

SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO SUPERIOR DE ZACAPOAXTLA



DIVISIÓN DE INGENIERÍA MECATRÓNICA ACADEMIA DE INGENIERÍA MECATRÓNICA

ANTOLOGÍA

Programación Básica



Lic. Edgar Hernández García

20 de Junio de 2016



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

PROPÓSITO DEL CURSO

El presente texto es una compilación de notas de clase impartidas entre 2011 y 2016, en la asignatura de Programación Básica, de la carrera de Ingeniería Mecatrónica en el Instituto Tecnológico Superior de Zacapoaxtla.

En base a las unidades propuestas en el temario ésta antología posee siete unidades o capítulos:

- En la unidad 1 se ofrece una muy breve introducción a los conceptos básicos de la computación. Entre otras cosas se distingue el concepto del bit y del Byte como el elemento base para la construcción de un sistema de códigos y de la arquitectura de computadoras.
- En la unidad 2 se estudian de manera breve los algoritmos y los diagramas de flujo como un principio lógico de la programación.
- La unidad 3 realiza una introducción al lenguaje C++, distinguiendo la sintaxis del mismo, los tipos de datos atómicos y la escritura de programas sencillos.
- Las estructuras de decisión y control de flujo son revisadas en la unidad 4.
- El manejo de arreglos y una breve introducción a la entrada y salida de datos por archivos se estudia en la unidad 5.
- La unidad 6 introduce los fundamentos de la programación modular.
- Más ambiciosa es la unidad 7 en la que se estudia la biblioteca OpenGL para la creación de gráficos en el lenguaje C.
- En la unidad 8 se presentan dos ejemplos de prácticas demostrativas del manejo de puertos de una computadora.

El curso se enfoca única y exclusivamente en el paradigma de la programación modular, en cumplimiento de los requerimientos solicitados en el temario de la asignatura mencionada. Se usa como lenguaje al estándar ISO C++.

En virtud de lo anterior es deseable que el estudiante domine el Álgebra y la Lógica, bases que se pudieron adquirir en los albores del curso de Cálculo Diferencial y reforzados en la asignatura de Álgebra Lineal.

A su vez ésta asignatura da sustento a las asignaturas de Métodos Numéricos, Ecuaciones Diferenciales, Vibraciones Mecánicas, Electrónica Digital, Microcontroladores, Controladores Lógico Programables y Programación Avanzada.

No pretendemos en ningún sentido sustituir libro de texto alguno, sino ofrecer humildemente una guía para la impartición de la materia.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Contenido

1.	Introducción a la Computación	6
1.1.	Breve Reseña	6
1.2.	Definiciones	6
1.2.1.	Hardware y Software	7
1.3.	Generaciones de Computadoras	11
1.4.	Arquitectura básica de una computadora	15
1.4.1.	El Bit y el Byte	15
1.4.2.	Diversas medidas de Bytes	17
1.4.3.	Tipos de Memoria	18
1.4.4.	Arquitectura Von Neumann y Arquitectura Harvard	19
2.	Diseño de Algoritmos	21
2.1.	Conceptos Básicos	21
2.2.	Datos	22
2.3.	Operaciones	23
2.4.	Expresiones	23
2.5.	Diagramas de Flujo	24
2.5.1.	Simbología Básica	24
2.5.2.	Bloques de decisión en un diagrama de flujo	26
2.5.3.	Estructuras iterativas	27
2.6.	Pseudocódigo	30
3.	Fundamentos del Lenguaje	31
3.1.	Breve historia del lenguaje C++	31
3.2.	Características básicas del lenguaje C++	31
3.3.	Secciones de un programa básico	35
3.4.	Variables	36
3.4.1.	Tipos de Datos	37
3.4.2.	Secuencias de escape en cadenas de caracteres	39
3.5.	Constantes	39



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

3.6.	Operadores Básicos.....	40
3.6.1.	Operaciones con caracteres	40
3.6.2.	Modificadores de operandos	41
3.6.3.	Operadores de Desplazamiento.....	41
3.6.4.	Operadores relacionales y operadores lógicos	42
3.6.5.	Precedencia de operaciones	43
3.7.	Entrada y salida de datos	43
4.	Estructuras de decisión y control.....	47
4.1.	Sentencia if-else	47
4.2.	Ciclo while	48
4.3.	Ciclo do-while.....	49
4.4.	Sentencia switch.....	50
4.5.	Ciclo for	51
4.6.	Ejercicios Complementarios.....	52
5.	Arreglos y Archivos.....	54
5.1.	Arreglos	54
5.1.1.	Arreglos Vectoriales	54
5.1.2.	Arreglos Matriciales	56
5.1.3.	Arreglos de cadenas Char.....	57
5.2.	Archivos.....	59
6.	Módulos	64
6.1.	Concepto de módulo.....	64
6.2.	Funciones	64
6.3.	Procedimientos	66
6.4.	Macros.....	67
6.5.	Recursividad	68
7.	Graficación	70
7.1.	Fundamentos	70
7.2.	Elementos de un ambiente gráfico	71



7.2.1.	Adaptador gráfico.....	72
7.2.2.	Software: API's y GUI's.....	76
7.3.	Librerías de OpenGL para graficación 2D	78
7.4.	El IDE y la API OpenGL.....	79
7.5.	Generando una ventana.....	83
7.6.	Preparación del área de graficado	86
7.7.	Primitivas 2D	89
7.7.1.	Trazos por puntos.....	91
7.7.2.	Líneas.....	92
7.7.3.	Polígonos	94
7.8.	Modelo de sombreado	95
7.9.	Sentencias útiles.....	96
7.10.	Ejercicios sugeridos	97
8.	Puertos	99
8.1.	Comentarios previos	99
8.2.	Arquitectura de puertos de una computadora	99
8.3.	Práctica propuesta 1: Control de LED's por puerto paralelo	103
8.3.1.	Pasos Previos	105
8.3.2.	Construcción del entorno gráfico.....	106
8.3.3.	Codificación	111
8.3.4.	Prueba del programa.....	117
8.4.	Práctica Propuesta 2: Encendido de LED's en un Raspberry Pi.....	118
8.4.1.	Estructura básica de la Raspberry Pi	118
8.4.2.	Pasos Previos	120
8.4.3.	Terminales GPIO	121
8.4.4.	Control de LED's en Python	122
	Bibliografía	125



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

1. Introducción a la Computación

En ésta unidad el estudiante:

- Adquiere los conceptos fundamentales de la computación.

6

1.1. Breve Reseña

¿Cuál fue la necesidad que provocó la invención de la computadora? ¿Por qué dicho aparato tiene ese nombre?

Los primeros seres humanos (en forma más precisa: el *homo sapiens*) no conocían el concepto de propiedad, ya que todo lo que recolectaban o cazaban era comunitario. Pero cuando dichos humanos descubrieron que podían producir sus propios alimentos mediante la crianza y el cultivo, y además que podían especializarse en algunos de esos productos, nació el trueque y por consiguiente la propiedad... Y con ella la necesidad de contar.

Los primeros humanos no tenían mucho que contar porque no producían en grandes cantidades. Entonces como herramienta de comparación y conteo utilizaron los dedos de la mano (¿será por eso que nuestro sistema de numeración es **decimal**?), aunque hay excepciones: los mayas manejaban un sistema vigesimal y los babilonios el sexagesimal).

Luego, a medida que fueron evolucionando los sistemas de producción, igualmente se desarrollaron otros métodos para contar, y en cada caso crecía la necesidad de contar más y más rápido. Como ejemplos de herramientas de conteo y cálculo podemos citar al ábaco, el sistema de nudos, la calculadora mecánica y la calculadora electrónica.

Y finalmente hizo su aparición la computadora, cuyo objetivo fundamental era el de realizar complicados cálculos sistematizados y automatizados. Precisamente las dos últimas palabras son las que le han dado su lugar en el desarrollo de la civilización actual, al ser capaz de realizar procesos **programados**.

1.2. Definiciones

¿Y qué significa la palabra **computación**? De entre todas las definiciones que ofrecen los diccionarios podríamos referirnos a dos respuestas:

- Operación consistente en representar información mediante un código, por ejemplo, representar cada carácter alfanumérico con una cadena de ceros y unos (el código 00100000 podría representar a la famosísima arroba (@)).
- Estudio científico que se desarrolla sobre sistemas automatizados de manejo de información, lo cual se lleva a través de herramientas pensadas para tal propósito.

¿Y qué definición tiene la **Computadora**? Podría decirse que es una máquina capaz de efectuar secuencias de operaciones mediante el establecimiento de un programa, así, realiza un procesamiento sobre un conjunto de datos de entrada obteniéndose a cambio un conjunto de datos de salida.

¿Qué es la **Informática**? Es la disciplina que estudia cómo realizar el tratamiento (procesamiento) automático de la información utilizando herramientas electrónicas y/o computacionales.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuac-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

**Datos de Entrada****Procesamiento****Datos de Salida**

Tareas Básicas en una Computadora

7

1.2.1. Hardware y Software

Una computadora es un aparato que evidentemente funciona a través de dispositivos electrónicos y se alimenta de corriente eléctrica, sin embargo, para su descripción básica se le suele dividir en dos conceptos:

Hardware: Es el conjunto de partes electrónicas que forman a la computadora, es decir, se refiere al aparato mismo.



Software: Es el conjunto de instrucciones que se encuentran programadas en una computadora para realizar tareas.



Si comparásemos a un humano con una computadora diríamos que el hardware se correspondería con su cuerpo, mientras que sus pensamientos, ideas, o su alma vendrían siendo el software.

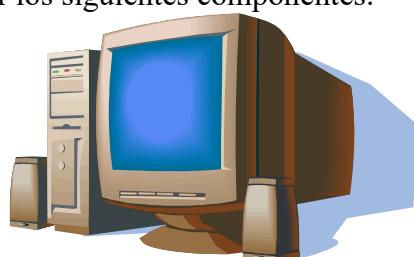
Por hardware entonces podemos entender que es todo aquel elemento de la computadora que se puede agarrar o tocar. De hecho, la palabra misma se refiere a un material que es duro (hard en inglés es duro) y que también es una herramienta (ware vendría significando herramienta o instrumento). Y bueno, el aparato computador es ciertamente duro.

Algo equivalente se aplica en el caso de la palabra software, soft se refiere a algo que es liviano, ligero o blando. Anteriormente los programas se cargaban en una computadora a través de dispositivos (tarjetas y disquetes) que tenían la característica de ser muy blandos, pues se doblaban con mucha facilidad (situación que continuamente traía dolores de cabeza).

**1.2.1.1. Hardware**

El Hardware generalmente está compuesto por los siguientes componentes:

- ✓ Gabinete.
- ✓ Monitor.
- ✓ Teclado.
- ✓ Ratón o mouse.
- ✓ Altavoces
- ✓ Micrófono
- ✓ Cámara Web.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

En el gabinete se encuentran las partes que quizá son las más importantes de la computadora.

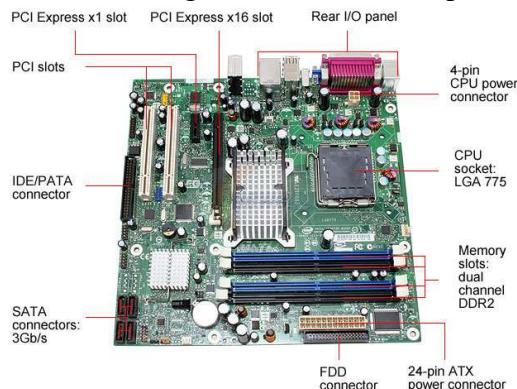
Externamente es posible observar:

- Lectores y/o quemadores de CD/DVD.
- Lectores de tarjetas de memoria.
- Puertos frontales de USB.



8

Internamente el gabinete contiene la parte modular de la computadora:



- Tarjeta madre (o Motherboard).
- Procesador (o Microprocesador).
- Tarjetas y ranuras de expansión.
- Memorias.

