

Unidad 7:

Programación Orientada a Objetos - Parte 3

Relaciones de agregación

<http://bit.ly/2HRBWgq>

Profesores:

Ernesto Cuadros- Vargas, PhD. ecuadros@utec.edu.pe

María Hilda Bermejo, M. Sc. mbermejo@utec.edu.pe

Telegram:

1. Configurar tu cuenta

2. <http://bit.ly/2TJnwBq>

ENCUESTA P001:

Ingresar al siguiente link:

link → <http://bit.ly/31y3pet>

Curso: Programación orientada a objetos 1

Profesor: Cuadros Vargas, Ernesto

Logro de la sesión:

Al finalizar la sesión, los alumnos diseñan POO, utilizando, archivos mapas y relaciones de agregación

- **Uso de archivos, mapas y multimapas**
- **Relaciones de agregación**

Archivos

Mapas

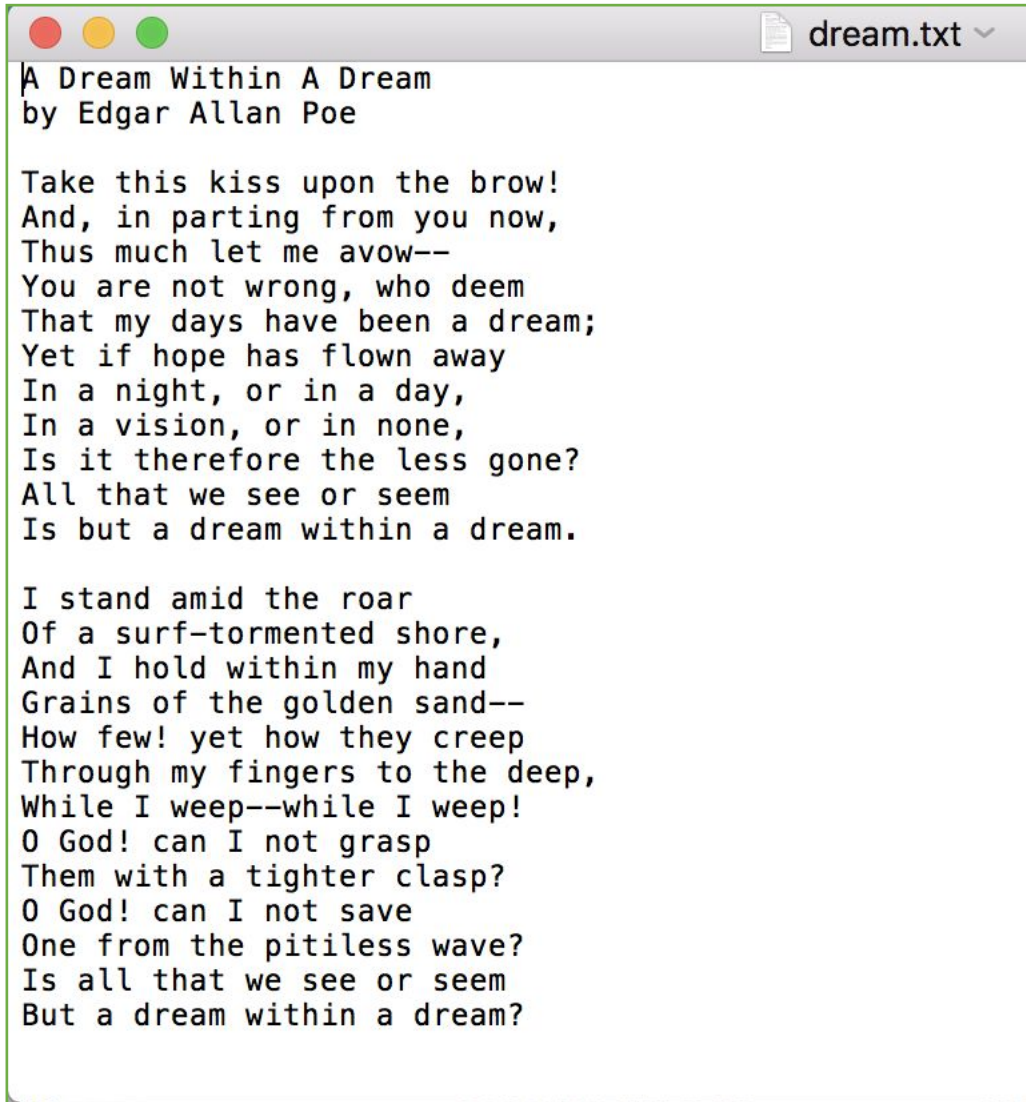
Multimaps

Ejemplo 1:

Escribir un programa que permita contar cuántas veces aparece cada letra del alfabeto inglés en un archivo texto. El programa leerá como dato el nombre físico del archivo texto.

