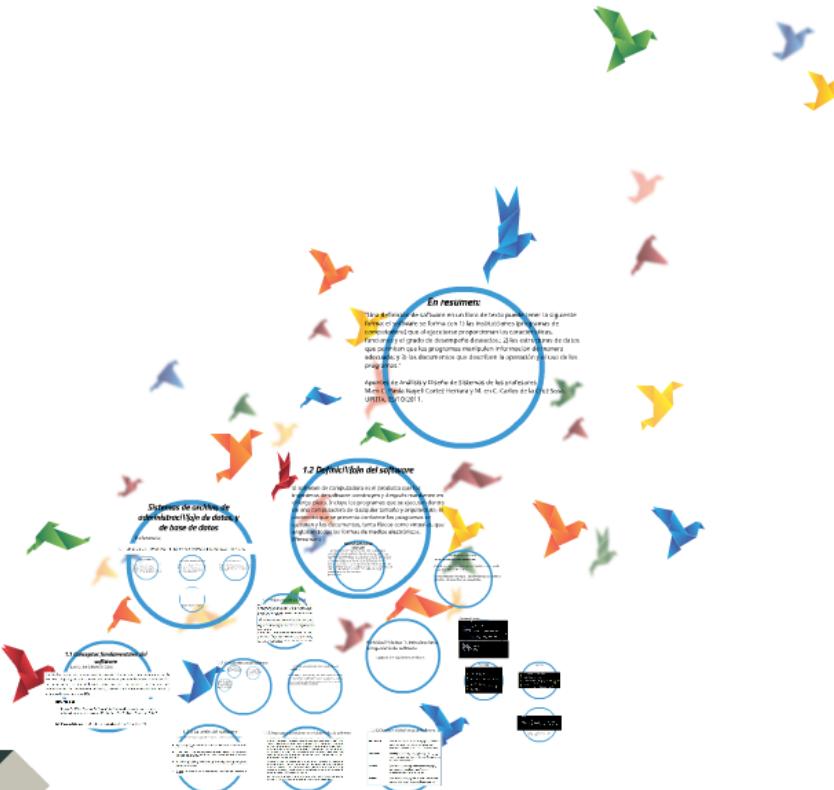


## Práctica UNO: Introducción a la Ingeniería del software



# ***Práctica No. 1 Introducción a la Ingeniería del Software***

Objetivo: Definir los requisitos y el ciclo de vida en el diseño de software

# **1.1 Conceptos fundamentales del software**

(DATO, INFORMACIÓN)

Un objetivo de los sistemas es que sus usuarios tengan que tratar solamente con la estructura lógica y las operaciones realizadas en el procesamiento de su información. Por esta razón es útil distinguir entre sistemas de archivo (mecanismos de administración de información básicos), sistemas de administración de datos y sistemas de base de datos ([2]).