Las tarjetas de expansión son placas con chips electrónicos especiales que ayudan al procesador en tareas específicas como la de emular el video o el sonido. Por lo mismo existen diversas clases de tarjetas:



- ❖ De Video y/o de TV.
- ❖ De Audio.
- ❖ De red (Ethernet).
- ❖ Expansoras de puertos.

En general todo hardware externo a la tarjeta madre es llamado **periférico** porque está conectado en torno (alrededor) del CPU (en la periferia). Existen tres clases de periféricos:

Periféricos de Entrada: Son aquellos que permiten introducir información en la computadora.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Ejemplos de Periféricos de Entrada.

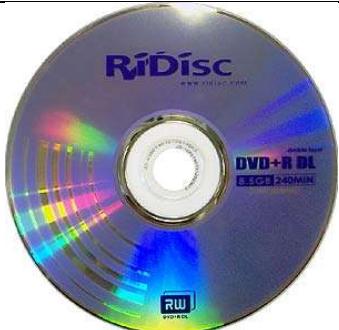
9

Periféricos de Salida: Son los que permiten obtener información de una computadora.



Ejemplos de Periféricos de Salida.

Periféricos Mixtos o Híbridos: Los que permiten tanto introducir como obtener información.



Ejemplos de Periféricos Mixtos o Híbridos

1.2.1.2. El Software

Ya se dijo que básicamente son los programas. Son los que se encargan de preparar al equipo electrónico para poder desarrollar tareas. Pero no todos los programas son iguales, también los tenemos clasificados en tres grupos:

Sistemas Operativos: Son los programas que permiten el intercambio de códigos de símbolos humanos con códigos de máquina. En otras palabras, son los programas que sirven de intérpretes entre el lenguaje humano y el lenguaje de máquina. Gracias a los



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



SO's, un usuario con conocimientos muy básicos de computación puede usar una computadora.



10

Lenguajes de Programación: Son los programas que mediante palabras clave tomadas del lenguaje humano codifican instrucciones en una computadora para la realización de procesos de información automatizados. Existen dos clases fundamentales de lenguajes: de **bajo nivel**, el cual usa código de máquina para programar directamente el microprocesador de una computadora; los de **alto nivel**, que se dividen a su vez en: **lenguajes interpretados** en los que usan sólo palabras clave tomadas del lenguaje humano y que a su vez requieren de un subprograma llamado **máquina intérprete** que se encarga de ir traduciendo las instrucciones al tiempo que éstas se ejecutan en el programa; y los **lenguajes compilados** que igualmente usan palabras del lenguaje natural pero que no requieren de un intérprete, pues un subprograma llamado **compilador**, es capaz de generar un archivo ejecutable directamente por la computadora. Algunos lenguajes de programación como el C++, tienen cualidades de ambos tipos de programación, por lo que se les suele clasificar como de **nivel medio**.



Aplicaciones de Usuario: Son los programas cuyos fines son de mera utilidad para resolver tareas específicas, como pueden ser la captura y edición de textos, dibujar, charlar, etc.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



1.3. Generaciones de Computadoras

Las computadoras han ido evolucionando a la par de las necesidades humanas de información y comunicación. Así, es posible distinguir cambios fundamentales en su tecnología, tales cambios son conocidos como **Generaciones de Computadoras**.

Como se dijo en la introducción, la necesidad fundamental era contar de manera eficiente y rápida, en este sentido podemos citar los antecesores de las computadoras:

- I. El **Ábaco** fue uno de los primeros instrumentos de cálculo, fue usado por las civilizaciones griegas y romanas (aunque aún se usa mucho en países orientales). Aunque se le puede dar un uso eficiente tiene la desventaja de que no se puede programar.
- II. La **Pascalina** es una de las primeras calculadoras mecánicas, que funcionaba a base de ruedas y engranes. Fue inventada por Blaise Pascal en 1645, su desarrollo dilató tres años. Su comportamiento era analógico, tantos giros de un engrane de un tamaño, respecto de otro engrane de tamaño distinto representaba el resultado de los cálculos.
- III. La **Máquina Analítica**, era el diseño de una computadora moderna de uso general realizado por el profesor británico de matemáticas Charles Babbage. Fue inicialmente descrita en 1837, aunque Babbage continuó refinando el diseño hasta su muerte en 1871. Su concepto estaba dirigido a la resolución de sumas repetitivas. Se la puede llamar también “computadora analógica o mecánica”.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuac-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

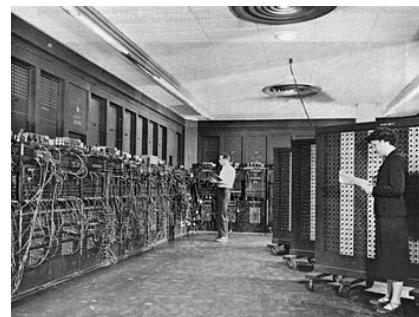


IV. La **Harvard Mark I** o **Mark I** fue la primera computadora **electromecánica** construida en la Universidad de Harvard por Howard H. Aiken en 1944, con la subvención de IBM. Tenía 760.000 ruedas y 800 kilómetros de cable y se basaba en la máquina analítica de Charles Babbage. Era una combinación de partes eléctricas y mecanismos, razón por la cual era un aparato bastante lento (de 3 a 5 segundos por cálculo), aunque era capaz de realizar cálculos muy complejos. Utilizaba **interruptores** para programarse y leía los datos en cintas de papel perforado.



12

V. La verdadera primera computadora electrónica fue la Computadora Electrónica Numérica e Integradora (Electronical Numerical Integrator and Computer), comúnmente llamada **ENIAC**. Fue construida en la Universidad de Pennsylvania por John Presper Eckert y John William Mauchly, ocupaba una superficie de 167 m² y operaba con un total de 17,468 válvulas electrónicas o tubos de vacío. Era un proyecto secreto que fue presentado al público en 1946.



De ahí en adelante el desarrollo de las computadoras se ligó directamente con el desarrollo de la electrónica. En particular las generaciones de computadoras se diferencian por la forma en que están construidas y por la forma en que interaccionan con el usuario.

1^a Generación (1951-1958).

- Máquinas construidas con tubos de vacío. Eran grandes y costosas (alrededor 10 mil dólares).
- Se programaban en **lenguaje de máquina**. La información se introducía mediante tarjetas perforadas por ingenieros especializados.
- La computadora más exitosa de la primera generación fue la IBM 650, de la cual se produjeron varios cientos. Esta



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



computadora que usaba un esquema de memoria secundaria llamado tambor magnético, que es el antecesor de los discos duros actuales.

2^a Generación (1958-1964)

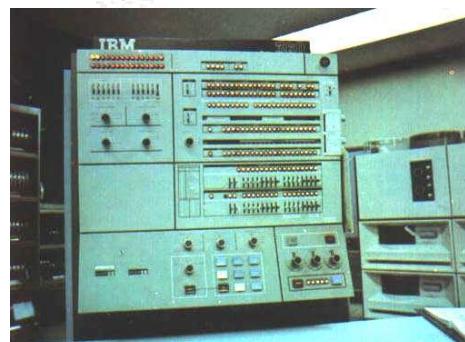
- Se fabricaban con transistores (un invento reciente para la época), disminuyendo su costo y tamaño (y sus inconvenientes de mantenimiento).
- Se programaban en lenguajes de alto nivel. La información se introducía mediante cintas perforadas o un tablero con interruptores.
- También aparecieron el *WordStar* (primer procesador de texto) y el *Visicalc* (hoja de cálculo).



13

3^a Generación (1964-1971)

- Se fabricaban con Circuitos Integrados que en sí mismos realizaban la tarea de cientos o miles de transistores, reduciendo su tamaño, a la vez que también se reducían sus costos y consumo de energía.
- Aparecen los Sistemas Operativos, y por consiguiente la industria del software.



- El Sistema Operativo de la IBM 360 llamado OS (Operating System) se convirtió en un estándar.
- Aparecieron las primeras computadoras de bajo espectro (dirigidas al usuario casero) llamadas *minicomputadoras*.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



CACEI



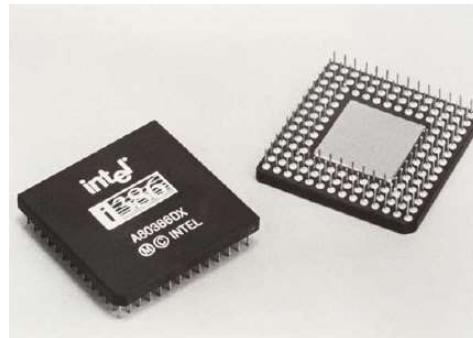
ITSZ

ESTUDIOS DE
GRADUACIÓN
PROFESIONALESTUDIOS DE
GRADUACIÓN
PROFESIONAL

CACEB A.C. COMEX A.C.

4^a Generación (1971-1982)

- La fabricación ahora se realizaba a través de componentes llamados microprocesadores, que en sí mismos son un “cerebro”. Miniaturizan el espacio antes usado por los Circuitos Integrados y son más veloces.
- Los sistemas operativos se volvieron más entendibles al usar palabras especiales (del idioma inglés) para gestionar su uso. De entre ellos el más famoso sería el MS-DOS, manufacturado por la recién creada empresa Microsoft; este sistema operativo es el antecesor directo de los sistemas Windows.
- En agosto de 1981 la IBM presentó su nuevo modelo comercial llamado *IBM PC* cuya patente de arquitectura hoy en día es usada por todas las empresas fabricantes de computadoras (exceptuando las Apple). Ésta es la razón por la que se les llama PC's (Personal Computer, Computadora Personal).

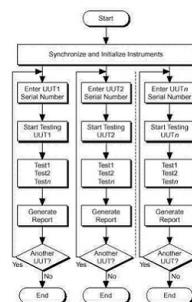


14



5^a Generación (1983- Actualidad)

- Dado el rápido avance de la tecnología en Japón, en el año de 1983, se realizó una propuesta que define los objetivos que se persiguen en el desarrollo de nuevos equipos informáticos: **Procesamiento Paralelo** y el manejo de lenguaje natural y de inteligencia artificial.
- El proceso paralelo se refiere a la ejecución de varias tareas a la vez por instante de tiempo. Actualmente las computadoras usan un sistema secuencial que resuelve una instrucción por vez.
- La inteligencia artificial se refiere a la capacidad que debería tener una computadora para tener conciencia y dialogar de forma natural con un ser humano.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

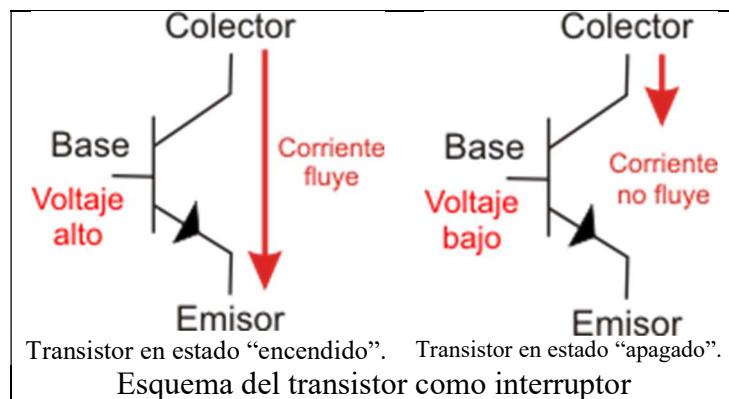
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

1.4. Arquitectura básica de una computadora

Cuando se habla de arquitectura de computadora nos referimos a la manera en que se han organizado todas sus partes para un proceso eficiente. De este modo, es claro que la relevancia de una computadora se encuentra implícitamente relacionada con la manera en que ésta puede retener y procesar información. A su vez, la información que ésta maneja sólo es una representación electrónica de los datos que se le proporcionan.