Archivos texto para probar el programa:

dream.txt

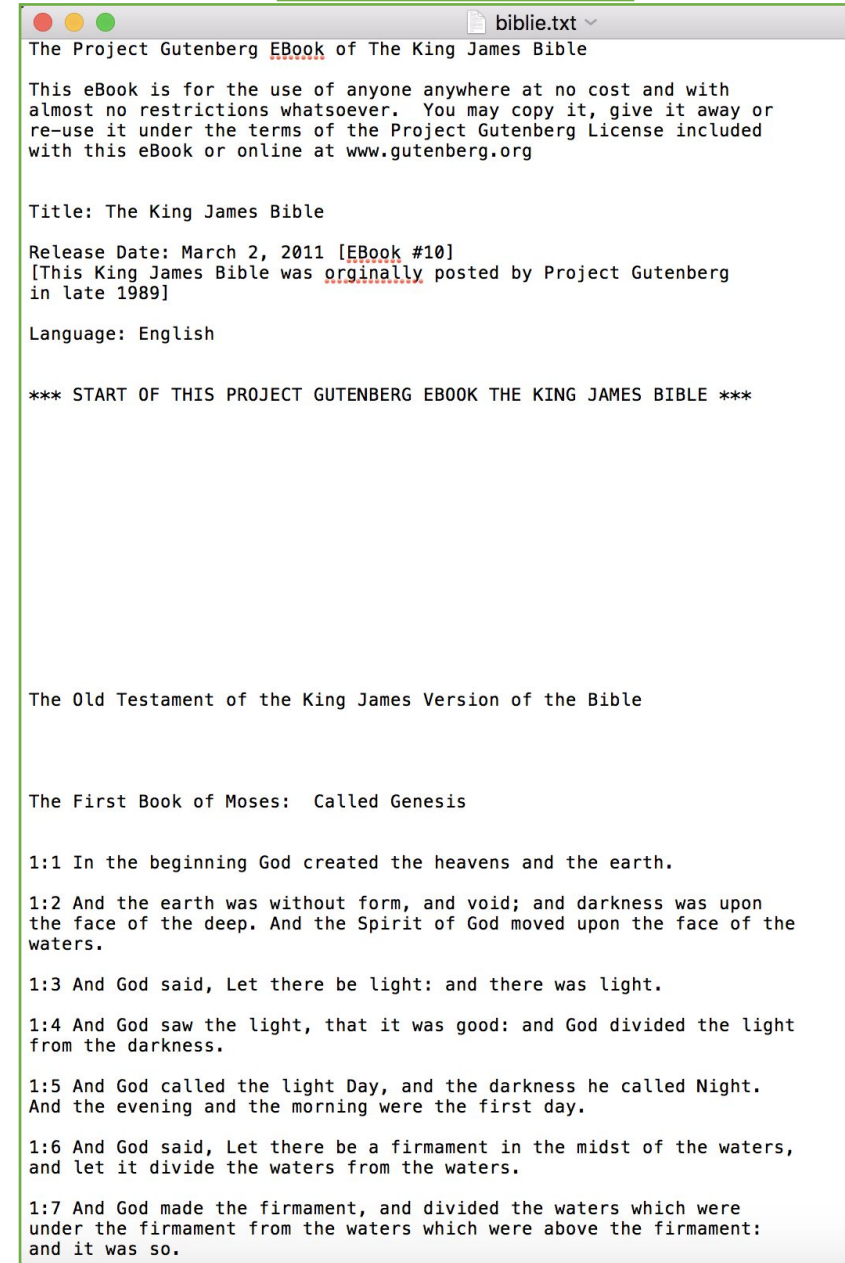


```
A Dream Within A Dream
by Edgar Allan Poe

Take this kiss upon the brow!
And, in parting from you now,
Thus much let me avow--
You are not wrong, who deem
That my days have been a dream;
Yet if hope has flown away
In a night, or in a day,
In a vision, or in none,
Is it therefore the less gone?
All that we see or seem
Is but a dream within a dream.

I stand amid the roar
Of a surf-tormented shore,
And I hold within my hand
Grains of the golden sand--
How few! yet how they creep
Through my fingers to the deep,
While I weep--while I weep!
O God! can I not grasp
Them with a tighter clasp?
O God! can I not save
One from the pitiless wave?
Is all that we see or seem
But a dream within a dream?
```