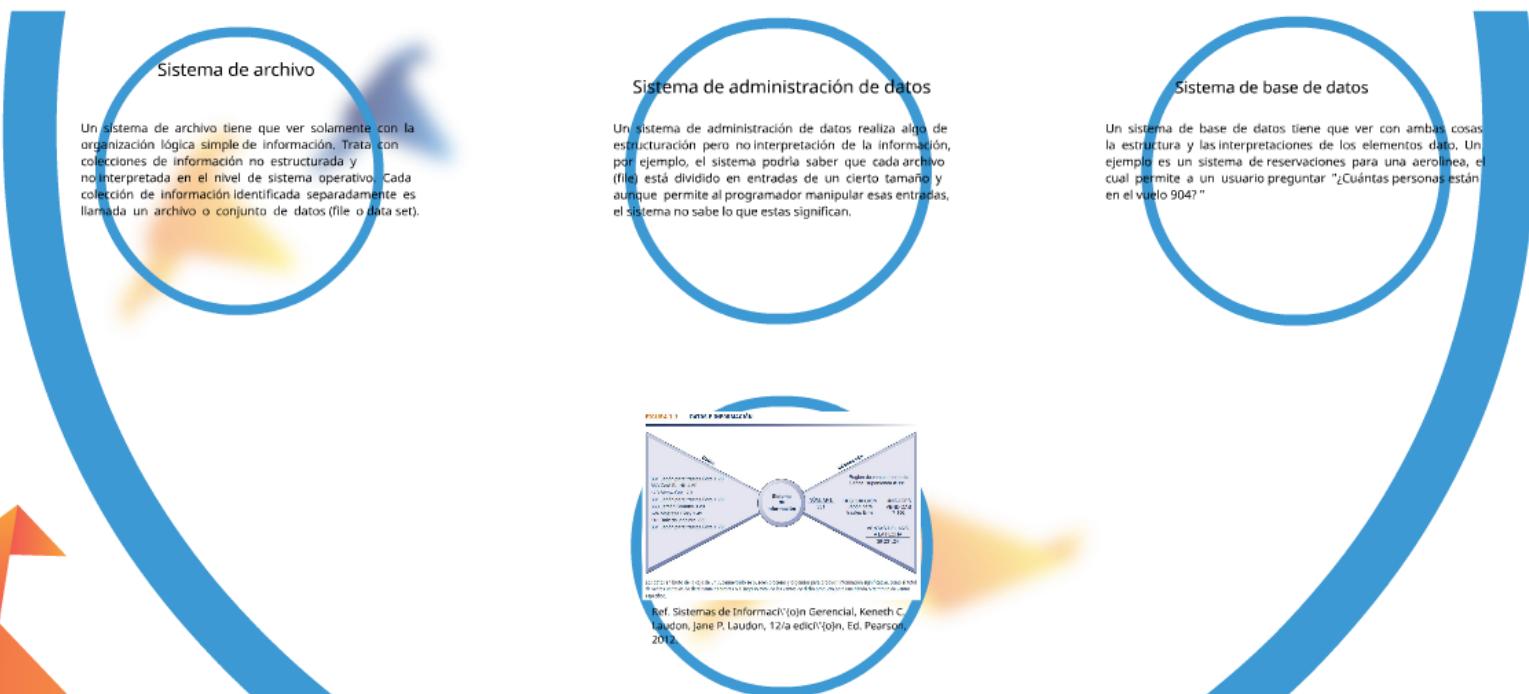
## **Referencias**

- [1] Booch, G., *Object Oriented Design with Applications*. Benjamin/Cummings series in Ada and software engineering. Menlo Park, CA: Benjamin/Cummings Pub. Co., 1991.
- [2] Madnick S. E., Donovan J. J., *Operating Systems*, Ed. McGraw-Hill, 1974.

# *Sistemas de archivo, de administraci\'on de datos, y de base de datos*

## Referencia:

- [2] Madnick S. E., Donovan J. J., *Operating Systems*, Ed. McGraw-Hill, 1974.



## 1.2.2 Import

Actualmente casi todos los países tienen infraestructuras nacionales y utilizan parte de los productos eléctricos en la fabricación industrial y distribución del sector financiero. Por lo tanto, producir soluciones de la economía nacional e internacional.

La ingeniería del software es una disciplina costeable de sistemas de software que no se basa en la fabricación de materiales. O gobernado por la ingeniería física, esta simplifica la ingeniería física del potencial del software. Esto significa que el software puede llegar a ser muy difícil de entender.

# Sistema de archivo

Un sistema de archivo tiene que ver solamente con la organización lógica simple de información. Trata con colecciones de información no estructurada y no interpretada en el nivel de sistema operativo. Cada colección de información identificada separadamente es llamada un archivo o conjunto de datos (file o data set).