1.4.1. El Bit y el Byte

Una computadora opera mediante un sistema de códigos, el cual está basado en una idea bastante simple: la dualidad de respuestas opuestas, el **sí** y el **no**, el **verdadero** y el **falso**, el **cero** y el **uno**. El aparato en sí mismo opera como si tuviera interruptores de **encendido** y **apagado**, pero controlados y operados de una forma especial. El transistor de hecho es una especie de interruptor, cuando el voltaje con que se alimenta es suficientemente alto deja pasar corriente, pero si el voltaje es bajo (como el de una pila ya gastada) entonces ya no deja pasar corriente por sus terminales.



Así, cuando el transistor está “apagado” entonces se tiene un cero, un no, un falso; y cuando está “encendido” se tiene un uno, un sí, un verdadero. Esto último es justamente la idea del **bit**, la posibilidad de representar un uno (1) o un cero (0) con dichos estados del transistor. Podría decirse entonces que el mismo transistor es un bit.

Sin embargo, con un solo bit no se podría hacer gran cosa, porque tan sólo para representar cantidades (que no números) los humanos usamos un sistema de códigos basados en combinaciones de diez dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), para representar los sonidos de que se componen las lenguas se usa un alfabeto (que no es más que otro conjunto de códigos) y qué hablar de los símbolos de puntuación, exclamación, interrogación, etc.

Examinando la idea que se sigue para escribir nuestros números sin tomar en cuenta la cantidad que representan, veremos que la regla que se sigue es la de combinar cada uno de los símbolos de forma tal que nunca se repitan, así, cada una de las combinaciones representa una cantidad diferente.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Símbolos Numéricos Básicos

0	1	2	3	4	5	6	7	8	9
Algunas combinaciones de símbolos representando cantidades									
10	11	12	13	14	15	16	17	18	19
20	21	...	29	30	31	...	55	...	99
100	101	...	999	1000	1001	...	10000	10001	...

Esa misma idea ha sido usada en la creación de lo que llamaremos **Lenguaje de Máquina**. El lenguaje de máquina se basa en la combinación de ceros y unos (voltajes altos y voltajes bajos) para representar todo en una computadora. Veamos cómo puede darse esto: Representemos por ésta vez al bit con un foco (página siguiente):



Si usamos dos focos en conjunto podríamos obtener las siguientes combinaciones

Combinaciones de Bits Código

		00
		01
		10
		11

Reflexionemos entonces:

- Con un bit sólo tenemos dos códigos: 0 y 1. Aunque también podríamos pensar en el Falso y Verdadero, en el No y el Sí, e incluso en el Negro y el Blanco.
- Con dos bits obtenemos hasta cuatro códigos, donde cada uno podría representar una idea diferente: una cantidad, un símbolo, un color. Examinemos algunas de esas ideas en la tabla siguiente.

Código	Cantidad	Color
00	0	Negro
01	1	Gris Oscuro
10	2	Gris Claro
11	3	Blanco



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Es evidente que a mayor cantidad de bits tendremos una mayor cantidad de códigos y por consiguiente una mayor cantidad de conceptos que se pueden representar.

Así es como nació el concepto del **Byte**. Al desarrollarse estas ideas se llegó a la conclusión de que todas las grafías del lenguaje humano (en particular del inglés) ni siquiera llegan a 256. Por consiguiente, se estableció un estándar de 8 bits para codificar cualquier símbolo del lenguaje humano, puesto que como cada bit tiene dos códigos posibles, entonces con 8 bits tenemos $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 256$ códigos. A ese conjunto de 8 bits se le llama **Byte (1 B = 8 b)**.

En 1963 el Comité Estadounidense de Estándares (ASA, conocido desde 1969 como el Instituto Estadounidense de Estándares Nacionales, ANSI) estableció la tabla de códigos para cada grafía conocida como **ASCII** (American Standard Code for Interchange Information, Código Estándar Americano para el Intercambio de Información). A continuación, algunos ejemplos de códigos ASCII.

Decimal	Código Binario	Símbolo
64	0100 0000	@
92	0101 1100	\
126	0111 1110	~

El byte es usado a partir de entonces para saber la capacidad de memoria de un equipo porque representa el espacio en el que se puede escribir un carácter; en términos de electrónica: la cantidad de transistores que se tienen para representar los códigos de cada carácter. A mayor cantidad de transistores, mayor cantidad de bits, mayor cantidad de bytes y mayor cantidad de caracteres.

Ahora bien, el byte no sólo se usa para representar caracteres, también se usa para representar códigos de sonido y color. El audio se codifica a través de la digitalización (en bits) de las frecuencias de sonido. Las imágenes se codifican mapeando los colores en códigos de 32 bits de longitud.

1.4.2. Diversas medidas de Bytes

A medida que las computadoras han ido evolucionando, la capacidad de memoria ha ido en aumento, por lo que el Byte como unidad de medida ha quedado empequeñecida, debido a lo cual ahora se acostumbra usar sufijos para indicar capacidades mayores. Estos prefijos han estado en uso en varios sistemas de unidades, por ejemplo, para unidades de peso se utiliza el **kilogramo**, para longitudes más o menos grandes se usa el **kilómetro**. Cada prefijo aumenta la proporción de la unidad en 1000 acuerdo con un sistema de notación científica.

Cantidad	Prefijo	Símbolo
1,000	kilo	k
1,000,000	Mega	M
1,000,000,000	Giga	G
1,000,000,000,000	Tera	T



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

En el área de las ciencias de la computación esto no es del todo cierto, ya que el sistema de numeración es binario, tradicionalmente se tomó la convención de que aquella potencia mínima de 2 que más se aproxime a la notación será quien tome ese prefijo (por ejemplo: 1 kB=1024 B). Sin embargo, ésta es una situación que a día de hoy sigue causando confusión. En 1998 se trató de solventar esta situación creando un sistema de prefijos combinando las unidades del Sistema Internacional con la palabra binario, en donde, por ejemplo: 1 KiB= 2^{10} B (KiB es la contracción de Kilo Byte Binario), desafortunadamente no todos los sistemas operativos han adoptado ésta resolución, Microsoft Windows, es el más claro ejemplo de ellos.

Para los fines de este curso esto no es algo que sea relevante, sin embargo, tomaremos la noción del SI, es decir, al hablar de kilobytes diremos que 1 kB= 1000 B, al hablar de megabytes diremos que 1 MB= 1000 kB= 1000000 B, etc.

Algunos de los ejemplos son los siguientes:

- El antiguo disquete de 3½ tenía una capacidad de 1.44 MB, haciendo cuentas $1.44 \times 1,000,000 = 1,440,000$ Bytes (sustituimos la M por su valor correspondiente y multiplicamos).
- 2.88 GB equivale entonces a 2,880,000,000 B.
- 0.024 kB equivale a 24 B.

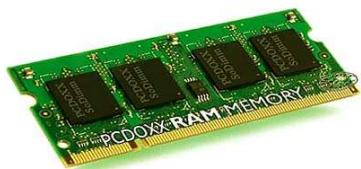
Hoy en día el Byte ya no sólo se usa de referencia de medida en una computadora, sino en cualquier dispositivo electrónico que tenga la capacidad de retener información, sea ésta temporal o no.

1.4.3. Tipos de Memoria

Una computadora maneja varios tipos de memoria, pero para fines prácticos las reduciremos a tres:



Memoria ROM: Usada por la computadora para constatar que cada una de sus partes electrónicas funciona correctamente.



Memoria RAM (también llamada memoria temporal): Es la memoria que usa la computadora para ejecutar programas y abrir archivos. Es volátil (si se apaga la computadora la RAM se borra) pero es muy rápida (porque opera con electricidad pura).



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



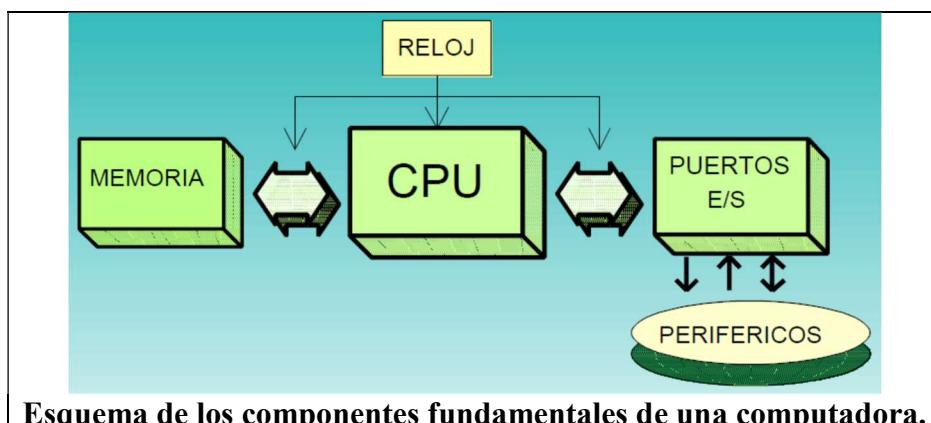
Memoria Permanente: Representada por el Disco Duro es la memoria que no se borra por apagar el equipo. El disco duro es lento con respecto a la velocidad de la electricidad porque tiene una parte mecánica (el cilindro giratorio), pero la información queda “permanentemente” grabada en forma de campos magnéticos.

19

Podríamos incluir también en este contexto a la memoria **caché**, que es una clase de memoria mucho más rápida que la RAM, pero más pequeña en bytes, que sirve como intermediaria en el traslado de la información entre los diversos componentes del sistema.

1.4.4. Arquitectura Von Neumann y Arquitectura Harvard

Los dispositivos fundamentales de una computadora son: memoria, unidad central de procesamiento y puertos de comunicación, todos ellos trabajando en forma síncrona mediante un sistema de reloj. Una manera simplificada de representar los componentes de una computadora es la que se muestra en la siguiente figura.



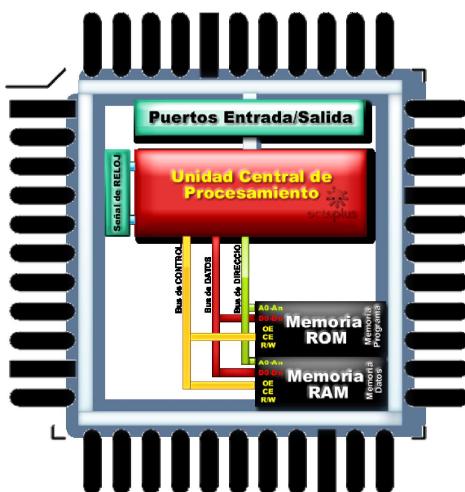
Las flechas anchas bidireccionales ahí mostradas representan los elementos por los que se hace el intercambio de información entre componentes, llamados **buses**.

Al proponer distintas maneras específicas de organizar la interacción de los componentes se obtienen distintos tipos de arquitecturas.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

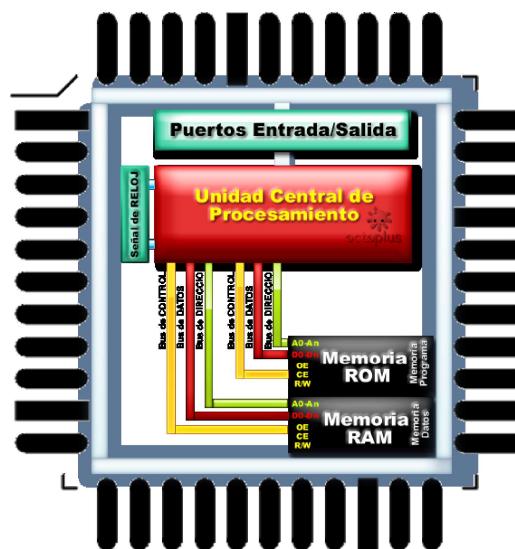
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Otra clase de arquitectura es la Harvard; en esta estructura la memoria de programa (pasiva, no cambiante) recibe un tratamiento diferente que la memoria de datos (activa), pudiéndose llegar a una total diferenciación entre los buses de comunicación: bus de datos y bus de instrucciones. La memoria de instrucciones y la memoria de datos no se encuentran en el mismo bloque de memoria general, sino que se encuentran separadas, de tal modo que cada una tiene sus propios buses. Esta arquitectura es usada en dispositivos específicos llamados microcontroladores.