biblie.txt



```
The Project Gutenberg EBook of The King James Bible

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever. You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org

Title: The King James Bible

Release Date: March 2, 2011 [EBook #10]
[This King James Bible was originally posted by Project Gutenberg
in late 1989]

Language: English

*** START OF THIS PROJECT GUTENBERG EBOOK THE KING JAMES BIBLE ***

The Old Testament of the King James Version of the Bible

The First Book of Moses: Called Genesis

1:1 In the beginning God created the heavens and the earth.

1:2 And the earth was without form, and void; and darkness was upon
the face of the deep. And the Spirit of God moved upon the face of the
waters.

1:3 And God said, Let there be light: and there was light.

1:4 And God saw the light, that it was good: and God divided the light
from the darkness.

1:5 And God called the light Day, and the darkness he called Night.
And the evening and the morning were the first day.

1:6 And God said, Let there be a firmament in the midst of the waters,
and let it divide the waters from the waters.

1:7 And God made the firmament, and divided the waters which were
under the firmament from the waters which were above the firmament:
and it was so.
```

Link a archivos :

Archivo: Biblie.txt <http://bit.ly/2ZCrWgF>

Archivo: dream.txt <http://bit.ly/2FhmO9N>

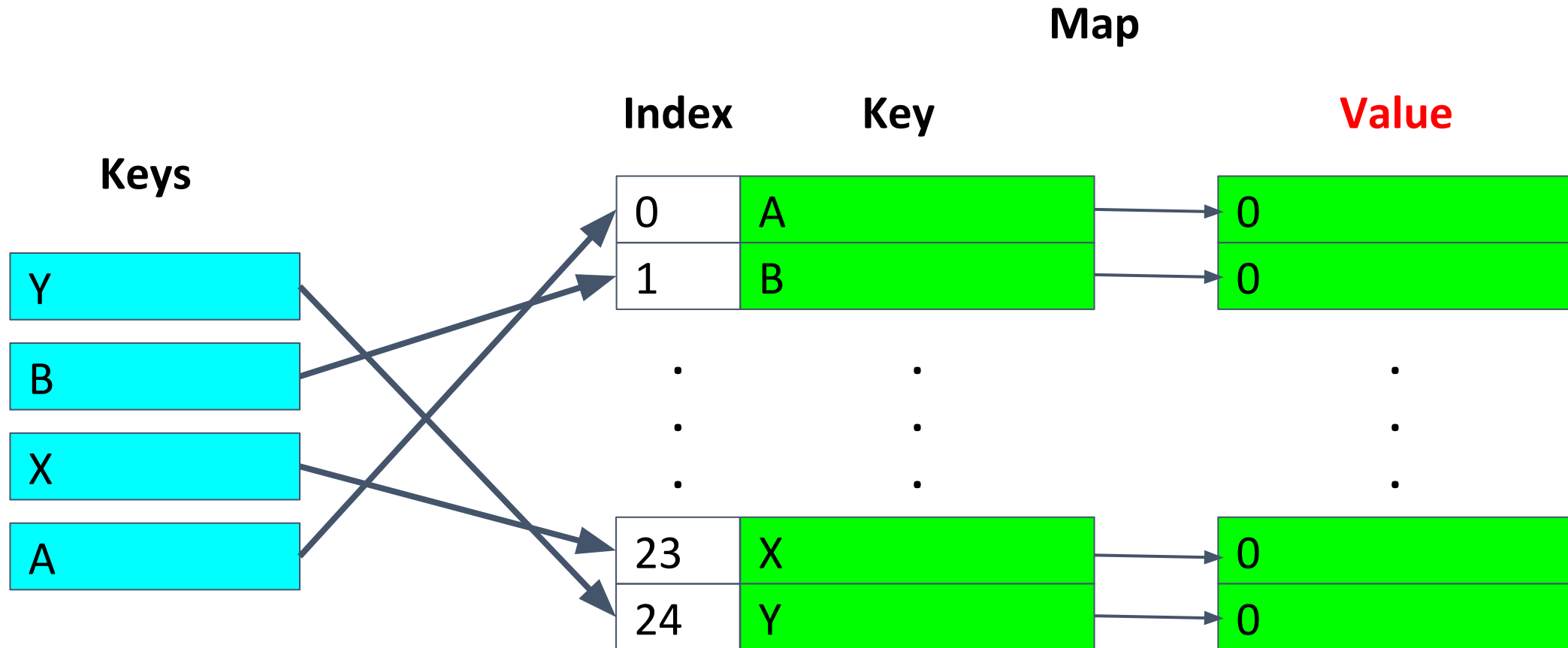
Ejemplo 1: archivo texto: [dream.txt](#)