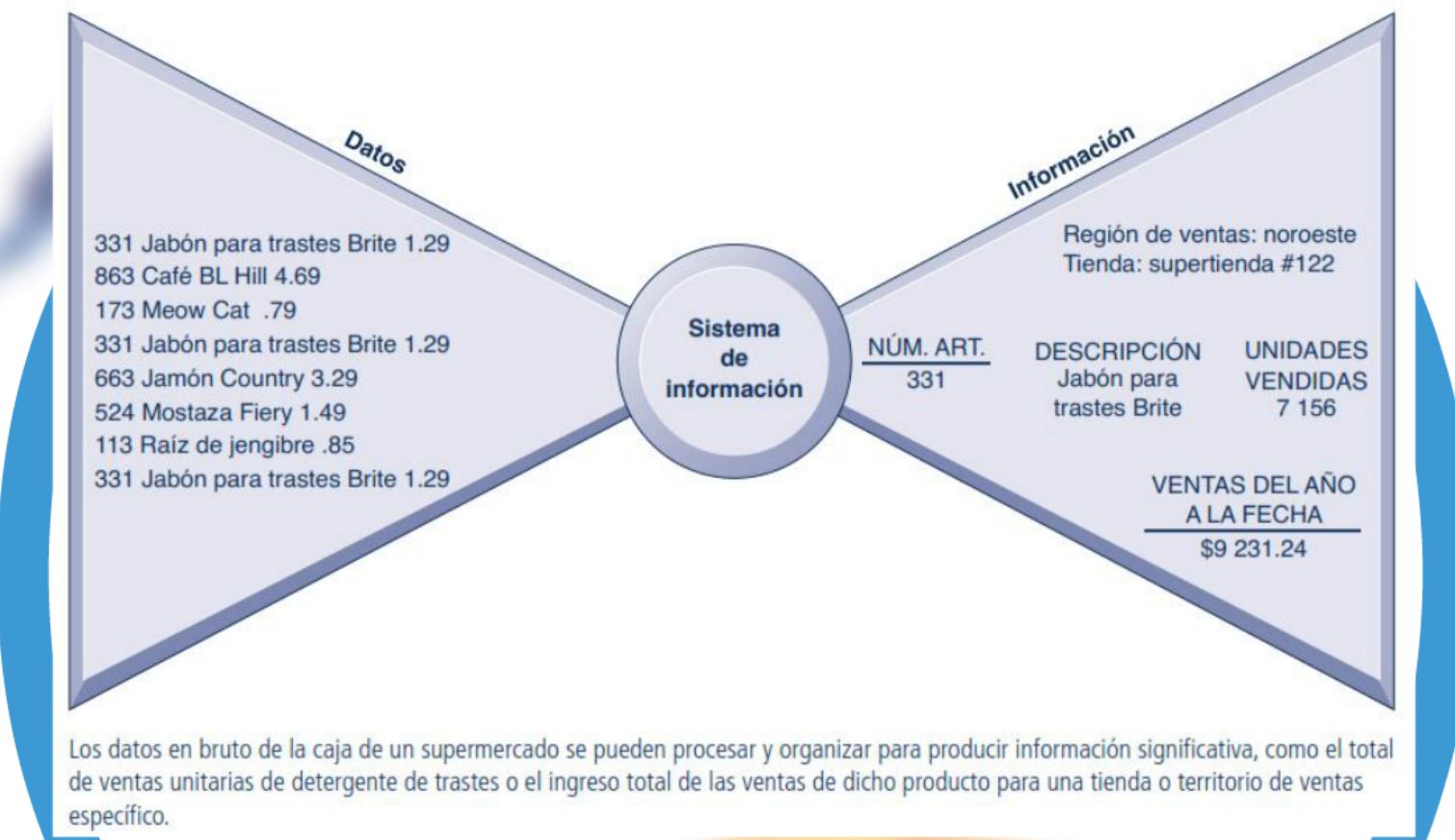
# Sistema de administración de datos

Un sistema de administración de datos realiza algo de estructuración pero no interpretación de la información, por ejemplo, el sistema podría saber que cada archivo (file) está dividido en entradas de un cierto tamaño y aunque permite al programador manipular esas entradas, el sistema no sabe lo que estas significan.

## Sistema de base de datos

Un sistema de base de datos tiene que ver con ambas cosas la estructura y las interpretaciones de los elementos dato. Un ejemplo es un sistema de reservaciones para una aerolínea, el cual permite a un usuario preguntar "¿Cuántas personas están en el vuelo 904? "

**FIGURA 1-3 DATOS E INFORMACIÓN**



Ref. Sistemas de Información Gerencial, Kenneth C. Laudon, Jane P. Laudon, 12/a edición, Ed. Pearson, 2012.

## 1.2 Definición del software

El software de computadora es el producto que los ingenieros de software construyen y después mantienen en el largo plazo. Incluye los programas que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, el contenido que se presenta conforme los programas se ejecutan y los documentos, tanto físicos como virtuales, que engloban todas las formas de medios electrónicos.

(Pressman)

### *Sobre el software se sabe que*

El software no son sólo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general, un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes.