Debería ser evidente que la potencia de una computadora no depende solamente de su unidad de procesamiento, sino también de su capacidad en memoria y en la manera en que ésta se gestiona.

La arquitectura de computadora usada en las computadoras de tipo compatible (PC) se denomina arquitectura **Princeton** o **Von Neumann** (por su desarrollador). Tiene la característica de que en la unidad de memoria no se hace distinción entre datos (información) e instrucciones por lo que comparten los dispositivos de intercambio (buses).



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

2. Diseño de Algoritmos

En ésta unidad el estudiante:

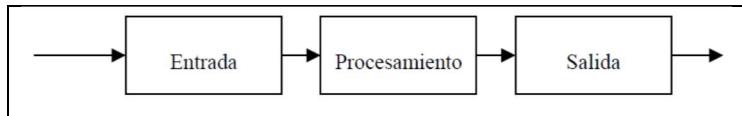
- Aplica diagramas de flujo y pseudocódigos.

2.1. Conceptos Básicos

Una computadora es un **sistema**, es decir, un conjunto de dispositivos que se relacionan entre sí para realizar alguna tarea específica.

Un **proceso** es un conjunto de actividades que se realizan con el fin de desarrollar una tarea específica.

Entonces la computadora es un sistema que realiza procesamiento de la información.



El proceso consta de componentes, a saber:

- **Entrada:** Es la parte del proceso que se encarga de recibir los datos que serán usados en el mismo.
- **Procesamiento:** Parte fundamental del proceso en la que los datos de entrada son “trabajados” para obtener información (o datos de salida).
- **Salida:** Parte del proceso en la que la información obtenida es presentada.

Un **algoritmo** es un conjunto ordenado y finito de pasos usados en la solución de un problema o en la realización de una tarea, por lo que está íntimamente relacionado con los procesos, incluso consta de las mismas tres etapas: Entrada, Proceso, Salida.

En el área de la computación los **lenguajes algorítmicos** proporcionan metodologías para la representación de problemas que posteriormente pueden expresarse en algún lenguaje de programación.

Ejemplo 2.1: Algoritmo que proporciona el área de un triángulo.

- Proporcione longitud de la base=
- Proporcione altura=
- Area=base*altura/2
- Resultado= Area

En algunos casos nos encontraremos con situaciones en las que:

- No se ocupan entradas o no se ocupan operaciones, pero todos ocupan salida.
- Una formula grande o muy compleja puede ser más segura y fácil de resolver, si es descompuesta y resuelta en partes, juntando al final los parciales para obtener el resultado final.
- Un problema puede tener más de una solución correcta.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

- El problema no está suficientemente explicado o enunciado, entonces, estudiarlo, analizarlo y construirlo de manera genérica.

Ejercicios 2.1. Analice los siguientes problemas.

- I. Convertir millas a kilómetros.
- II. Convertir 125 metros a centímetros.
- III. La Sra. López y sus 8 hijos solo compran una vez al mes su mandado en conocido supermercado, en dicha tienda el kilogramo de frijol cuesta \$8.75, el paquete de tortillas cuesta \$3.55 y el frasco de café vale \$14.25, si sólo compran de estos tres productos para su mandado, calcular su gasto total.
- IV. La distancia Tijuana - Ensenada es de 110 kilómetros. Si un automóvil la recorre a una velocidad constante de 30 millas por hora, ¿cuánto tiempo tarda en llegar? (1 milla = 1.609 Km).

22

Por lo anterior, es claro que en la resolución de un problema se pueden proponer distintos algoritmos, pero cada uno tendrá distintos niveles de exactitud y/o rapidez. Así que un buen algoritmo debería cumplir algunos requisitos:

- Ser preciso, debe indicar el orden de realización de forma específica.
- Estar definido, si se ejecuta el mismo algoritmo varias veces debe obtenerse el mismo resultado cada vez.
- Ser finito, debe terminar el proceso en un determinado momento.

2.2. Datos

Un dato es una secuencia de símbolos (letras, números, etc.) que representan un valor y que por sí mismos no tienen un significado.

Otro concepto propio de la computación es el sentido formal de la palabra Información, que en pocas palabras es el proceso que se hace de los datos. Por razones prácticas los datos con los que trabaje una PC deben estar bien definidos ya que posee recursos limitados (es un dispositivo discreto). De este modo los datos deben estar clasificados en rangos de valores que definen las operaciones que se pueden realizar con ellos. Cada tipo de rango define un **Tipo de Dato**.

Podríamos enumerar tres tipos de datos:

- **Numéricos:** Los que sólo pueden contener números, que en la mayoría de los casos serán enteros y decimales. Las variables que representen datos deberán representarse simplemente con su nombre: A, X1, edad, etc.
- **Alfanuméricos:** Son los que trabajan expresamente con letras y números, es decir caracteres. Denotaremos a las variables alfanuméricas con un signo de pesos (\$) precediendo a la variable: \$nombre, \$Y2, etc.
- **Lógicos:** Sólo permiten almacenar valores de verdad (V o F), los cuales generalmente son devueltos por una condición o pregunta.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

2.3. Operaciones

En los algoritmos se pueden realizar operaciones aritméticas, relacionales, lógicas y de asignación. Por lo general cada operación queda representada por símbolos específicos.

Operaciones Aritméticas: Son las que se realizan con números formalmente dichos. Las operaciones aritméticas son sensibles al nivel de precedencia, es decir, cuáles operaciones se ejecutan antes que las otras.

Precedencia	Operador	Operación
Más alta	\wedge	Potencia o Radicación
	*	Producto
	/	División
	+	Suma
	-	Resta
Más baja	$\%$	Módulo

El problema de no tomar en cuenta la precedencia en los operadores casi siempre conduce a resultados equivocados. Por ejemplo, ¿cuál es el resultado de operar $2+3*4$? Si dijo 20, es incorrecto.

Para romper la precedencia en una operación aritmética se usan los paréntesis circulares. Para que en el ejemplo anterior el resultado fuera 20 se debería escribir $(2+3)*4$. De este modo en una línea de operaciones se ejecuta primero lo que está entre paréntesis, si existen varios paréntesis anidados se ejecutan primero los más internos.

Operaciones Relacionales: Son las operaciones en las que se investiga las condiciones de tricotomía de un par de números. En estos casos la respuesta se corresponde con valores de afirmación: Sí o No (¿le recuerda al concepto del bit?)

Símbolo	Operación
$=$	Igualdad
$>$	Mayor que
$<$	Menor que
\geq	Mayor o igual que
\leq	Menor o igual que
\neq	Diferente de, No es igual a

Operadores Lógicos: Son las operaciones que permiten evaluar las condiciones de verdad (cierto o falso, ¿otra vez el bit?) de proposiciones.

Precedencia	Operador	Operación
Más alta	NOT	Negación
	AND	Conjunción
Más baja	OR	Disyunción

Operador Asignación: Representa la acción de darle algún valor numérico específico a un objeto (letra, palabra o símbolo).

Símbolo
 \leftarrow

2.4. Expresiones

Computacionalmente hablando los datos consisten de información **variable** o **constante**. Una variable es entonces un objeto (dato en la memoria de la computadora), creado por un desarrollador de aplicaciones, que tiene la cualidad de poder cambiar de valor durante la



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

ejecución o funcionamiento de un programa. Las constantes representan datos cuyo valor no cambia a lo largo de la ejecución o funcionamiento de un programa.

Las expresiones se componen de operandos (variables o constantes) y operadores. A continuación, citamos algunos ejemplos de expresiones. Compruebe los resultados indicados.

24

Valores asignados	Expresiones	Resultado
A ← 3	A+C-D	12
B ← 5	B+C/2	10
C ← 10	A+D-B^2	-21
D ← 1	A+B==3	0

Ejercicios 2.2. Obtenga los resultados de las siguientes expresiones. Dados: A←2, B←4, C←10, D←1

- I. $B+C+D/2$
- II. $(B+C+D)/2$
- III. $A^2/2+D-B/2^2$
- IV. $((A+B)=3) \text{ AND } (C^3>1000)$

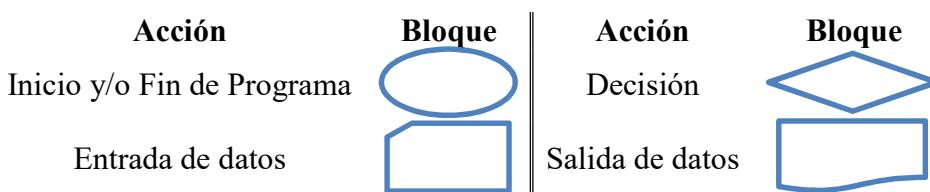
2.5. Diagramas de Flujo

Un diagrama de flujo es la representación gráfica de un algoritmo, por lo que cada parte de un algoritmo tiene una representación en un diagrama de flujo. Son aplicables a cualquier lenguaje de programación.

Los diagramas de flujo permiten clarificar el proceso de un algoritmo, pero a cambio requieren de espacio para ser dibujados.

2.5.1. Simbología Básica

No existe una simbología estándar para representar diagramas de flujo, ya que cada autor suele proponer sus propios elementos gráficos, por lo tanto elegiremos, para los propósitos de este curso, aquellos que suelen usarse más frecuentemente, los cuales presentamos a continuación.



Cada bloque va unido mediante una flecha que indica la dirección de flujo del proceso.

Ejemplo 2.2. Elaborar un diagrama de flujo que lea dos valores numéricos, calcule su suma e imprima el resultado en pantalla.

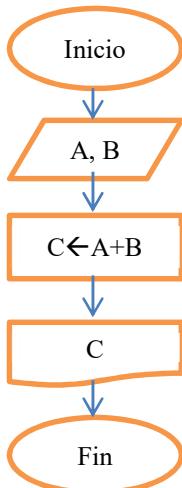


"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Algoritmo

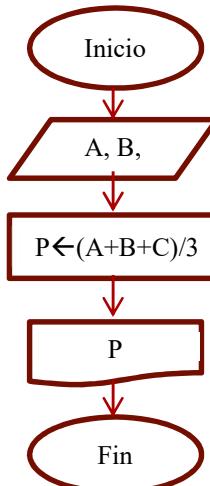
- i. Entrada: A, B
- ii. Proceso: $C \leftarrow A+B$
- iii. Salida: C



Ejemplo 2.3. Diseñar un diagrama de flujo que lea tres datos numéricos, calcule su promedio e imprima el resultado en pantalla.

Algoritmo

- i. Entrada: A, B, C
- ii. Proceso: $P \leftarrow (A+B+C)/3$
- iii. Salida: P



Para que un diagrama de flujo sea funcional y eficiente debe cumplir con ciertos parámetros:

1. Todo diagrama debe tener un principio y un fin.
2. Las líneas de conexión siempre deben ser rectas, y si es posible que sean sólo verticales y horizontales (no deben cruzarse ni estar inclinadas). Así mismo, se deben usar los conectores sólo en casos estrictamente necesarios.
3. Las líneas que enlazan símbolos entre si deben estar todas conectadas.
4. Se deben dibujar todos los símbolos de modo que se pueda seguir el proceso visualmente de arriba abajo (diseño top down) y de izquierda a derecha.
5. Realizar un diagrama claro y estructurado procurando que la parte central del diagrama sea la parte central de la hoja de papel.
6. Evitar la utilización de terminología específica de un lenguaje de programación.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



7. En las operaciones lógicas recurrir preferentemente a la lógica positiva y que a la lógica negativa.

Utilizar diagramas de flujo tiene ventajas y desventajas. Entre las ventajas tenemos:

- Rápida comprensión de las relaciones.
- Análisis efectivo de las diferentes secciones del programa.
- Documentación adecuada de los programas.
- Codificación eficaz de los programas.
- Depuración y pruebas ordenadas de los programas.