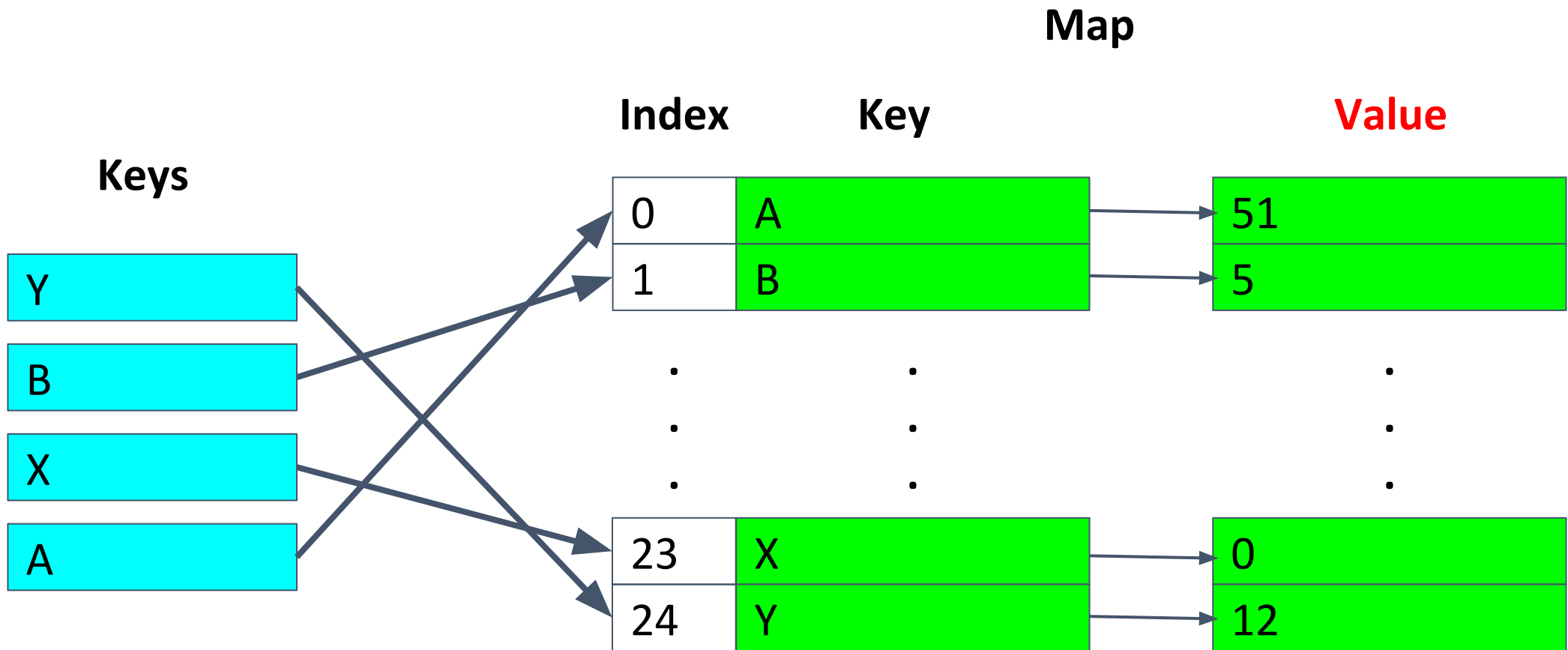
```
"/Users/hildabermejo/Documents/2018-2 Cursos
Nombre del archivo: dream.txt
Letra A esta 51 veces
Letra B esta 5 veces
Letra C esta 5 veces
Letra D esta 23 veces
Letra E esta 66 veces
Letra F esta 10 veces
Letra G esta 13 veces
Letra H esta 35 veces
Letra I esta 39 veces
Letra J esta 0 veces
Letra K esta 2 veces
Letra L esta 15 veces
Letra M esta 20 veces
Letra N esta 36 veces
Letra O esta 40 veces
Letra P esta 11 veces
Letra Q esta 0 veces
Letra R esta 31 veces
Letra S esta 27 veces
Letra T esta 42 veces
Letra U esta 9 veces
Letra V esta 5 veces
Letra W esta 22 veces
Letra X esta 0 veces
Letra Y esta 12 veces
Letra Z esta 0 veces
```