(Sommerville)

Produ...  
Los productos softw...  
Genéricos.- desarrol...  
rango de clientes di...  
Particulares (por e...  
individual de acuerdo

# ***Sobre el software se sabe que***

El software no son sólo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general, un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes.

(Sommerville)

# Productos software

Los productos software pueden ser:

Genéricos.- desarrollados para ser vendidos a un amplio rango de clientes diferentes.

Particulares (por encargo) .- desarrollado para un cliente individual de acuerdo a sus necesidades.

## ***En resumen:***

"Una definición de software en un libro de texto puede tener la siguiente forma: el software se forma con 1) las instrucciones (programas de computadora) que al ejecutarse proporcionan las características, funciones y el grado de desempeño deseados.; 2) las estructuras de datos que permiten que los programas manipulen información de manera adecuada; y 3) los documentos que describen la operación y el uso de los programas."

Apuntes de Análisis y Diseño de Sistemas de los profesores  
M.en C. Paola Nayeli Cortez Herrera y M. en C. Carlos de la Cruz Sosa,  
UPIITA, 05/10/2011.

## 1.2.1 Características del software

### 1. El software se desarrolla o se construye; no se manufatura en el sentido clásico.

A pesar de que existen similitudes entre el desarrollo del software y la manufactura del hardware, las dos actividades son diferentes en lo fundamental. En ambos, la alta calidad es deseada, pero el resultado final es muy diferente. La manufactura puede incluir problemas de calidad inexistentes (o que son fáciles de corregir) en el software. Ambas actividades requieren la construcción de un "producto", pero los enfoques son diferentes. Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no se pueden manejar como si fueran proyectos de manufactura.

### 2. El software no se "desgasta".

El software es immune a los males ambientales que desgastan el hardware. Los defectos sin descubrir causan tasas de falla altas en las primeras etapas de vida de un programa. Sin embargo, los errores se corrigen (en el mejor de los casos sin agregar otros errores). Esto concluye que el software no se desgasta, pero sí se deteriora. Durante su vida, el software experimenta cambios. Conforme estos ocurren se presenta la posibilidad de introducir errores, lo que puede ocasionar fallas. Por lo cual el software se deteriora debido a los cambios.

3. A pesar de que la industria tiene una tendencia hacia la construcción por componentes, la mayoría del software aún se construye a la medida. Considerese la forma en que se diseña y v construye un hardware de control para un producto de cómputo. El ingeniero de diseño dibuja un esquema simple del sistema de circuitos digital, realiza algunos análisis fundamentales para asegurarse de que el diseño realizará las funciones apropiadas y después busca en los catálogos de componentes digitales cada circuito integrado de acuerdo con un número de parte, una función definida y validada, una interfaz bien definida y un conjunto estandarizado de directrices de integración. Una vez seleccionado cada componente, puede solicitarse para después ensamblarlo.

"Cuando una disciplina de Ingeniería evoluciona se crea una colección de diseños estándar de componentes. Los tornillos y los circuitos integrados son solo dos ejemplos de los miles de componentes estándar que utilizan los ingenieros mecánicos y eléctricos al diseñar sistemas nuevos. Los componentes reutilizables se han creado para que el ingeniero se pueda concentrar en los elementos que en realidad son innovadores en el diseño, es decir, en las partes que representan algo nuevo. En el mundo del hardware, la reutilización de componentes es una parte natural del proceso de Ingeniería. En el ámbito del software, dicha actividad apenas se ha comenzado a extender."

Apuntes de Análisis y Diseño de Sistemas de los profesores  
M. en C. Paola Nayeli Cortez Herrera y M. en C. Carlos de la Cruz Sosa,  
UPIITA, 09/10/2011.

1. El software se desarrolla o se construye; no se manufactura en el sentido clásico.

A pesar de que existen similitudes entre el desarrollo del software y la manufactura del hardware, las dos actividades son diferentes en lo fundamental. En ambas, la alta calidad se alcanza por medio del buen diseño, pero la fase de manufactura del hardware puede incluir problemas de calidad inexistentes (o que son fáciles de corregir) en el software. Ambas actividades requieren la construcción de un “producto”, pero los enfoques son diferentes. Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no se pueden manejar como si fueran proyectos de manufactura.