26

Las desventajas son:

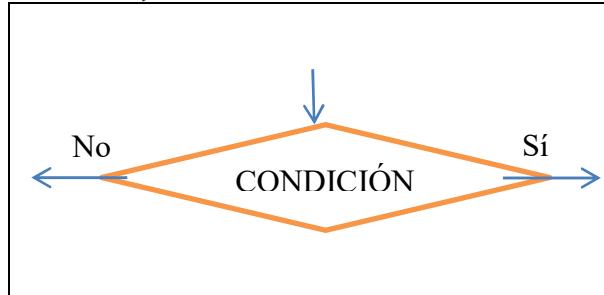
- Los diagramas complejos y detallados suelen ser complejos en planteamiento y elaboración.
- No existen normas fijas para la elaboración de diagramas de flujo que permitan incluir todos los detalles que el usuario desea introducir.

Ejercicios 2.3. Realizar los diagramas de flujo que resuelvan los siguientes problemas.

- i. Leer una cantidad en pesos y convertirla en dólares (suponiendo que el dólar vale 12 pesos).
- ii. Leer una cantidad en $^{\circ}\text{C}$ y convertirla en $^{\circ}\text{K}$ y $^{\circ}\text{F}$ ($^{\circ}\text{C} = ^{\circ}\text{K} - 273$, $^{\circ}\text{C} = (\text{F} - 32)/1.8$).
- iii. Que lea los catetos A y B de un triángulo rectángulo y que calcule e imprima la hipotenusa.

2.5.2. Bloques de decisión en un diagrama de flujo

Los diagramas de flujo de los ejemplos 1 y 2, se ejecutan de manera incondicional y de forma consecutiva, de arriba hacia abajo. Sin embargo, en ocasiones será necesario controlar la manera en que se produce el flujo del proceso, lo cual se realiza con un bloque especial, mostrado en la figura a la izquierda.



Cuando el flujo llega al bloque, en él se prueba una condición, que dependiendo de su validez se sigue el proceso, por un lado, derecho, si, como en este caso, la prueba resulta ser verdadero; o, en sentido contrario (izquierdo) si la prueba resulta ser falsa.

Ejemplo 2.4. Diseñar el diagrama de flujo que decida si una persona puede votar o no (en México).

Algoritmo

- I. Entrada: Edad
- II. Proceso: Si $\text{Edad} \geq 18$
 - a. Entonces: "Puede votar"

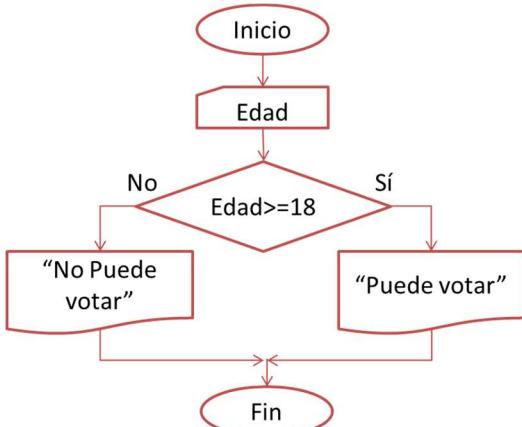


"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

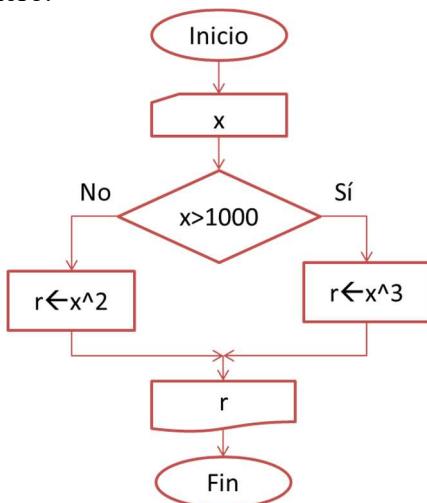
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



- b. **Si no:** “No puede votar”
III. Fin.



Ejemplo 2.5. Diseñar el diagrama de flujo en el que se lea un número, y se verifique si es mayor a 1000, en cuyo caso deberá mostrar en pantalla el cubo del número, y si no, que muestre el cuadrado del número.



Ejercicios 2.4. Realice los diagramas de flujo indicados en los siguientes problemas.

- i. Realizar un diagrama de flujo en el que se lean dos números y se pruebe la propiedad de tricotomía.
- ii. Diseñe el diagrama de flujo en el que se lea un número y se pruebe si es par o no, el diagrama debe indicarlo con los mensajes respectivos.

2.5.3. Estructuras iterativas

En la lógica de programación existen procesos que tienen que repetirse de manera predeterminada una cierta cantidad de veces. A cada repetición del grupo de tareas se le



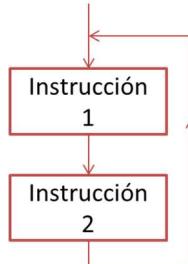
“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

llama iteración o bucle. Entonces, cuando hablamos de iteraciones en diagramas de flujo, hablamos de estructuras iterativas.

La estructura consta de una entrada, que puede estar compuesta de varias instrucciones; y una salida que se realizará dada una condición (sea que se cumpla o no).

Las iteraciones que se ejecutan indefinidamente se llaman bucles infinitos, los que en la lógica de programación no son aceptables, dado que supone que la computadora nunca acabaría sus procesos y por lo tanto nunca podría devolvernos información.



28

Así que los bucles útiles en la lógica de programación son aquellos que son finitos.

Los bucles finitos deben contar con las siguientes partes:

1. Preparación y arranque.
2. Cuerpo.
3. Modificación.
4. Control de la condición.

Existen tres clases fundamentales de estructuras iterativas:

Estructura

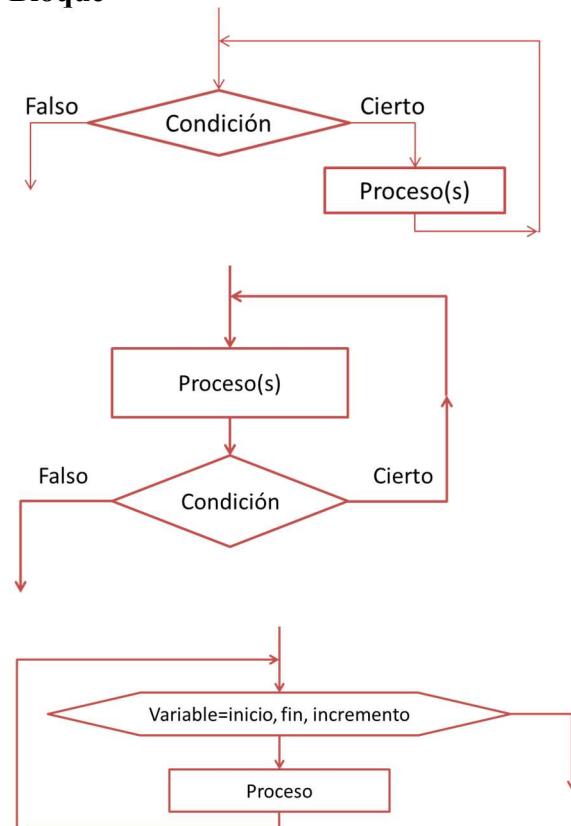
Mientras-Hacer.

Esta clase de bloque analiza una condición en primer término, y en tanto ésta se cumpla realiza el proceso repetitivo. Un caso particular sería que si en un primer intento la condición no se cumpliera, entonces el proceso no se ejecutaría ni una sola vez.

Hacer-Mientras.

En este caso se realiza una primera vez el proceso, luego se analiza la condición y si ésta se cumple se repite el proceso. Lo anterior implica que si la condición no se cumpliera, entonces el proceso se realizaría al menos una sola vez.

Bloque



Ciclos automáticos.

Este bloque utiliza una variable entera específica que permite contar las iteraciones que se van realizando. La variable sirve además para indicar una cantidad de inicio de conteo y otra cantidad de finalización, lo que a su vez implica que para utilizar este



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

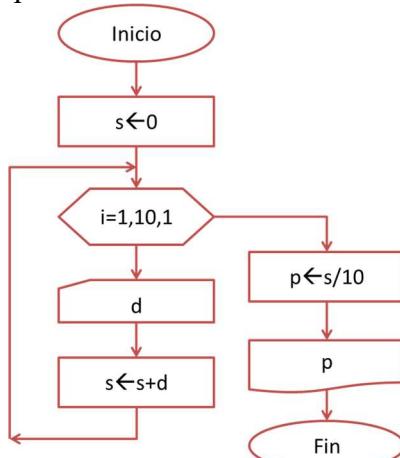


bloque es necesario conocer previamente, cuantas iteraciones totales se realizarán.

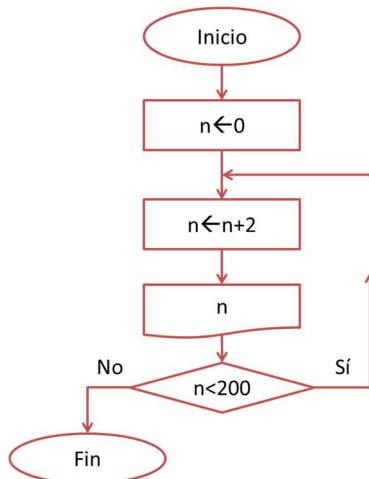
A continuación, presentamos algunos ejemplos de uso de estos bloques.

29

Ejemplo 2.6. Diseñar el diagrama de flujo que lea hasta 10 datos numéricos (es decir que solicite 10 datos) y obtenga su promedio.



Ejemplo 2.7. Diseñar un diagrama de flujo que imprima visualmente números pares de 2 hasta 200.



Ejercicios 2.5. Realice los diagramas de flujo indicados en los siguientes problemas.

- Generar el diagrama de flujo que permita calcular la media aritmética de n números.
- Realice el algoritmo y diagrama de flujo en el que dada la edad de una persona le diga cuantas horas ha vivido.
- Realice un diagrama de flujo en el que se lea un número N , y que genere la



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

siguiente secuencia: 1,4,9,...,n².

2.6. Pseudocódigo

Un pseudocódigo es una aproximación entre un algoritmo y un verdadero código en algún lenguaje de programación. La diferencia entre un pseudocódigo y un lenguaje de programación estriba en que en el pseudocódigo se suelen utilizar palabras comunes al programador (incluso, en ocasiones, en el idioma propio), y no se requieren encabezados para indicar variables y bibliotecas, enfocándose entonces en el proceso.

Como en el caso de los diagramas de flujo, no existe un modo estándar de realizar pseudocódigos, de hecho, cuando se estudia un lenguaje de programación en particular, el autor se toma la libertad de copiar la estructura de tal lenguaje para adaptarlo lo más parecido al del pseudocódigo, con la finalidad de acelerar el aprendizaje.

En la imagen mostrada a continuación se presenta un ejemplo comparativo entre pseudocódigo y su equivalente código en lenguaje Pascal.

Pseudocódigo

Inicio

Escribir mensaje “Teclea el primer número”

Leer variable A

Escribir mensaje “Teclea el segundo número”

Leer variable B

Procesar : hacer la variable S igual a la suma de A+B

Procesar : hacer la variable P igual a S/2

Escribir el mensaje “El promedio es ” y el valor de P

Fin

Código

Program promedio;

Var s,a,b: integer;

p: real;

Begin

write('Teclee el primer numero');

readln(a);

write('Teclee el segundo numero');

readln(b);

s:=a+b;

p:=s/2;

write('El promedio es: ',p:10:2);

End

Para los fines de este curso bastará con la comprensión de los Algoritmos y Diagramas de Flujo, pero debemos conocer también el uso de los pseudocódigos.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

3. Fundamentos del Lenguaje

En esta unidad el estudiante:

- Diseña e implementa programas básicos.

31

3.1. Breve historia del lenguaje C++

El lenguaje C (no confundir con el C++) tuvo sus orígenes en los primeros años la década de los 70's, un programador de nombre Dennis Ritchie (de la AT&T Bell) ideó un lenguaje de programación que le permitiese manejar el hardware como en el Ensamblador y al mismo tiempo usar programación estructurada como en los lenguajes de alto nivel. La primera versión de C corría sobre máquinas PDP-7 y PDP-11 con sistema operativo UNIX. Posteriormente reescribió el propio compilador de C de UNIX en el mismo C, y aún más tarde el propio UNIX se escribió en C.

