

ESTRUCTURAS DE DATOS

DICCIONARIOS

El TAD Diccionario

Manuel Montenegro Montes

Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

Motivación

- Leer un texto de entrada e imprimir el número de veces que aparece cada palabra.

David tomó la llave para entregársela a Laura. Esta última, no obstante, declinó hacer uso de la llave mientras que no fuera absolutamente necesario.

David 1
tomó 1
la 2
llave 2
para 1
...

- ¿Cómo almacenamos las palabras que nos encontramos?

Motivación

- Tabla en la que almacenamos el número de veces que aparece cada palabra encontrada hasta el momento.
- Con cada palabra recibida:
 - Si existe una entrada en la tabla con esa palabra, incrementamos su contador.
 - Si no, insertamos una nueva entrada con esa palabra y con su contador a 1.

Palabra	Contador
“David”	1
“tomó”	1
“la”	2
“llave”	2
...	...

¿Qué es un diccionario?

- Un tipo abstracto de datos que almacena un conjunto de pares.
- A cada par se le llama **entrada**.
- A la primera componente de cada par se le denomina **clave**.
- A la segunda componente se le denomina **valor asociado** a esa clave.
- No existen dos pares con la misma clave.

Palabra	Contador
"David"	1
"tomó"	1
"la"	2
"llave"	2
...	...

Claves

Valores

Terminología

diccionarios

tablas

arrays asociativos

maps

associative arrays

dictionaries

symbol tables

Palabra	Contador
"David"	1
"tomó"	1
"la"	2
"llave"	2
...	...

Modelo conceptual de diccionarios

- Sean:
 - K – conjunto de claves
 - V – conjunto de valores
- Un diccionario M es un conjunto de pares (k, v) , donde $k \in K$, $v \in V$.
- No existen pares $(k, v), (k, v') \in M$ tales que $v \neq v'$.

Palabra	Contador
“David”	1
“tomó”	1
“la”	2
“llave”	2
...	...

$$M = \{ (“David”, 1), (“tomó”, 1), (“la”, 2), \dots \}$$

Operaciones en el TAD Diccionario

- Constructoras:
 - Crear un diccionario vacío: ***create_empty***
- Mutadoras:
 - Añadir una entrada al diccionario: ***insert***
 - Eliminar una entrada del diccionario: ***erase***
- Observadoras:
 - Saber si existe una entrada con una clave determinada: ***contains***
 - Saber el valor asociado con una clave: ***at***
 - Saber si el diccionario está vacío: ***empty***
 - Saber el número de entradas del diccionario: ***size***

Operaciones constructoras y mutadoras

$\{ \text{true} \}$

create_empty() $\rightarrow (M: \text{Map})$

$\{ M = \emptyset \}$

$\{ \text{true} \}$

insert($k: \text{Key}, v: \text{Value}, M: \text{Map}$)

$M = \begin{cases} \text{old}(M) & \text{si } \exists v'. (k, v') \in \text{old}(M) \\ \text{old}(M) \cup \{(k, v)\} & \text{en otro caso} \end{cases}$

$\{ \text{true} \}$

erase($k: \text{Key}, M: \text{Map}$)

$M = \{ (k', v') \in \text{old}(M) \mid k' \neq k \}$

Operaciones observadoras

$\{ \text{true} \}$

contains(k : Key, M : Map) \rightarrow (b : bool)

$\{ b \Leftrightarrow \exists v. (k, v) \in M \}$

$\{ \exists v'. (k, v') \in M \}$

at(k : Key, M : Map) \rightarrow (v : value)

$\{ (k, v) \in M \}$

$\{ \text{true} \}$

empty(M : Map) \rightarrow (b : bool)

$\{ b \Leftrightarrow M = \emptyset \}$

$\{ \text{true} \}$

size(M : Map) \rightarrow (n : int)

$\{ n = |M| \}$

Interfaz en C++

```
template <typename K, typename V>
class map {
public:
    map();
    map(const map &other);
    ~map();

    void insert(const K &key, const V &value);
    void erase(const K &key);

    bool contains(const K &key) const;

    const V & at(const K &key) const;
    V & at(const K &key);

    int size() const;
    bool empty() const;

private:
    // ...
};
```



Interfaz en C++

```
template <typename K, typename V>
class map {
public:
    map();
    map(const map &other);
    ~map();

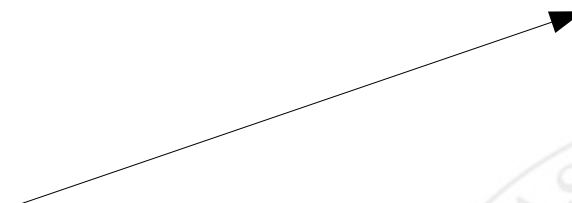
    void insert(const map_entry &entry);
    void erase(const K &key);

    bool contains(const K &key) const;

    const V & at(const K &key) const;
    V & at(const K &key);

    int size() const;
    bool empty() const;

private:
    // ...
};
```



```
struct map_entry {
    K key;
    V value;
};
```

Ejemplo

```
map<string, int> personas;
```

```
personas.insert({"Aarón", 42});  
personas.insert({"Estela", 41});
```

```
cout << personas.contains("Aarón") << endl;  
cout << personas.at("Aarón") << endl;
```

```
personas.insert({"Carlos", 31});
```

```
personas.erase("Estela");
```

```
personas.at("Aarón") = 43;
```

personas = \emptyset

personas = {("Aarón", 42), ("Estela", 41) }

true
42

personas = {("Aarón", 42), ("Carlos", 31), ("Estela", 41) }

personas = {("Aarón", 42), ("Carlos", 31) }

personas = {("Aarón", 43), ("Carlos", 31) }

Ejemplo

```
string palabra;  
map<string, int> dicc;  
  
cin >> palabra;  
  
while (!cin.eof()) {  
    if (dicc.contains(palabra)) {  
        dicc.at(palabra)++;  
    } else {  
        dicc.insert({palabra, 1});  
    }  
  
    cin >> palabra;  
}
```



Dos implementaciones

- Mediante **árboles binarios de búsqueda** (MapTree)
- Mediante **tablas *hash*** (MapHash)