2. El software no se “desgasta”

El software es immune a los males ambientales que desgastan el hardware. Los defectos sin descubrir causan tasas de falla altas en las primeras etapas de vida de un programa. Sin embargo, los errores se corrigen (en el mejor de los casos sin agregar otros errores). Esto concluye que el software no se desgasta, pero sí se deteriora. Durante su vida, el software experimenta cambios. Conforme estos ocurren se presenta la posibilidad de introducir errores, lo que puede ocasionar fallas. Por lo cual el software se deteriora debido a los cambios.

3. A pesar de que la industria tiene una tendencia hacia la construcción por componentes, la mayoría del software aún se construye a la medida. Considérese la forma en que se diseña y v construye un hardware de control para un producto de cómputo. El ingeniero de diseño dibuja un esquema simple del sistema de circuitos digital, realiza algunos análisis fundamentales para asegurarse de que el diseño realizará las funciones apropiadas y después busca en los catálogos de componentes digitales cada circuito integrado de acuerdo con un numero de parte, una función definida y validada, una interfaz bien definida y un conjunto estandarizado de directrices de integración. Una vez seleccionado cada componente, puede solicitarse para después ensamblarlo.

"Cuando una disciplina de ingeniería evoluciona se crea una colección de diseños estándar de componentes. Los tornillos y los circuitos integrados son solo dos ejemplos de los miles de componentes estándar que utilizan los ingenieros mecánicos y eléctricos al diseñar sistemas nuevos. Los componentes reutilizables se han creado para que el ingeniero se pueda concentrar en los elementos que en realidad son innovadores en el diseño, es decir, en las partes que representan algo nuevo. En el mundo del hardware, la reutilización de componentes es una parte natural del proceso de ingeniería. En el ámbito del software, dicha actividad apenas se ha comenzado a extender."

Apuntes de Análisis y Diseño de Sistemas de los profesores  
M.en C. Paola Nayeli Cortez Herrara y M. en C. Carlos de la Cruz Sosa,  
UPIITA, 05/10/2011.

## 1.2.2 Importancia del software

Actualmente casi todos los países dependen de complejos sistemas informáticos. Infraestructuras nacionales y utilidades dependen de sistemas informáticos, y la mayor parte de los productos eléctricos incluyen una computadora y software de control. La fabricación industrial y distribución está completamente informatizada, como el sistema financiero. Por lo tanto, producir software costeable es esencial para el funcionamiento de la economía nacional e internacional.

La ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Éste es abstracto e intangible. No está restringido por materiales. O gobernado por leyes físicas o por procesos de manufactura. De alguna forma, esto simplifica la ingeniería del software ya que no existen limitaciones físicas del potencial del software. Sin embargo, esta falta de restricciones naturales significa que el software puede llegar a ser extremadamente complejo y, por lo tanto, muy difícil de entender.

Se puede afirmar que hemos hecho enormes progresos desde 1968 y que el desarrollo de esta ingeniería ha mejorado considerablemente nuestro software. Se comprenden mucho mejor las actividades involucradas en el desarrollo de software. Se han desarrollado métodos efectivos de especificación, diseño e implementación del software. Las nuevas notaciones y herramientas reducen el esfuerzo requerido para producir sistemas grandes y complejos.

### 1.2.3 Complejidad del diseño y desarrollo del software

La complejidad inherente al diseño y desarrollo del software se deriva de cuatro elementos: la complejidad del dominio del problema, la necesidad de gestionar el proceso de desarrollo, la flexibilidad que se puede alcanzar a través del software y los problemas que plantea la caracterización del comportamiento de sistemas discretos.

## 1.2.4 La crisis del software

La crisis del software se puede resumir en estos cuatro puntos

- Hace referencia a un conjunto de problemas encontrados en el desarrollo del software (años 70's).
- A partir de los 70, se realiza grandes avances en la tecnología del hardware que hace que su coste disminuya y que puedan aportarse soluciones a problemas más complejos.
- La evolución del software por el contrario ha sido más lenta y por tanto, puede decirse que el cuello de la botella de la informática está en la optimización de la producción del software.
- El coste informático de una empresa se centra en el coste producido por el software.