Ejemplo 2: archivo texto: [biblie.txt](#)

```
"/Users/hildabermejo/Documents/2018-2
Nombre del archivo: biblie.txt
Letra A esta 275727 veces
Letra B esta 48875 veces
Letra C esta 55067 veces
Letra D esta 158094 veces
Letra E esta 412232 veces
Letra F esta 83543 veces
Letra G esta 55301 veces
Letra H esta 282678 veces
Letra I esta 193959 veces
Letra J esta 8889 veces
Letra K esta 22292 veces
Letra L esta 129938 veces
Letra M esta 79940 veces
Letra N esta 225055 veces
Letra O esta 243185 veces
Letra P esta 43254 veces
Letra Q esta 964 veces
Letra R esta 170327 veces
Letra S esta 190029 veces
Letra T esta 317744 veces
Letra U esta 83473 veces
Letra V esta 30365 veces
Letra W esta 65487 veces
Letra X esta 1478 veces
Letra Y esta 58576 veces
Letra Z esta 2972 veces
```

Utilizando mapas:





main.cpp

```
#include <iostream>
#include "Letras.h"

using namespace std;

int main()
{
    map<char,int> elMapa;
    string ruta;

    cout<<"Nombre de Archivo: ";
    cin>>ruta;
    string texto = leerTexto(ruta);
    ContarCaracteresEnMapa(elMapa,texto);
    ImprimirMapa(elMapa);
    return 0;
}
```

Letras.h

```
#ifndef CUENTALETRASCONMAPAS_LETRAS_H
#define CUENTALETRASCONMAPAS_LETRAS_H
#include <iostream>
#include <map>
#include <cctype>
#include <fstream>
using namespace std;

string leerTexto(string ruta);
void ContarCaracteresEnMapa( map<char,int> &contador, string& texto);
void ImprimirMapa(map<char,int>& contador);

#endif //CUENTALETRASCONMAPAS_LETRAS_H
```

Letras.cpp

```
#include "Letras.h"
string leerTexto(string ruta)
{ //--Retorna un string con el contenido del archivo
    ifstream archivo;
    archivo.open(ruta);
    //--- validando archivo
    if (archivo.fail()) {
        cout << "Error: No puede abrirse el archivo\n";
        exit(1);
    }
    //--- leyendo todo el archivo en un texto
    string texto((istreambuf_iterator<char>(archivo)), istreambuf_iterator<char>());
    archivo.close();
    return texto;
}

void ContarCaracteresEnMapa( map<char,int> &item, string& texto)
{//-- recibe el string y realiza el conteo
    for(auto &i: texto)
        if(isalpha(i))
            contador[toupper(i)]++;
}

void ImprimirMapa(map<char,int> &item)
{//---- imprime el mapa-----
    for(auto &i: contador)
        cout<<"La letra "<<i.first<<" está "<<i.second<<" veces\n";
}
```

Ejemplo 2:

El programa halla las 20 palabras que con mayor frecuencia aparecen en un archivo.

Algoritmo:

1. Leer el contenido del archivo aun string
2. Depurar del string los signos de puntuación y otros caracteres tales como: ? () tabuladores
3. Colocar palabra a palabra en un mapa
4. Crear un multimapa
5. Imprimir el multimapa desde la última posición hacia la primera posición

Main.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <iterator>
#include <algorithm>
#include <map>
#include <cctype>
using namespace std;

string remove_extra_whitespaces(const string &input)
{ //--- limpia espacios en blanco innecesarios
  string output;
  unique_copy (input.begin(), input.end(), back_inserter_iterator<string>(output),
               [](char a,char b){ return isspace(a) && isspace(b);});
  return output;
}

string cleanup_text(string& texto)
{ //--- se depura el texto de caracteres como: , . : ; _ / ? \t \r ! ( )
  char invalidos[] = {',', '.', ':', ';', '_', '/', '?', '\t', '\r', '!', '(', ')'};
  for (auto c: invalidos)
    replace(begin(texto), end(texto), c, ' ');
  return remove_extra_whitespaces(texto);
}
```



