1. Módulo DiccionarioString(α)

1.1. Interfaz

parámetros formales
géneros α
se explica con: DICCIONARIO(STRING, α).
géneros: $\texttt{dicc}_T(\alpha)$.

1.1.1. Operaciones básicas

CREARDICC() $\rightarrow$ *res* : $\texttt{dicc}_T(\alpha)$
Pre $\equiv$ {true}
Post $\equiv$ {*res* $=_{\text{obs}}$ vacio}
Complejidad: $O(1)$
Descripción: crea un diccionario vacio

DEFINIDO?(**in** c: `string`, **in** d: $\texttt{dicc}_T(\alpha)$) $\rightarrow$ *res* : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {*res* $=_{\text{obs}}$ def?(c, d)}
Complejidad: $O(|c|)$
Descripción: devuelve si la clave fue previamente definida en el diccionario

DEFINIR(**in** c: `string` , **in** s: α, **in/out** d: $\texttt{dicc}_T(\alpha)$)
Pre $\equiv$ {$d =_{\text{obs}} d_0$}
Post $\equiv$ {$d =_{\text{obs}}$ definir(c, s, d_0)}
Complejidad: $O(|c|+$ copy(s))
Descripción: define la clave c con el significado s en d

OBTENER(**in** c: `string`, **in** d: $\texttt{dicc}_T(\alpha)$) $\rightarrow$ *res* : α
Pre $\equiv$ {def?(c, d)}
Post $\equiv$ {alias($res =_{\text{obs}}$ obtener(c, d))}
Complejidad: $O(|c|)$
Descripción: devuelve el significado correspondiente a la clave en el diccinario
Aliasing: *res* es modificable si y solo si d es modificable.

Obs: copy es una función de α en que devuelve el costo de copiar un elemento del género α.

1.2. Representación

1.2.1. Estructura de Representación

$\texttt{dicc}_T(\alpha)$ **se representa con** `estr donde estr es puntero(nodo)`

donde `nodo` es `tupla`(*significado*: `puntero`(α) ,
caracteres: `arreglo[256] de puntero(nodo)`)

1.2.2. Invariante de Representación

(I) Todas las posiciones del arreglo de caracteres están definidas.

(II) No hay claves de 0 caracteres. Esto es, el nodo raiz tiene el campo significado `NULL`.

(III) No hay ciclos en el trie. Esto es, existe un número natural n tal que la cantidad de niveles del árbol está acotada por n.

(IV) Dado un nodo cualquiera del trie, existe un único camino desde la raíz hasta dicho nodo.

Rep : estr $\longrightarrow$ bool

$\text{Rep}(e) \equiv \text{true} \iff$
$(e \to \mathit{significado} = \text{NULL}) \wedge$
$(\forall i\text{: nat})(i < 256 \Rightarrow \text{definido?}(\text{e}\to \text{caracteres}, i)) \wedge_{\text{L}}$
$(\exists n\text{: nat})(\text{finaliza}(e, n)) \wedge_{\text{L}}$
$(\forall p, q\text{: puntero(nodo)})(p \in \text{punteros}(e) \wedge q \in (\text{punteros}(e) - \{p\}) \Rightarrow p \neq q)$

$\text{finaliza} : \text{estr } e \times \text{nat} \longrightarrow \text{bool}$ $\quad \{(\forall i\text{: nat})\ (i < 256 \Rightarrow \text{definido?}(e \to \mathit{caracteres}, i))\}$
$\text{finaliza}(e, n) \equiv n > 0 \wedge_{\text{L}} (e = \text{NULL} \vee_{\text{L}} \text{finalizaAux}(e \to \mathit{caracteres}, n-1, 0))$

$\text{finalizaAux} : \text{ad(puntero(nodo))}\ a \times \text{nat} \times \text{nat}\ k \longrightarrow \text{bool}$ $\quad \{k \leq \text{tam}(a)\}$
$\text{finalizaAux}(a, n, k) \equiv$ **if** $k = \text{tam}(a)$ **then** true **else** $\text{finaliza}(e \to \mathit{caracteres}[k], n) \wedge \text{finalizaAux}(a, n, k+1)$ **fi**

$\text{punteros} : \text{estr } e \longrightarrow \text{multiconj(puntero(nodo))}$
$\text{punteros}(e) \equiv$ **if** $e = \text{NULL}$ **then** $\emptyset$ **else** $\text{punterosAux}(e \to \mathit{caracteres}, 0)$ **fi**