## 1.2.5 Aspectos a considerar en el desarrollo de software

La práctica de cualquier disciplina de ingeniería – sea civil, mecánica, química, eléctrica o informática – involucra elementos tanto de ciencia como de arte. Como Petroski elocuentemente afirma, <<la concepción de un diseño para una nueva estructura puede involucrar un salto de la imaginación y una síntesis de experiencia y conocimiento tan grandes como el que requiere cualquier artista para plasmar su obra en una tela o en un papel. Y una vez que el ingeniero ha articulado ese diseño como artista, debe analizarlo como científico aplicando el método científico con tanto rigor como cualquier científico lo haría>>.

Cualquier proyecto de software se inicia por alguna necesidad de negocios: la necesidad de corregir un defecto en una aplicación existente; el imperativo de adaptar un sistema heredado a un ambiente de negocios cambiante; el requerimiento de extender las funciones y características de una aplicación existente; o la necesidad de crear un producto, servicio o sistema nuevos.

Con frecuencia, en el inicio de un proyecto de ingeniería del software la necesidad de negocios se expresa de manera informal durante una simple conversación.

## 1.2.6 Claves en el diseño del software

### Mantenibilidad

El software debe escribirse de tal forma que pueda evolucionar para cumplir las necesidades de cambio de los clientes. Éste es un atributo critico debido a que el cambio en el software es una consecuencia inevitable de un cambio en el entorno de negocios.

### Confiabilidad

La confiabilidad del software tiene un gran número de características, incluyendo la fiabilidad, protección y seguridad. El software confiable no debe causar daños fisicos o economicos en el caso de una falla en el sistema.

### Eficiencia

El software no debe hacer que se malgasten los recursos del sistema, como la memoria y los ciclos de procesamiento. Por lo tanto, la eficiencia incluye tiempos de respuesta y de procesamiento, utilización de la memoria, etc.

### Usabilidad

El software debe ser fácil de utilizar, sin esfuerzo adicional, por el usuario para quien esta diseñado. Esto significa que debe tener una interfaz de usuario apropiada y una documentación adecuada.

# Actividad Práctica 1: Introducción a la ingeniería de software

Capture los siguientes archivos:

# Makefile

```
1 EXE:=TestFecha.exe
2 CC=g++
3 LIMPIEZA:=$(EXE)
4 $(EXE):DNOMBREDia.cpp main.cpp
5           $(CC) -I./ $^ -o $@
6 clean:
7         rm -v $(LIMPIEZA)
```

```
1 /**DFecha.h
2 */
3 struct Fecha{
4     unsigned int d,m,a; /*dia,mes,año*/
5 }/*end struct Fecha*/
6 string get_string_fecha(struct Fecha F);
7
8
```

Con esta estructura modelaremos una fecha con un entero para el día, un entero para el mes y un entero para el año.

```
1 /**DNOMBRE.cpp
2 */
3 #include <iostream>
4 using namespace std;
5 #include <stdio.h> /*sprintf()*/
6 #include <DFecha.h>
7
8 /*Numero magico*/
9 unsigned int NM[]={0,6,3,2}; /*{*,enero,febrero,marzo}*/
10 string DIA[]={
11     "Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"
12 };
13 string MES[]={
14     "string MES[]","enero","febrero","marzo","abril","mayo","junio",
15     "julio","agosto","septiembre","octubre","noviembre","diciembre"
16 };
17 string get_string_fecha(struct Fecha F)
18 {
19     char tmp[5];
20     string fecha="";
21     fecha=fecha+DIA[(F.d+7-NM[F.m])%7];
22     fecha=fecha+" ";
23     sprintf(tmp,"%d",F.d);
24     fecha=fecha+string(tmp);
25     fecha=fecha+" de ";
26     fecha=fecha+MES[F.m];
27     fecha=fecha+" de 2020";
28     return fecha;
}
```



# main.cpp

```
1 /**main.cpp
2 */
3 #include <iostream>
4 using namespace std;
5 #include <DFecha.h>
6
7 int main(int argc,char *argv[])
8 {
9     struct Ffecha f;
10    f.d=7;f.m=2;f.a=2020;
11    cout<<get_string_fecha(f);
12    return 0;
13 }/*end main()*/
14
```



## Práctica UNO: Introducción a la Ingeniería del software