```

int main()
{
    map<string, int> mapa;    // es un mapa

    ifstream file("biblie.txt");
    //--- validando archivo
    if (file.fail()) {
        cout << "Error: No puede abrirse el archivo\n";
        exit(1);
    }
    string palabra;
    //--- se lee el contenido del archivo con una sola instrucción en texto
    string texto((istreambuf_iterator<char>(file)), istreambuf_iterator<char>());
    //--- se depura el texto de caracteres como: , . : ; _ / ? \t \r ! ( )
    //--- NOTA: C++ tiene una propiedad conocida como return value optimization,
    //          similar al move semantic para evitar copias innecesarias en valores de retorno.
    texto = cleanup_text(texto);
    //--- se coloca palabra a palabra al mapa (no existe una función split en C++ Standard)
    //--- y a la vez se va contabilizando cuántas veces aparece
    stringstream ss(texto);
    while(getline(ss, palabra, ' '))
        mapa[palabra] += 1;
}

```

//--- para poder determinar las palabras que aparecen con mayor frecuencia
//--- se pasaran a otra estructura que es un multimap colocando como “clave”
//--- la cantidad de veces que aparece la palabra en el mapa anterior y
//--- como “valor” la palabra.
//--- NOTA: std::map no puede ser ordenado como el caso de un vector debido a que
std::map es un contenedor asociativo, usualmente se puede transferir
a un vector para luego ser ordenado. Una solución que no requiere ordenar
(ya que al ingresar se ordena automáticamente) y soporta claves repetidas
es el multimap.

```
multimap<int, string> top_values;
for (auto& item: mapa) {
    top_values.insert(make_pair(item.second, item.first));
}
//--- se imprime desde el final hacia el inicio, utilizando los iteradores en
//--- reversa (rbegin, rend).
int i=0;
int top = 20;
for (auto it = rbegin(top_values);
     it != rend(top_values) && i < top;
     ++it, ++i)
    cout << it->first << ", " << it->second << '\n';
file.close();
return 0;
}
```

Resultado

62264, the
 38915, and
 34588, of
 13474, to
 12846, And
 12589, that
 12387, in
 9762, shall
 9668, he
 8942, unto
 8854, I
 8385, his
 8001, a
 7249, for
 6972, they
 6895, be
 6858, is
 6649, him
 6572, not
 6541, LORD

Relaciones de agregación

Ejemplo:

**El programa muestra un tablero y permite alojar, mostrar a Robots.
De cada robot, se conoce su nombre, su ubicación (coordenada x, y) y el color.
El programa trabaja a través de un menú con opciones.**

Ejemplo:

MENU

1. Agregar un nuevo objeto
2. Remover objeto
3. Dibujar Mapa
0. Para Salir

Ingrese Nombre: Bomblebee
Ingrese color (Un caracter): A
Ingrese posición X : 4
Ingrese posición Y : 7

MENU

1. Agregar un nuevo objeto
2. Remover objeto
3. Dibujar Mapa
0. Para Salir

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0
1
2
3
4
5
6
7	A
8
9
10
11
12
13
14
15
16
17
18
19
20

* * * * * [0] Nombre = Bomblebee X = 4 Y = 7 Color = A

Presione C y Enter para continuar...█

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0
1
2	I
3
4
5
6
7	A
8
9
10	.	0
11
12
13
14	R
15
16
17
18
19
20

- * * * * * [0] Nombre = Bomblebee X = 4 Y = 7 Color = A
- * * * * * [1] Nombre = Optimus_Prime X = 1 Y = 10 Color = 0
- * * * * * [2] Nombre = Arcee X = 8 Y = 14 Color = R
- * * * * * [3] Nombre = Ironhide X = 7 Y = 2 Color = I

Presione C y Enter para continuar...

Menu

Tierra

Objeto

Se usa relación de
Agregación

Utilizando vectors

main.cpp

```
#include "Menu.h"

int main() {
    Menu menu;
    menu.ejecutar();
    return 0;
}
```

Tipos.h

```
#ifndef AGREGACION_TIPOS_H
#define AGREGACION_TIPOS_H

#include <string>
using namespace std;
// Definiendo alias
using TipoEntero = int;
using TipoCaracter = char;
using TipoString = string;

#endif //AGREGACION_TIPOS_H
```

```
#ifndef AGREGACION_MENU_H
#define AGREGACION_MENU_H
```

Menu.h

```
#include "Tierra.h"
#include "Tipos.h"
```

```
template <typename T>
T input(string label) {
    T value;
    cout << label;
    cin >> value;
    cin.clear();
    cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    return value;
}
```

```
/*
//--- Esta seria la funcion para leer un string,
//--- esta función si se quiere generalizar para colocar un label
//--- y leer cualquier tipo de dato se tendría que convertir al
template anterior
```

