(j)$ no puede tener subárbol izquierdo, como mucho sólo tendrá derecho. Si tuviera un subárbol izquierdo no sería el $\text{succ}(j)$.

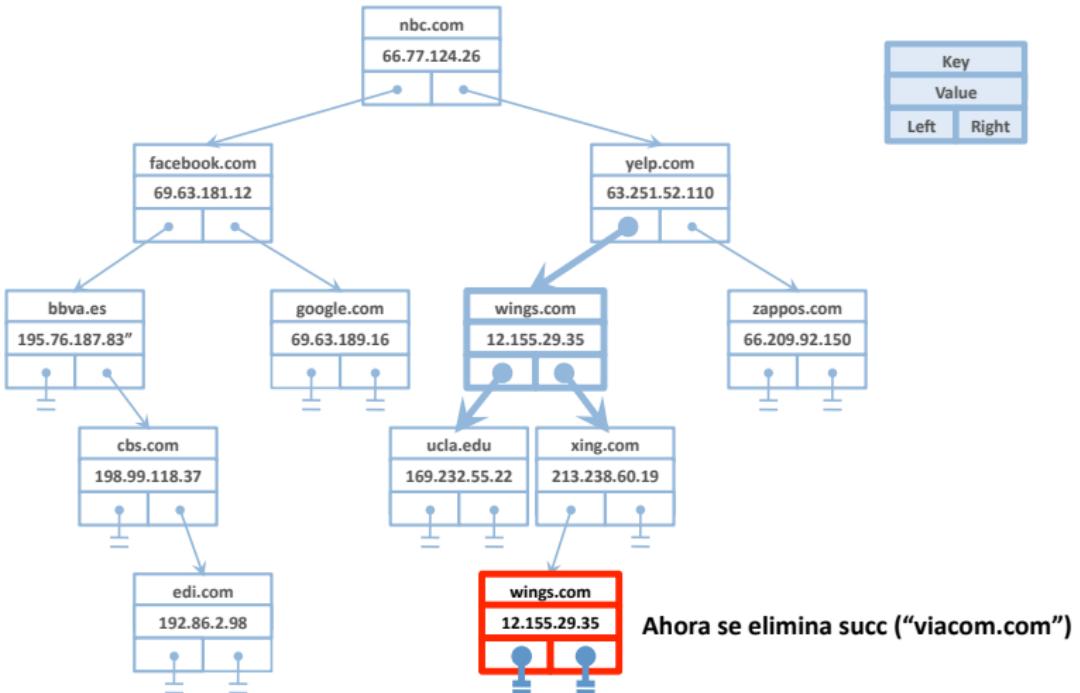
De esta forma se preserva el orden entre los nodos



