1. Módulo DiccClavesRapidas

Interfaz

```
\begin{array}{ll} \mathbf{parametros} \ \mathbf{formales} \\ \mathbf{g\acute{e}nero} & significado \\ \mathbf{funci\acute{o}n} & \mathrm{Copiar}(\mathbf{in} \ a: significado) \rightarrow res: significado \\ \mathbf{Pre} \equiv \{\mathrm{true}\} \\ & \mathbf{Post} \equiv \{res =_{\mathrm{obs}} a\} \\ & \mathbf{Complejidad:} \ \Theta(copy(a)) \\ & \mathbf{Descripci\acute{o}n:} \ \mathrm{funci\acute{o}n} \ \mathrm{de} \ copia \ \mathrm{de} \ significado \\ \mathbf{se} \ \mathbf{explica} \ \mathbf{con:} \ \mathrm{Diccionario}(\mathrm{String}, \ \mathrm{Significado}) \\ \mathbf{g\acute{e}neros:} \ \mathrm{dcr} \end{array}
```

El modulo funciona como un diccionario, pero solo se utiliza con claves del tipo string.

Operaciones básicas de DiccClavesRapidas

```
VACIO() \rightarrow res : dcr
\mathbf{Pre} \equiv \{ \text{true} \}
\mathbf{Post} \equiv \{res =_{obs} vacio\}
Complejidad: O(1)
Descripción: Crea un diccionario vacio.
DEFINIR(in n: string, in s: significado, in/out d: dcr)
\mathbf{Pre} \equiv \{d =_{\mathrm{obs}} d_0\}
\mathbf{Post} \equiv \{d =_{obs} \operatorname{definir}(n, s, d_0)\}\
Complejidad: O(Longitud(n))
Descripción: Define la clave n con significado s en el diccionario d.
DEF?(in n: string, in d: dcr) \rightarrow res: bool
\mathbf{Pre} \equiv \{ \text{true} \}
\mathbf{Post} \equiv \{res =_{obs} def?(n, d)\}
Complejidad: O(Longitud(n))
Descripción: Devuelve true si el string n esta definido en el diccionario d.
OBTENER(in s: string, in d: dcr) \rightarrow res: significado
\mathbf{Pre} \equiv \{ \operatorname{def}?(n,d) \}
\mathbf{Post} \equiv \{res =_{obs} obtener(p, m) \land alias(res, obtener(p, n))\}\
Complejidad: O(Longitud(n))
Descripción: Retorna el significado de la clave n.
Aliasing: res devuelve el significado por referencia.
BORRAR(in s: string, in/out d: dcr)
\mathbf{Pre} \equiv \{ \operatorname{def}?(p,m) \wedge d =_{obs} d_0 \}
\mathbf{Post} \equiv \{d =_{obs} borrar(p, d_0)\}\
Complejidad: O(Longitud(n))
Descripción: Elimina la clave n y su significado del diccionario d.
CLAVES(in d: dcr) \rightarrow res: conj(string)
\mathbf{Pre} \equiv \{ \mathrm{true} \}
\mathbf{Post} \equiv \{res =_{obs} claves(d) \land alias(res =_{obs} claves(d))\}\
Complejidad: O(1)
Descripción: Devuelve el conjunto de claves del diccionario d
Aliasing: res no es modificable. Si se definen nuevas claves, res se invalida.
```

Representación

Representación de DiccClavesRapidas

```
dcr se representa con estr
```

```
 \begin{array}{c} {\rm donde\ estr\ es\ tupla} \big( dicc: {\rm arreglo}({\rm puntero}({\rm nodo}))\ ,\ claves:\ {\rm conjunto}({\rm string})\ \big) \\ {\rm donde\ nodo\ es\ tupla} \big( id:\ {\rm char}\ ,\ clave:\ {\rm string}\ ,\ definido:\ {\rm bool}\ ,\ dato:\ {\rm significado}\ ,\ sig:\ {\rm arreglo}({\rm puntero}({\rm nodo})) \big) \\ \end{array}
```

Invariante de representacion en castellano:

- 1. Toda clave de claves esta definida en dicc, es decir, dicc posee un nodo con la misma clave.
- 2. Todo nodo donde su .definido es true, su .clave esta contenida en claves.

```
Rep : estr \longrightarrow bool
Rep(e) \equiv true \iff
                    1. (\forall s : string) \ s \in e.claves \Rightarrow (hayCamino(s, e.dicc)) \land_L \Pi_3(tuplaEnDicc(s, e.dicc))) \land
                    2. (\forall s : string) (hay Camino(s, e.dicc) \land_L \Pi_3(tupla EnDicc(s, e.dicc)) \Rightarrow s \in e.claves
Abs : estr e \longrightarrow \text{dicc}(string, significado)
                                                                                                                                                      \{\operatorname{Rep}(e)\}
Abs(e) \equiv d : dicc(string, significado)
                (\forall \ s : \text{string}) \ \text{def?}(s, \, \text{e.claves}) =_{\text{obs}} \text{def?}(p, d) \ \land \\
                (\forall s : \text{string}) \text{ def?}(s, \text{e.claves}) \Rightarrow_{\text{L}} \Pi_4(\text{tuplaEnDicc}(s, \text{e.dicc})) =_{\text{obs}} \text{obtener}(s, d))
Operaciones auxiliares de TAD:
hay<br/>Camino : string s \times \operatorname{arreglo} \operatorname{array} \longrightarrow \operatorname{bool}
                                                                                                                                       \{tam(array) = 256\}
hayCamino(s, e.dicc) \equiv if Longitud(s) = 0 then
                                         true
                                    else
                                        if definido(ord(prim(s)), array) then
                                             hayCamino(fin(s), array[ord(prim(s))])
                                         else
                                             false
                                        fi
                                    fi
tuplaEnDicc : string s \times \text{arreglo } array \longrightarrow \text{tupla}(\alpha_1...\alpha_n)
                                                                                                   \{\text{hayCamino}(s, array) \land \text{tam}(\text{array}) = 256\}
tuplaEnDicc(s, e.dicc) \equiv if Longitud(s) = 1 then
```

```
\begin{aligned} \text{blaEnDicc} &: \text{string } s \times \text{arreglo } array &\longrightarrow \text{tupla}(\alpha_1...\alpha_n) & \{\text{hayCamino}(s, array) \wedge \text{tam}(\text{array}) = 256 \\ \text{blaEnDicc}(s, \text{e.dicc}) &\equiv & \textbf{if } \text{Longitud}(s) = 1 & \textbf{then} \\ & & \text{array}[\text{ord}(\text{prim}(s))] \\ & & \textbf{else} \\ & & \text{tuplaEnDicc}(\text{fin}(s), \text{array}[\text{ord}(\text{prim}(s))] \\ & \textbf{fi} \end{aligned}
```

Algoritmos

Lista de algoritmos

```
1.
2.
3.
4.
5.
 iVacio() \rightarrow res: estr
 begin
                                                                                                                        //O(1)
     res.dicc \leftarrow CrearArreglo(256)
     res.claves \leftarrow Vacio()
                                                                                                                        //O(1)
     return res
 end
 Complejidad: O(1)
                                                   Algoritmo 1: Vacio
 iDefinir(in \ n: string, in \ s: significado, in/out \ e: estr)
 begin
     var t : string; p : puntero(nodo) tuplaVacia : nodo
                                                                                                                        //O(1)
     t \leftarrow fin(n)
     tuplaVacia \leftarrow \langle false, s, CrearArreglo(256) \rangle
                                                                                                                        //O(1)
     if \neg Def?(n, e.claves) then
                                                                                                                          //O()
         if \neg Definido(e.dicc, Ord(Prim(t))) then
                                                                                                                        //O(1)
          | e.dicc[Ord(Prim(t))] \leftarrow tuplaVacia
         end
                                                                                                                        //O(1)
                                                                                                                        //O(1)
         p \leftarrow e.dicc[Ord(Prim(t))]
         while Longitud(t) > \theta do
                                                                                                         //O(Longitud(n))
            if \neg Definido(p.sig, Ord(Prim(t))) then
                                                                                                                        //O(1)
              | p.sig[Ord(Prim(t))] \leftarrow tuplaVacia
             \mathbf{end}
                                                                                                                        //O(1)
             p \leftarrow p[Ord(Prim(t))]
                                                                                                                        //O(1)
            t \leftarrow fin(t)
                                                                                                                        //O(1)
         \mathbf{end}
         p.definido \leftarrow true
                                                                                                                        //O(1)
                                                                                                                        //O(1)
         p.dato \leftarrow S
         Agregar(n, e.claves)
                                                                                                                        //O(1)
     \mathbf{end}
 end
 Complejidad: O(Longitud(n))
```

3

4

4

4

Algoritmo 2: Definir

Comentarios: Agregar(n, e.claves) es O(1) dado que se sabe por el if que no esta definido.

```
i\mathrm{Def}?(in n: string, in e: estr) \to res: bool
begin
| res \leftarrow Def?(n, e.claves)
                                                                                                             //O(1) return res
\mathbf{end}
Complejidad: O(1)
                                                    Algoritmo 3: Def?
iObtener(in n: string, in e: estr) \rightarrow res: significado
begin
    var t : string; p : puntero(nodo)
    t \leftarrow fin(n)
                                                                                                                           //O(1)
                                                                                                                            //O(1)
    p \leftarrow e.dicc[prim(n)]
    while Longitud(t) > \theta do
                                                                                                            //O(Longitud(n))
        p \leftarrow p[Ord(\dot{P}\dot{r}im(t))]
                                                                                                                           //O(1)
        t \leftarrow fin(t)
                                                                                                                           //O(1)
    \mathbf{end}
   res \leftarrow p.dato \ \mathbf{return} \ \mathit{res}
\mathbf{end}
Complejidad: O(Longitud(n))
                                                  Algoritmo 4: Obtener
iBorrar(in s: string, in/out d: dcr)
begin
    var t : string; p : puntero(nodo)
   t \leftarrow fin(n)
                                                                                                                           //O(1)
   p \leftarrow e.dicc[prim(n)]
                                                                                                                            //O(1)
    while Longitud(t) > 0 do
                                                                                                            //O(Longitud(n))
        p \leftarrow p[Ord(Prim(t))]
                                                                                                                           //O(1)
       t \leftarrow fin(t)
                                                                                                                           //O(1)
    \quad \text{end} \quad
   p.definido \leftarrow false
\mathbf{end}
Complejidad: O(Longitud(n))
                                                   Algoritmo 5: Borrar
iClaves(in e: estr) \rightarrow res: conj(string)
begin
\vdash res \leftarrow e.claves
end
Complejidad: O(1)
                                                   Algoritmo 6: Claves
```