```
string&& input(string label) {
    string value;
    cout << label;
    cin >> value;
    return move(value);
}
*/
```

```
class Menu {
    TipoEntero opcion;
    Tierra tierra;
    void imprimirMenu();
    void seleccionarOpcion();
    void agregarObjeto();
    void removerObjeto();
    void dibujarMapa();
public:
    Menu(): opcion{} {}
    void ejecutar();
};
```

```
#endif //AGREGACION_MENU_H
```

```
#include "Menu.h"
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;
enum class Opciones { Agregar=1, Remover, Mostrar}; // se usa un tipo enumerado para indicar las opciones

void limpiar() {
    cout << "\033[2J\033[0;0H";
}

void esperar() {
    TipoCaracter w;
    do {
        w = input<TipoCaracter>("Presione C y Enter para continuar...");
    }while (toupper(w) != 'C');
}

void Menu::imprimirMenu() {
    limpiar();
    cout << "MENU\n";
    cout << string(4, '-') << "\n\n";
    cout << "1. Agregar un nuevo objeto\n";
    cout << "2. Remover objeto\n";
    cout << "3. Dibujar Mapa\n\n";
    cout << "0. Para Salir\n\n";
}
```

```
void Menu::agregarObjeto()
{
    auto nombre = input<TipoString>("Ingrese Nombre : ");
    auto color  = input<TipoCaracter>("Ingrese color (Un caracter): ");

    auto x = input<TipoEntero>("Ingrese posicion X : ");
    while (x < 0 || x >= tierra.getAncho()) {
        cout << "Posicion X Incorrecta, los limites son: 0, "
              << tierra.getAncho() - 1 << "\n";
        x = input<TipoEntero>("Ingrese posicion X : ");
    }

    auto y = input<TipoEntero>("Ingrese posicion Y : ");
    while (y < 0 || y >= tierra.getAncho()) {
        cout << "Posicion Y Incorrecta, los limites son: 0, "
              << tierra.getAltura() - 1 << "\n";
        y = input<TipoEntero>("Ingrese posicion Y : ");
    }

    tierra.adicionarObjeto(new Objeto(nombre, color, x, y));
}
```

```
void Menu::removerObjeto() {
    auto nombre = input<TipoString>("Ingrese Nombre: ");

    auto obj = tierra.removerObjeto(nombre);  //-- separa el objeto de la tierra
    if (obj == nullptr) {
        cout << "Objeto No existe\n";
    }
    else {
        delete obj;
        cout << "Objeto: " << nombre << " ha sido removido\n";
    }
    esperar();
}

void Menu::dibujarMapa() {
    limpiar();
    tierra.actualizarTierra();
    tierra.dibujarTierra();
    cout << '\n';
    tierra.imprimirObjetos();
    cout << '\n';
    esperar();
}
```

```
void Menu::ejecutar() {
    do {
        imprimirMenu();
        cin >> opcion;
        seleccionarOpcion();
    } while (opcion != 0);
    cout << "Fin del programa...\n";
}

void Menu::seleccionarOpcion() {
    limpiar();
    switch(Opciones(opcion)) {
        case Opciones::Agregar: // Agregar Objeto
            agregarObjeto();
            break;
        case Opciones::Remover: // Remover Objeto
            removerObjeto();
            break;
        case Opciones::Mostrar: // Dibujando Tierra
            dibujarMapa();
            break;
    }
}
```

```
#ifndef AGREGACION_TIERRA_H
#define AGREGACION_TIERRA_H
#include <iostream>
#include <vector>
#include "Tipos.h"
#include "Objeto.h"
using namespace std;
// Valores constantes
const TipoEntero ALTURA = 21;
const TipoEntero ANCHO = 21;
const TipoCaracter COLOR = '.';

class Tierra {
private:
    vector<vector<char>> plano;
    vector<Objeto*> objetos;
public:
    Tierra();
    Tierra(TipoEntero altura, TipoEntero ancho);
    virtual ~Tierra();
    void adicionarObjeto(Objeto* objeto);
    Objeto* removerObjeto(string& nombre);
    void imprimirObjetos();
    TipoEntero getAltura();
    TipoEntero getAncho();
    TipoEntero getCantidadObjectos();
    void dibujarTierra();
    void actualizarTierra();
};

#endif //AGREGACION_TIERRA_H
```



```
#include "Tierra.h"
#include <string>
#include <iomanip>
#include <algorithm>

using namespace std;

Tierra::Tierra() {
    plano.resize(ALTURA);
    for (auto& item: plano)
        item.resize(ANCHO);
}