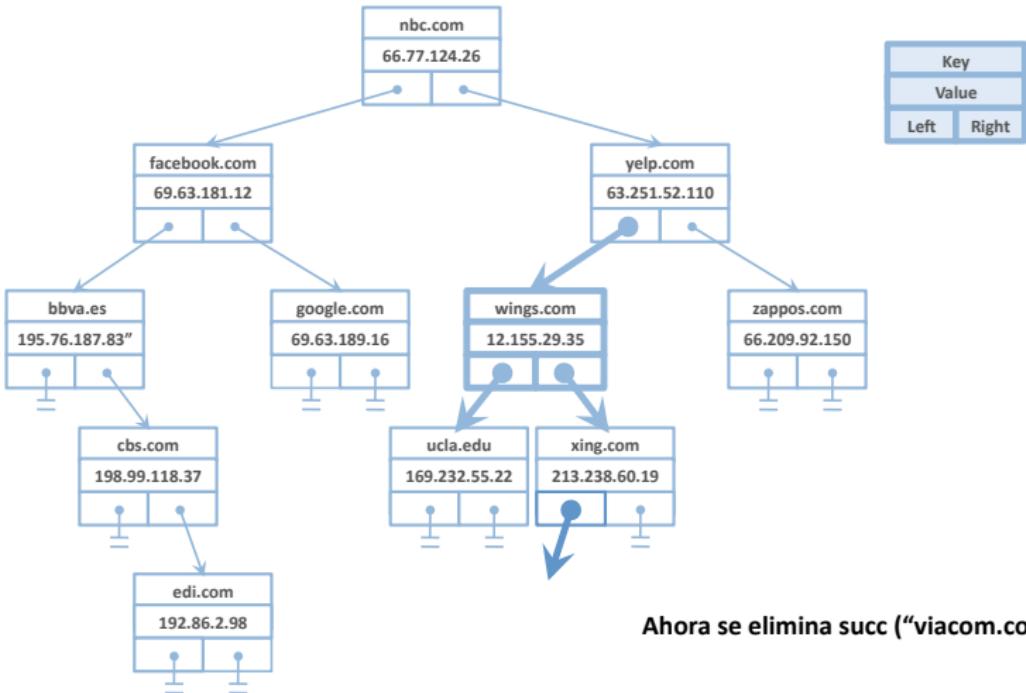
```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```



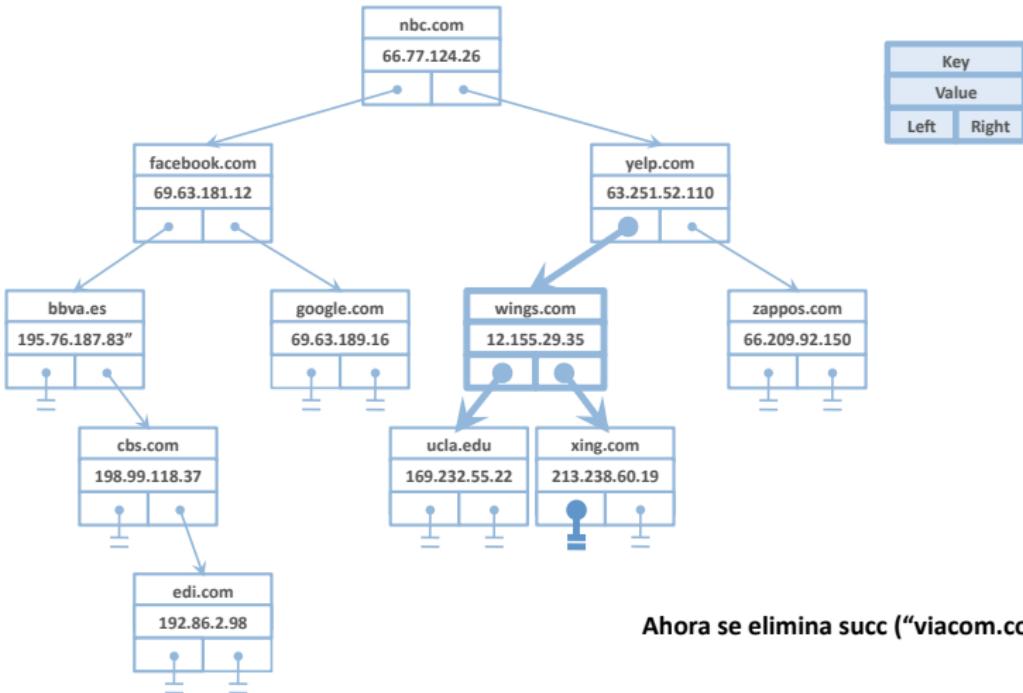
```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```



```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```



```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```



```
Delete (A_Map, ASU.To_Unbounded_String ("viacom.com"), Success);
```

3. Borrado de un nodo j con dos subárboles hijos

¿Cómo programamos $\text{succ}(j)$?

- $\text{succ}(j) = \min(j.\text{Right})$
 - $\text{succ}(j)$ es el nodo mínimo del subárbol derecho de j : el que tiene la menor clave
 - Para encontrar el nodo mínimo de un árbol se recorre recursivamente desde la raíz el árbol a través de los hijos izquierdos hasta encontrar un nodo que no tenga hijo izquierdo.

Borrado del nodo mínimo de un árbol

- El nodo mínimo de un árbol no tiene hijo izquierdo, por lo que ya sabemos cómo borrarlo

3. Borrado de un nodo j con dos subárboles hijos: Programación

```
package Maps is
    package ASU renames Ada.Strings.Unbounded;

    type Map is limited private;

    ...

    procedure Delete (M      : in out Map;
                      Key    : in  Asu.Unbounded_String;
                      Success : out Boolean);

private
    ...
end Maps;
```

3. Borrado de un nodo j con dos subárboles hijos: Programación

```
package body Maps is

    ...
    function Delete_Min (M : Map)  return Map  is
    begin
        ...
    end Delete_Min;

    procedure Delete (M          : in out Map;
                      Key       : in  Asu.Unbounded_String;
                      Success   : out Boolean) is
    begin
        ...
    end Delete;

end Maps;
```

Resumen

Tabla de Símbolos

- La **tabla de símbolos** es una estructura de datos que almacena elementos compuestos por parejas (**Clave, Valor**)
- **Clave** y **Valor** pueden ser tipos de datos cualesquiera
- Tiene tres operaciones básicas:
 - **Put**: Dado un nuevo elemento (**Clave, Valor**) como parámetro, se añade éste a la tabla. Si ya existía un elemento con la misma Clave, se sustituye su Valor asociado por el especificado en la llamada a Put
 - **Get**: Dada una Clave como parámetro, devuelve el Valor asociado a la misma en la tabla en caso de que exista un elemento (**Clave, Valor**)
 - **Delete**: Dada un Clave como parámetro, se borra de la tabla, si existe, el elemento (**Clave, Valor**)

Implementaciones de una tabla de símbolos

- Mediante un **Array no ordenado**
- Mediante una **Lista enlazada no ordenada**
- Mediante un **Array ordenado con búsqueda binaria**
- Mediante una **Lista enlazada ordenada**
- Mediante un **Árbol de búsqueda binaria**