Cuando AT&T decidió ceder (a un precio bastante bajo) el sistema operativo a varias universidades, empezó el auge del lenguaje C. De ésta manera cuando las PC's se volvieron comerciales aparecieron varias versiones de compiladores de C (de distintas compañías de software).

Dada la situación anterior en 1983 el Instituto Americano de Estandarización (ANSI, American National Standards Institute) decidió estandarizar al C, lo cual se cumplió en 1989 con la ayuda de la Organización Internacional de Normalización (ISO).

En 1998 se definió al estándar C++, que era una evolución de C hacia la programación orientada a objetos, desarrollada por Bjarnes Stroutstrup de la misma AT&T Bell.

3.2. Características básicas del lenguaje C++

Para comprender algunas de las características básicas del lenguaje C++, primero debemos investigar las distintas concepciones que se han usado en el desarrollo de los lenguajes de programación, de entre ellas podemos citar:

- **Programación Spaguetti.** Este término no es técnico, sino más bien peyorativo, debido a lo confuso y enmarañado que resulta analizar el flujo de los procesos en un lenguaje de este tipo. En este tipo de programación el flujo del proceso se realiza mediante saltos incondicionales entre etiquetas que enmarcaban una lista de instrucciones, de tal manera que, en la estructura del código se podía ir desde el principio al final, del final a en medio, de en medio a la mitad, de la mitad al inicio, etcétera, según lo requiriera la "lógica" del programador.

- **Programación Estructurada.** La programación estructurada esta compuesta por un conjunto de técnicas que han ido evolucionando aumentando considerablemente la productividad del programa reduciendo el tiempo de depuración y mantenimiento del mismo.

Se utiliza un número limitado de estructuras de control, reduciendo así considerablemente los errores.

Esta técnica incorpora:



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

- Diseño descendente (top-down): el problema se descompone en etapas o estructuras jerárquicas.
- Recursos abstractos (simplicidad): consiste en descomponer las acciones complejas en otras más simples que son resueltas con mayor facilidad.
- Estructuras básicas, existen tres tipos:
 - Estructuras secuenciales: cada acción sigue a otra acción secuencialmente. La salida de una acción es la entrada de otra.
 - Estructuras selectivas: en estas estructuras se evalúan las condiciones y en función del resultado de las mismas se realizan unas acciones u otras. Se utilizan expresiones lógicas.
 - Estructuras repetitivas: son secuencias de instrucciones que se repiten un número determinado de veces.

Las principales ventajas de la programación estructurada son:

- Los programas son más fáciles de entender.
- Se reduce la complejidad de las pruebas.
- Aumenta la productividad del programador.
- Los programas quedan mejor documentados internamente.

Un programa está estructurado si posee un único punto de entrada y sólo uno de salida, existen de "1 a n" caminos desde el principio hasta el fin del programa, y por último, si todas las instrucciones son ejecutables sin que aparezcan bucles infinitos.

- **Programación modular.** Un programa modular consta de varias secciones divididas de tal forma que interactúan a través de llamadas a procedimientos, los cuales integran el programa en su totalidad. Aquí el programa principal coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros. A su vez cada módulo puede contener sus propios datos y llamar a otros módulos o funciones.
- **Programación orientada a objetos.** Se llama objeto a un conjunto de datos y programas que poseen su propia estructura. Los objetos además se encuentran organizados. Así, esta clase de programación se enfoca en la utilización de dichos objetos, reduciendo de manera considerable el tiempo de desarrollo de programas.

De este modo el lenguaje C++, es una evolución del C original, que se orienta a la filosofía de la programación orientada a objetos, pero conservando características que le permiten seguir operando como un lenguaje de programación estructurada.

Lenguajes de Programación No Estructurada	Lenguajes de Programación Estructurada
FORTRAN	Pascal
Basic	Ada
Cobol	C, C++ Java



Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

También se considera que tanto C como C++, son, en cuanto a la comunicación con el procesador de una computadora, un lenguaje de nivel medio, porque es capaz de utilizar comandos propios de un lenguaje de alto nivel, como comandos que le son propios al lenguaje Ensamblador.

De este modo podemos generar programas de usuario de forma simple, o incluso podemos programar directamente las direcciones de memoria de una computadora para controlar su comportamiento.

Alto Nivel	Ada Modula-2 Pascal Cobol Basic
Nivel Medio	Java C, C++ FORTH Macroensamblador
Bajo Nivel	Ensamblador

Ahora aparecen dos conceptos:

- **Compilador:** Es aquel programa que permite traducir un código en lenguaje de alto nivel (llamado código fuente), en un código cuyo lenguaje la computadora puede interpretar y ejecutar (llamado código objeto o código de máquina). En otras palabras, es aquel que convierte en ejecutables los programas que hemos elaborado.
- **Entorno Integrado de Desarrollo:** conocido por sus siglas en inglés (IDE), es la interfaz (o conjunto de ventanas), que tratan de hacer más confortable la programación en un lenguaje en particular. Posee herramientas y aplicaciones propias que tratan de irnos ayudando a escribir nuestros códigos, resaltando palabras reservadas, señalando nuestros errores y, en algunos casos, haciendo sugerencias para corregir o mejorar el programa en desarrollo.

Ahora bien, no existe un único compilador de C++, sino que existen varios, como ya se ha comentado en una sección anterior, siendo algunos de paga y otros libres, de entre ellos podemos citar a los más comunes:

- De paga: Borland C++, Microsoft Visual C++.
- Libres: GCC, Mingw, Cygwin.

Es de observar que el compilador GCC y Mingw, se basan en el ISO Standard C++.

En cuanto a los IDE's, generalmente aquellos compiladores que son de paga precisamente vienen acompañados de una interfaz de desarrollo, mientras que los que son libres, pueden instalarse independientemente de dichas interfaces, existiendo, por tanto, una buena cantidad de interfaces de desarrollo para el Standard C++.

Como en este curso nos enfocaremos en el uso del lenguaje estándar, proponemos los siguientes IDE's:

- Bloodshed Dev-C++/ wxDev-C++.
- CodeBlocks.
- NetBeansIDE.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```
#include <iostream>

using namespace std;

int main (int argc, char *argv[])
{
    char quit;
    quit = '0';
    while (quit != 'q')
    {
        cout << "Hello ! This is a console app." << endl;
        cout << "To create a console, go to Project Options and select" << endl;
        cout << "'Win32 Console'." << endl;
        cout << "Press q to quit " << endl;
        cin >> quit;
    }
    return 0;
}
```

Interfaz de wxDevC++

Code::Blocks v1.0

Open Files list

Opened Files

Start here

Create a new project

Open an existing project

Visit the Code::Blocks forums

Report a bug

CB_VAR_RECENT_FILES_AND_PROJECTS

Management

Projects Resources Workspace

Messages

Welcome to Code::Blocks!

Interfaz de CodeBlocks

NetBeans IDE 7.0

Aprender y descubrir

Hacer una visita guiada

Probar un proyecto de ejemplo

Lo nuevo en el IDE

Rincón de la comunidad

Mi NetBeans

Demostraciones y tutoriales

Interfaz Gráfica de usuario (GUI) Java y J.

Aplicaciones Java y Java Web

Aplicaciones C/C++

Aplicaciones PHP

Aplicaciones Móviles e Iniciadas

Toda la documentación en línea >>

La nueva en el IDE

Oracle WebLogic Server

Aplicaciones

Enterprise Applications

EJB Modules

Web Applications

Resources

JDBC

Deploying a Web App to Oracle WebLogic Ser.

Interfaz de NetBeans IDE



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



CACEI



CACEB A.C. COMEX

COMEX

3.3. Secciones de un programa básico

Es una costumbre en el aprendizaje de los lenguajes de programación presentar como primer acercamiento un código cuya única finalidad es desplegar en pantalla el mensaje “Hola Mundo”. Aprovechamos éste hecho para mostrar una comparativa entre un código en el estándar del C++, y el C.

35

C++	C
<pre>#include <cstdlib> #include <iostream> using namespace std; int main(int argc, char *argv[]) { cout<<"Hola mundo"<<endl; system("PAUSE"); return EXIT_SUCCESS; }</pre>	<pre>#include<stdio.h> #include<conio.h> int main() { printf("Hola Mundo"); getch(); }</pre>

En la siguiente tabla se da una breve explicación del programa “Hola mundo” presentada arriba.

<pre>#include <cstdlib> #include <iostream></pre>	<p>La palabra include se refiere a la biblioteca (también llamada librería) de funciones necesarias para el funcionamiento del programa, se debe agregar una a una cada biblioteca la siguiente sintaxis: #include <librería_solicitada></p> <p>Las funciones contenidas en las bibliotecas son códigos de carácter utilitario para, entre otras cosas gestionar los recursos de la computadora al crear una aplicación, además de proporcionar otras herramientas específicas para facilitar la programación. Incluso un programador puede crear su propia biblioteca de funciones.</p> <p>En el caso del programa “Hola Mundo” se están incluyendo dos bibliotecas: cstdlib, que es la biblioteca de funciones de propósito general del C++, incluyendo la gestión dinámica de la memoria, generación de números aleatorios, comunicación con el entorno, aritmética entera, búsqueda, clasificación y conversión. Por su parte la librería iostream contiene funciones para el flujo de entrada y salida de datos, es decir, implementa “comandos” para “leer” datos proporcionados por el usuario y para mostrar los resultados del proceso.</p> <p>En el estándar C las bibliotecas se declaran con la extensión que poseen “.h” y, en algunos casos tienen nombres parecidos, en este caso, los correspondientes serían stdio.h para cstdlib y conio.h para iostream.</p> <p>Dado que C++ es un derivado de C, las bibliotecas del segundo pueden usarse en el primero, pero computacionalmente se considera una corrupción de código la realización de éstas combinaciones, por lo que deberemos evitarlo.</p>
using namespace std;	<p>Aquí se indica el espacio de nombres a usarse (estándar en este caso), aplicable en la programación modular y a la programación orientada a objetos.</p>



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



**int main(int argc, char *argv[])**

En particular, `int main()` supone la función (o programa) principal, el cual va a devolver un argumento entero (de error o de no error); los paréntesis indican los parámetros y argumentos que va a recibir la función principal. Cuando no se indica nada entre los paréntesis se indica que la función no los requerirá. Aquí, `argc` es un número que describe la cantidad de argumentos de la función principal, `argv[]` representa un arreglo de punteros en una cantidad de `argc` elementos, donde el elemento `argv[i]` representa el *i*-ésimo argumento entregado al programa.

Note que al final de la línea `int main` aparece una llave de apertura y al final de todo el código una llave de cerradura `{--}`, los cuales encuadran el principio y fin de la función principal, todo código de dicha función deberá ir entre estas llaves. Conforme avancemos en el curso veremos que algunas sentencias también requieren delimitar entre llaves varias líneas de código, además podremos crear otras funciones que igualmente incluirán sus propias llaves de apertura y cerradura.

`cout<<"Hola mundo"<<endl;`

`cout` es una instrucción definida en la biblioteca de funciones `iostream`, permite enviar textos o datos de salida a la consola de ejecución del programa (`cout`= salida de C). Note las comillas que sirven para indicar una cadena de texto. La instrucción `endl` indica que es el final de la línea de salida y que se debe saltar a la siguiente línea. Note el punto y coma al final, toda instrucción termina en un punto y coma.

`system ("pause");`

Es una función especial de la biblioteca cargada e implementada en algunos IDE's, que permite hacer una pausa durante la ejecución del programa, en general nos permite apreciar la salida del proceso. No siempre se incluye ésta línea.

`return EXIT_SUCCESS;`

Equivale al “`return 0;`” de otros compiladores que indica la devolución del argumento entero indicado en `main`, el `EXIT_SUCCESS` o `0` indica que se ha completado el proceso sin errores.

3.4. Variables

En la unidad 2 hablamos de variables y tipos de variables, en C++ es importante el uso eficiente de las variables dado que la creación de una de ellas significa el uso de la memoria disponible en la PC. Crear demasiadas variables puede saturar la memoria y obtener un funcionamiento lento del proceso. Para declarar una variable se sigue la siguiente sintaxis:

`tipo_dato nombre_variable [=valor];`

En la anterior declaración los corchetes indican una parte opcional de la declaración, lo cual significa que podemos asignarle inmediatamente a la variable un valor (concepto conocido como inicialización de variable), o sólo se declara la variable.

Ejemplo 3.1. Programa que crea una variable llamada *variable*, y que le asigna el valor de 5. Finalmente se muestra en la salida de consola el valor asignado a la variable.