$\text{punterosAux} : \text{ad(puntero(nodo))}\ a \times \text{nat}\ k \longrightarrow \text{multiconj(puntero(nodo))}$ $\quad \{k \leq \text{tam}(a)\}$
$\text{punterosAux}(a, k) \equiv$ **if** $k = \text{tam}(a)$ **then**
$\emptyset$
else
(**if** $a[k] = \text{NULL}$ **then** $\emptyset$ **else** $\text{Ag}(a[k], \text{punteros}(a[k]))$ **fi**) $\cup$ $\text{punterosAux}(a, k+1)$
fi

1.2.3. Función de Abstracción

$\text{Abs} : \text{estr } e \longrightarrow \text{dicc}_T(\alpha)$ $\quad \{\text{Rep}(e)\}$
$\text{Abs}(e) =_{\text{obs}} \text{d: dicc}_T(\alpha) \mid (\forall c\text{: string})(\text{def?}(c, d) = \text{esClave?}(c, e) \wedge_{\text{L}}$
$(\text{def?}(c, d) \Rightarrow_{\text{L}} \text{obtener}(c, d) = \text{significado}(c, e)))$

$\text{esClave?} : \text{string } c \times \text{estr } e \longrightarrow \text{bool}$ $\quad \{\text{Rep}(e)\}$
$\text{esClave?}(c, e) \equiv$ **if** $\text{vacía?}(c)$ **then**
$e \to \mathit{significado} \neq \text{NULL}$
else
$e \to \mathit{caracteres}[\text{ord}(\text{prim}(c))] \neq \text{NULL} \wedge_{\text{L}} \text{esClave?}(\text{fin}(c), e \to \mathit{caracteres}[\text{ord}(\text{prim}(c))])$
fi
$\text{significado} : \text{string } c \times \text{estr } e \longrightarrow \alpha$ $\quad \{\text{Rep}(e) \wedge \text{esClave?}(c, e)\}$
$\text{significado}(c, e) \equiv$ **if** $\text{vacía?}(c)$ **then** $*(e \to \mathit{significado})$ **else** $\text{significado}(\text{fin}(c), e \to \mathit{caracteres}[\text{ord}(\text{prim}(c))])$ **fi**

1.3. Algoritmos

iCrearDicc () → res: estr

```
(res→ significado) ← NULL                                   O(1)
(res→ caracteres) ← CrearArreglo(256)                       O(1)
for i ← 0 to 255 do                                         O(1)
    (res→ caracteres[i]) ← NULL                             O(1)
end for
```

Complejidad : $O(1)$

iDefinido? (**in** c: `string`, **in** d: `estr`) → res: bool

```
nat i ← 0                                                   O(1)
bool esta ← true                                            O(1)
puntero(nodo) actual ← d                                    O(1)
while i < Longitud(c) ∧L esta do                            O(|c|)
    if actual→ caracteres[ord(c[i])] = NULL then            O(1)
```

```
            esta ← false                                                    O(1)
        end if
        actual ← (actual → caracteres[ord(c[i])])                           O(1)
        i ← i + 1                                                           O(1)
    end while
    res ← (esta ∧_L ¬(actual→ significado = NULL))                          O(1)
```

Complejidad : $O(|c|)$

iDefinir (**in** c: `string`, **in** s: α, **in/out** d: `estr`)

```
    nat i ← 0                                                               O(1)
    puntero(nodo) actual ← d                                                O(1)
    while i < Longitud(c) do                                                O(|c|)
        if actual→ caracteres[ord(c[i])] = NULL then                        O(1)
            (actual→ caracteres[ord(c[i])]) ← CrearDicc()                   O(1)
        end if
        actual ← (actual→ caracteres[ord(c[i])])                            O(1)
        i ← i + 1                                                           O(1)
    end while
    (actual→ significado) ← &Copiar(s)                                      O(copy(s))
```

Complejidad : $O(|c| + copy(s))$

iObtener (**in** c: `string`, **in** d: `estr`) → res: α

```
    nat i ← 0                                                               O(1)
    puntero(nodo) actual ← d                                                O(1)
    while i < Longitud(c) do                                                O(|c|)
        actual ← (actual→ caracteres[ord(c[i])])                            O(1)
        i ← i + 1                                                           O(1)
    end while
    res ← *(actual→ significado)                                            O(1)
```

Complejidad : $O(|c|)$

1.4. Servicios Usados

α debe proveer la operación:
COPIAR(**in** s: α) → res : α
Pre ≡ {true}
Post ≡ {$res =_{\text{obs}} s$}
Complejidad: O(copy(s))
Donde se copia s, de modo que no haya aliasing entre s y res