Tierra::Tierra(TipoEntero altura, TipoEntero ancho) {
    plano.resize(altura);
    for (auto& item: plano)
        item.resize(ancho);
}

Tierra::~Tierra() {}

void Tierra::adicionarObjeto(Objeto* objeto) {
    objetos.emplace_back(objeto);
}
```

```

Objeto* Tierra::removeObjeto(string& nombre) {
    // Si vector esta vacio
    if (objetos.size() == 0)
        return nullptr;
    // Buscando objeto
    auto iter = find_if(begin(objetos), end(objetos),
                        [&nombre](Objeto* obj){ return obj->getNombre() == nombre; });
    if (iter != end(objetos)) {
        // Eliminando la referencia al puntero objeto dentro del vector objetos
        objetos.erase(iter);
        //-- si encuentra al objeto lo separa del vector,
        //-- (no libera de memoria el objeto encontrado), esto se hará en el menú,
        //-- donde fue asignado, debido a que el objeto no es parte sino el objeto es un visitante.
        return *iter;
    }
    // Si vector esta vacio
    return nullptr;
}

void Tierra::imprimirObjetos() {
    int i = 0;
    for (auto& item: objetos) {
        cout << "* * * * * [" << i << "] ";
        cout << " Nombre = " << item->getNombre() << " "
             << item->mostrarPosicion()
             << " Color = " << item->getColor() << '\n';
        i++;
    }
}

```

```

void Tierra::actualizarTierra() {
    for (auto &row: plano)
        for (auto &cell: row)
            cell = COLOR;

    for (auto& item: objetos)
        plano[item->getPosY()][item->getPosX()]
            = item->getColor();
}

```

```

void Tierra::dibujarTierra() {
    cout << '\n';
    cout << setw(3) << ' ';
    for (auto j = 0; j < getAncho(); ++j)
        cout << setw(3) << j;
    cout << '\n';

    for (auto i = 0; i < getAltura(); ++i) {
        cout << setw(3) << i;
        for (auto j = 0; j < getAncho(); ++j) {
            cout << setw(3) << plano[i][j];
        }
        cout << '\n';
    }
}

```

```

TipoEntero Tierra::getAltura() {
    return plano.size();
}

```

```

TipoEntero Tierra::getAncho(){
    return plano[0].size();
}

```

Objeto.h

```
#ifndef AGREGACION_OBJETO_H
#define AGREGACION_OBJETO_H
#include <iostream>
#include "Tipos.h"
using namespace std;

class Objeto {
private:
    string      nombre;
    TipoCaracter color;
    TipoEntero  posX;
    TipoEntero  posY;
public:
    Objeto();
    Objeto(const TipoString& nombre, TipoCaracter color,
           TipoEntero posX, TipoEntero posY);
    virtual ~Objeto();
    void setNombre(const TipoString& nombre);
    string getNombre();
    TipoEntero getPosX();
    TipoEntero getPosY();
    char getColor();
    void moverse(TipoEntero x, TipoEntero y);
    string mostrarPosicion();
};

#endif //AGREGACION_OBJETO_H
```

```
#include "Objeto.h"
```

```
Objeto::Objeto(): color{}, posX{}, posY{} {}
```

```
Objeto::Objeto(const TipoString& nombre, TipoCaracter color,
               TipoEntero posX, TipoEntero posY):
    nombre{nombre}, color{color},
    posX{posX}, posY{posY} {}
```

```
Objeto::~~Objeto() {}
```

```
void Objeto::setNombre(const TipoString& nombre) { this->nombre = nombre; }
void Objeto::moveTo(TipoEntero x, TipoEntero y) {} //-- por implementar
```

```
TipoString  Objeto::getNombre() { return nombre; }
TipoEntero  Objeto::getPosX()   { return posX; }
TipoEntero  Objeto::getPosY()   { return posY; }
TipoCaracter Objeto::getColor() { return color; }
```

```
TipoString Objeto::mostrarPosicion() {
    return "X = " + to_string(posX) + " Y = " + to_string(posY);
}
```

El programa lo pueden encontrar en el Github

<https://github.com/Hildiu/agregacion.git>

Unidad 7:

Programación Orientada a Objetos - Parte 3

Relaciones de agregación

<http://bit.ly/2HRBWgq>

Profesores:

Ernesto Cuadros- Vargas, PhD. ecuadros@utec.edu.pe

María Hilda Bermejo, M. Sc. mbermejo@utec.edu.pe