```
#include <iostream>
#include <cstdlib>
using namespace std;
```



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx





```
int main()
{
    cout<< "Programa que declara una variable, le asigna valor y la muestra"<<endl;
    int variable;
    variable=5;
    cout << "variable=" <<variable<<endl;
    return 0;
}
```

37

Los nombres de las variables pueden estar formados por combinaciones de letras y números, permitiéndose incluir sólo al símbolo especial guion bajo (_). Recuerde que se distinguen minúsculas de mayúsculas.

Existen ciertas palabras que, en el argot de la programación se denominan **reservadas**, dado que el lenguaje “se las reserva” para su propio uso, por ésta razón dichas palabras no pueden usarse como nombres de variables. La siguiente tabla muestra una lista (no exhaustiva) de palabras reservadas.

and	And_eq	asm	auto	bitand	bitor
bool	break	case	match	char	class
compl.	const	const_cast	continue	default	delete
do	double	dynamic_cast	else	enum	explicit
export	extern	false	float	for	friend
goto	if	inline	int	long	mutable
namespace	new	not	not_eq	operator	or
or_eq	private	protected	public	register	reinterpret_cast
return	short	signed	sizeof	static	static_cast
struct	switch	template	this	throw	true
try	typedef	typeid	typename	unión	unsigned
using	virtual	void	volatile	wchar_t	while
xor	xor_eq				

3.4.1. Tipos de Datos

Aquí en vez de tipos de variables hablaremos de tipos de datos. Llamaremos *tipos de datos atómicos* a aquellos que ya se encuentran predefinidos en el propio lenguaje C++. Se les da este nombre debido a que, dada la naturaleza de la programación orientada a objetos, es posible definir nuevos tipos de datos, a partir de los que ya están predefinidos. Los tipos de datos atómicos abarcan distintos rangos de valores, y, por lo tanto, de distinto tamaño en memoria. Una lista no exhaustiva de variables se presenta en la tabla siguiente.

Tipo	Tamaño en bytes	Intervalo
Short	2	-32768 a 32767
unsigned short	2	0 a 65535



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



unsigned long	4	-2147483648 a 2147483647
Int	Dependiendo del compilador utilizado, puede ser de 2 o 4	-32768 a 32767
unsigned int	2 o 4	0 a 65535
float	4	1.17549535e-38 a 3.402823466e+38 con 8 cifras decimales
double	8	2.2250738585072014e-308 a 1.7976931348623158e308 con 16 cifras decimales
long double	10	
char	1	-128 a 127
unsigned char	1	0 a 255
bool	1	true o false (1 o 0, Verdadero o Falso)

Es posible realizar un programa que permita visualizar los valores límite de algunos tipos de datos, pero requiere utilizar una biblioteca de C: **limits.h**.

Ejemplo 3.2. Programa que muestra los límites de valores para algunos tipos de datos atómicos.

```
#include <cstdlib>
#include <iostream>
#include <limits.h>

using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Limites de tipos de datos"<<endl<<endl;
    cout <<"Entero limite max: "<<INT_MAX<<endl;
    cout <<"Entero limite min: "<<INT_MIN<<endl;
    cout <<"Caracter maximo: "<<CHAR_MAX<<endl;
    cout <<"Caracter minimo: "<<CHAR_MIN<<endl;
    cout <<"Caracter tamaño en bits: "<<CHAR_BIT<<endl;
    return 0;
}
```

C/C++ también permite el uso de datos en formato octal y hexadecimal:

```
int variable1=022;
int variable2=0x12;
```

También se pueden cargar números en formato exponencial: float variable3=2e-6
 ¿Qué se hará en la siguiente instrucción? char nombre='@';



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
 Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

¿Cómo se declararía una constante de tipo char que llevara el símbolo “#”? Explique cómo se almacena el carácter en la variable.

3.4.2. Secuencias de escape en cadenas de caracteres

Se llama secuencia de escape (o secuencia de control) a aquella que combinación de caracteres que permite formatear una salida de texto, como la inclusión de saltos de línea o tabulaciones. Así podemos presentar la siguiente tabla de secuencias.

\n	Salto de línea
\t	Tabulador horizontal
\v	Tabulador vertical
\a	Alerta sonora
\0	Carácter nulo
\b	Mueve el cursor hacia atrás

Cualquier carácter se puede representar con el símbolo “\” seguido del propio carácter o de su respectivo código octal o hexadecimal, por ejemplo, las comillas dobles podrían ser: \"\", o \'\\042', o \'x22'.

3.5. Constantes

En el caso de la Programación, las constantes son valores numéricos que son asignadas a una especie de variable que nunca cambia durante la ejecución de un programa.

Se declaran de forma similar a las variables:

const tipo dato nombre cte=valor;

Ejemplo 3.3. Declaración y uso de constantes en C++.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int radio=5;
    const float pi=3.141592;
    float area;
    cout<<"Programa que calcula el area de un circulo de radio 5" << endl;
    area=pi*radio*radio;
    cout<<"\nEl área es: "<<area<< endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



3.6. Operadores Básicos

A continuación, citamos los caracteres usados en C++ para indicar las operaciones aritméticas básicas.

Operador	Significado
+	Suma
-	Resta
*	Producto
/	División
%	Módulo

40

La operación módulo es una división en la que en vez de devolver un cociente devuelve el residuo. De este modo:

$$7\%2 = 1$$

Un detalle importante de la operación módulo es que, aplicada sobre valores múltiplos es igual a cero:

$$6\%3 = 0$$

3.6.1. Operaciones con caracteres

Al construir operaciones aritméticas con datos de tipo carácter, el compilador toma en cuenta el código binario del dato y con él realiza las operaciones.

Ejemplo 3.4. Programa en que se realizan operaciones aritméticas con datos tipo char.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    char car1='a', car2, car3;
    cout<<"Programa que muestra operaciones con caracteres"<<endl;
    car2=car1+10;
    car3=car1-10;
    cout<<"\nCon Car1= "<<car1<<endl;
    cout<<"\nCar2="<<car2<<"\tCar3="<<car3<<endl;
    cout<<"\nLa suma de los tres caracteres es: "<<car1+car1+car3<<"\n"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



3.6.2. Modificadores de operandos

En la programación es usual que se realicen operaciones de incremento y decremento unitario de valores enteros, sobre todo en los casos en que aparecen iteraciones. Estos casos quedan reflejados en un algoritmo como $i \leftarrow i+1$, o bien, $i \leftarrow i-1$. Este proceso se puede simplificar con el uso de modificadores:

41

De incremento (++)	$++x$, x se incrementa en uno antes de que se ejecute la operación en que se le implique. $x++$, x se incrementa en uno después de que se ejecute la operación en donde se le implique.
De decremento (--)	$--x$, x se decrementa en uno antes de que se ejecute la operación en que se le implique. $x--$, x se decrementa en uno después de que se ejecute la operación en donde se le implique.

Ejemplo 3.5. Aplicación de modificadores

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int a=3,b=4,c,d;
    cout<<"Programa que muestra modificadores de operadores"<<endl;
    cout<<"\nLa variable a="<<a<<".\tLa variable b="<<b<<endl;
    a*=2;
    b-=a;
    cout<<"\nAhora:\tta="\<<a<<"\t b="\<<b<<endl;
    c=a++ -3;
    d=++a * --b;
    cout<<"\nPor ello: \tc="\<<c<<"\t d="\<<d<<endl<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

3.6.3. Operadores de Desplazamiento.

Es un concepto enfocado al lenguaje en bajo nivel. En términos de bits el desplazamiento significa el traslado (a izquierda o derecha) de los bits contenidos en un registro memoria. Veamos un ejemplo, tenemos la siguiente representación binaria:

128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	0

Al aplicar un desplazamiento a la derecha del número binario tenemos:

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	0



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

De este modo el desplazamiento a derecha, numéricamente significa una división entre dos. En consecuencia, un desplazamiento a la izquierda es una multiplicación por dos. En C++ se implementan un doble menor (<<) para el desplazamiento a la izquierda, y un doble mayor (>>) para el desplazamiento a la derecha.

42

Ejemplo 3.6. Programa que aplica operaciones de desplazamiento.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Operaciones de desplazamiento";
    int var0=16,var1,var2;
    var1=var0>>1;
    var2=var0<<1;
    cout<<"\n\nVariable="<<var0;
    cout<<"\nDesplazamiento derecho: "<<var1;
    cout<<"\nDesplazamiento izquierdo: "<<var2<<"\n\n";
    return 0;
}
```

3.6.4. Operadores relacionales y operadores lógicos

Los operadores lógicos son aquellos que permiten comparar valores o expresiones. Los usados en C++ se muestran en la siguiente tabla.

Operador	Significado
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual a
!=	Diferente de

Los operadores lógicos realizan una evaluación de las condiciones de verdad. También son llamados operadores booleanos. En C++ hay operadores dirigidos a Bits y a expresiones.

Operadores Lógicos			
De bits		De expresiones	
&	And	&&	And
^	Or exclusiva		Or
	Or Inclusiva		



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



3.6.5. Precedencia de operaciones

Como se ha comentado las operaciones en una computadora se encuentran jerarquizadas, en principio, para respetar las reglas de la aritmética, en segundo lugar, para organizar adecuadamente el proceso de la información en una computadora.

Ofrecemos a continuación una lista de operaciones, ordenadas de mayor a menor jerarquía.

43

Símbolo	Significado	Símbolo	Significado
::	Resolución de alcance	>>	Desplazamiento a la izquierda
++	Incremento sufijo	<<	Desplazamiento a la derecha
--	Decremento sufijo	>>	Desplazamiento a la izquierda
()	Llamada a función	<<	Desplazamiento a la derecha
[]	Elemento de tabla	<	Desplazamiento a la derecha
->	Acceso a miembro desde un puntero	<=	Menor que
.	Acceso a miembro	>	Menor o igual
++	Incremento prefijo	>=	Mayor que
--	Decremento prefijo	==	Mayor o igual
!	Negación lógica	!=	Comparador de igualdad
~	Complemento a uno	&	No igual
-	Cambio de signo (operador unario)	^	Conjunción de bits
+	Identidad (operador unario)		Or exclusiva de bits
&	Dirección	&&	Or inclusiva de bits
*	Seguir un puntero		Conjunción lógica
sizeof	Tamaño en bytes	=	Disyunción lógica
new	Reserva en memoria	*=, /=,	Asignación simple
delete	Liberación en memoria	%=, +=,	
(tipo)	Conversión explícita de tipos	-=, <<=,	
.*	Acceso a miembros de clase	>>=,	
->*	Acceso a miembros desde puntero	&=,	
*	Multiplicación	^=, =	
/	División	?:	Asignaciones compuestas
%	Módulo	throw	Excepción condicional
+	Suma	.	Lanzamiento de condición
-	Resta		Separador de expresiones

3.7. Entrada y salida de datos

Las funciones **cin** y **cout** utilizadas forman parte de la librería iostream y definen los, así llamados, flujos de entrada y salida estándar de datos. Una instrucción más es **cerr**, que es el flujo de error estándar asociado a pantalla.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Los elementos operadores que acompañan a los flujos ('<<', '>>') son de inserción y de extracción de datos.

La función cout no requiere conocer el tipo de dato de lo que se va a desplegar, pero lo presenta en función de cómo se ha introducido, así que si se desea que aparezca en un formato en especial se debe considerar implementarlo en el código.

44

Ejemplo 3.7. Fragmento de código que muestra un formateo de salidas.

```
cout<<"Cadena de caracteres"<<endl;
cout<<2+2<<endl; //imprime un entero
cout<<9/2<<endl; //imprime un flotante
cout<<(int)(3.141592+2)<<endl; //imprime un entero
```

Igualmente cin tiene varias formas de aplicarse.

Ejemplo 3.8. Fragmento de código donde se verifican varias formas de usar el flujo cin.

```
cout<<"Distintas formas de uso del flujo cin "<<endl<<endl;
int numero;
char caracter;
float otronumero;
cout<<"Introduzca un numero: ";
cin>>numero;
cout<<endl<<"El numero introducido es: "<<numero<<endl;
cout<<"Introduzca una letra: ";
cin>>caracter;
cout<<endl<<"Letra tecleada: "<<caracter<<endl;
cout<<"Introduzca un número con decimales: ";
cin>>otronumero;
cout<<endl<<"Número flotante introducido: "<<otronumero<<endl;
```

Se le llama **cast** al hecho de impedir que se pierdan las características del tipo de datos de destino, es decir, al tomar un valor con un tipo de dato distinto del requerido en el destino, se toma del valor fuente aquella porción de código simbólico que se adecúe al destino.

Algunos ejemplos se muestran a continuación.

cout<<(int) (3.141592+2)<<endl; //Muestra la suma flotante como entero
int b; //Se declara a b como entero
double a=(double) b; //Se lee a b como double pero sin modificar el tipo de b
int b; double a=double(b); //Aquí double actúa como conversor de b sin modificarlo



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

También existen los manipuladores de flujo que permiten mostrar las salidas numéricas en sus equivalentes decimal, octal y hexadecimal, los cuales funcionan con entrada y salida de datos.

Ejemplo 3.9. Varias maneras de formatear la salida de datos de tipo numérico.

```
cout<<"Varias maneras de formatear salida de datos"<<endl<<endl;
int numero=25, leido;
cout<<"Numero declarado= "<<numero<<endl;
cout<<"Numero en formato hexadecimal: "<<hex<<numero<<endl;
cout<<"Numero en formato decimal: "<<dec<<numero<<endl<<endl;
cout<<"Ahora proporcione un numero: ";
cin>>hex>>leido;
cout<<"El dato leido es: "<<hex<<leido<<endl;
cout<<"que en decimal es: "<<dec<<leido<<endl;
cout<<"y en octal es: "<<oct<<leido<<endl;
```

45

En cuanto a la precisión, los datos flotantes utilizan hasta seis decimales después del punto y redondeando el último de ellos.

Para indicar una precisión específica deberemos cargar la biblioteca **iomanip**, que contiene al manipulador **setprecision()**. También se incluye a la función miembro **precision()**.

Ejemplo 3.10. Formateando la cantidad de dígitos decimales (incluir en el encabezado a la librería iomanip)

```
cout<<"Formateando digitos decimales"<<endl<<endl;
cout<<"Numero original= "<<1.23456789<<endl; //Por default solo 5 digitos mostrados
cout<<"Cambiamos precision a cuatro digitos"<<endl;
cout.precision(4);
cout<<"Al desplegar el mismo numero: "<<1.23456789<<endl<<endl;
cout<<"Encadenamos en un solo flujo la precision y el despliegue:"<<endl;
cout<<setprecision(7)<<1.23456789<<endl;
cout<<"Para que aparezcan todos los digitos:"<<endl;
cout<<setprecision(9)<<1.23456789<<endl;
```

Para mejorar el aspecto de nuestros programas podemos echar mano de algunas funciones como son:

- **setw** que se encarga de hacer aparecer el texto alineado determinados espacios a la derecha, si el número de espacios solicitados para su alineación es menor que el número de caracteres que se imprimirán, no verificará su efecto.
- **setfill** se ocupa del carácter de relleno, el carácter por defecto es el espacio en blanco, pero nosotros podemos especificar el que queramos.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Ejemplo 3.11. Comandos para la inclusión de arreglos de caracteres de relleno.

```
#include <cstdlib>
#include <iostream>
#include <iomanip>

using namespace std;

int main(int argc, char *argv[])
{
    cout<<setw(8)<<setfill('*')<<"hola"<<endl;
    cout<<setw(5)<<"89";
    cout<<setw(8)<<"centavo"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

46



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



4. Estructuras de decisión y control

En ésta unidad el alumno:

- Aplica las estructura de decisión y control en programas.

4.1. Sentencia if-else

Es la instrucción de decisión básica. Se traduce como: “Si (condición) ---- si no ----”. Tiene el siguiente formato:

```
if (condición)
{
    Instrucciones si condición verdadera;
}
[else
{
    Instrucciones si condición falsa;
}]
```

Es de notar que lo que aparece entre paréntesis cuadrados es opcional. El bloque que corresponde al “else” no siempre se incluye, en cuyo caso el punto y coma va en la primera llave de cierre. Si se incluye al bloque “else”, entonces el punto y coma se quita de la primera llave de cierre, y se coloca al final.

Ejemplo 4.1. Programa que, en base a la edad, decide si una persona puede votar o no.

```
int main(int argc, char *argv[])
{
    int edad;
    cout<<"Programa que decide si una persona puede votar o no" << endl;
    cout<<"\nProporcione su edad: ";
    cin>>edad;
    if (edad>=18)
    {
        cout<<"\n\n\tResp: Ud es mayor de edad y puede votar.\n" << endl;
    }
    else
    {
        cout<<"\n\n\tResp: Ud no puede votar.\n" << endl;
    };
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

**Ejercicios 4.1.** Realizar los siguientes programas, en los que:

- i. Dado un número permita decidir es par o impar.
- ii. Dados dos números decida la tricotomía de los mismos.
- iii. Leídos tres números enteros A, B, y C, indique cual es el mayor de los tres.
- iv. Dados tres números enteros A,B, y C, los imprima (los tres) en pantalla de mayor a menor.
- v. Calcule la pendiente de una recta, considerando cuando ésta pudiera ser horizontal y vertical.
- vi. Resuelva una ecuación de segundo grado, tomando en cuenta aquellos casos en los que las raíces pudieran ser imaginarias

4.2. Ciclo while

Esta sentencia prueba una condición y si se cumple ejecuta un juego de instrucciones, y repite la ejecución mientras se siga cumpliendo la condición. Sintaxis:

```
while (condición)
{
    Instrucciones;
    Instrucción (es) de control de fin de ciclo;
};
```

Ejemplo 4.2. Programa que calcula la media aritmética de 10 números.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int calif, suma=0, iteracion=1;
    float media;
    cout<<"Programa que calcula la media aritmética de diez números."<<endl;
    while (iteracion<=10)
    {
        cout<<"\nProporcione el valor ["<<iteracion<<"]= ";
        cin>>calif;
        suma+=calif;
        ++iteracion;
    }
    media=(float)suma/10;
    cout<<"\n\nLa media aritmética es: "<<media<<endl<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Una mejora del programa anterior es: dejar que el usuario ingrese una serie de valores, mientras se va contando cuántos de ellos han sido introducidos; luego, el proceso termina cuando el usuario ingresa algún símbolo o valor específico. Posteriormente realiza el cálculo de la media aritmética.

En el ejemplo mostrado a continuación el ciclo termina cuando el usuario introduce el número -1. Al conteo de iteraciones se le resta uno para eliminar el ciclo en que se introduce el -1.

49

Ejemplo 4.3. Modificación del programa del ejemplo 4.2 para la lectura de cualquier cantidad de valores.

```
while (calif!=-1)
{
    ++iteracion;
    suma+=calif;
    cout<<"\nIntroduzca -1 para terminar o... ";
    cout<<"proporcione el valor ["<<iteracion<<"]= ";
    cin>>calif;
}
cout<<iteracion;
media=(float)suma/(iteracion-1);
cout<<"\n\nLa media aritmética es: "<<media<<endl<<endl;
```

4.3. Ciclo do-while

Tiene la misma finalidad que el ciclo *while*, con la diferencia de que se ejecuta al menos una vez el ciclo, y se sigue repitiendo en tanto la condición sea verdadera.

Sintaxis:

```
do
{
    Instrucciones;
}
while (condición);
```

Los ciclos while y do-while resultan útiles cuando es necesario crear ciclos cuya cantidad de iteraciones desconocemos.

Ejemplo 4.4. Menú de opciones con ciclo *do-while*. Se presenta un esqueleto de menú de opciones a realizar por un programa en C++. El programa simplemente despliega el menú hasta que se digita un valor distinto de los del rango de opciones que aparecen en el menú.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```

int main(int argc, char *argv[])
{
    int opcion;
    cout<<"Programa ejemplo de menú con ciclo do-while"<<endl;
    do
    {
        cout<<"\nElija una opción de la lista"<<endl;
        cout<<"1.Operacion 1"<<endl;
        cout<<"2.Operacion 2"<<endl;
        cout<<"3.Operacion 3"<<endl;
        cin>>opcion;
    }
    while((opcion<1) || (opcion>3));
    cout<<"\nOpción de operación válida"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Ejercicios 4.2. Utilizar tanto con el ciclo *while* como con el ciclo *do-while* para:

- Realizar el programa que permite visualizar la tabla de multiplicar de cualquier número.
- Realizar el programa que permite calcular opcionalmente el área de un rectángulo, triángulo o círculo.
- Modificar el programa anterior para que una vez terminado el cálculo, se pueda realizar un nuevo cálculo.

4.4. Sentencia switch

Es una instrucción que ejecuta tareas de acuerdo a un conjunto de opciones, compara el valor alojado en una variable contra un juego de opciones pre-programadas y ejecuta las tareas asociadas con aquella que resulta igual a dicha variable.

La sintaxis para la sentencia se muestra en el recuadro de la derecha.

La variable indicada debe ser de preferencia una letra (mayúscula o minúscula) o un número entero, y, evidentemente debe poder coincidir con alguna de las que aparecen en las opciones (*case*), aquí representadas por las letras *a* y *b*. Si ninguna de las opciones coincide, se ejecutan las instrucciones asociadas en la opción *default*.

Sintaxis:

```

switch (variable)
{
    case a:
        Instrucciones;
    case b:
        Instrucciones;
        ...
    default:
        Instrucciones;
}

```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

**Ejemplo 4.5. Ejemplo de un menú con la sentencia switch.**

```
{
    cout<<"Ejemplo de menu con switch \n\n";
    int opcion;
    cout<<"Menú de opciones: \n";
    cout<<"1. Opcion 1\n";
    cout<<"2. Opcion 2\n";
    cout<<"3. Opcion 3\n";
    cout<<"Elija una opcion: ";
    cin>>opcion;
    switch (opcion)
    {
        case 1:
            cout<<"\nEjecucion 1\n";
            break;
        case 2:
            cout<<"\nEjecucion 2\n";
            break;
        case 3:
            cout<<"\nEjecucion 3\n";
            break;
        default:
            cout<<"\nOpcion no implementada\n";
            break;
    }
    return 0;
}
```

4.5. Ciclo for

Es quizá el ciclo más conocido y utilizado en los lenguajes de programación estructurada y orientada a objetos. Es la opción más viable cuando se requieren realizar operaciones repetitivas, en donde se conoce el número total de iteraciones.

Sintaxis:

```
for (asignación inicial de conteo; condición; incremento o decremento)
{
    Bloque de instrucciones;
};
```

Ejemplo 4.6. Se desea desarrollar un programa que calcule las coordenadas (de al menos diez puntos) de un proyectil que se mueve en trayectoria parabólica. Las ecuaciones que rigen este movimiento son: $x = v_0 t \cos \theta$, $y = v_0 t \sin \theta + \frac{1}{2}gt^2$.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```
#include <cstdlib>
#include <iostream>
#include <cmath>
using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Calculo de puntos de trayectoria en disparo parabolico";
    int t;
    float x=0, y=0, ang,vi;
    const float g=9.81;
    const float pi=3.141592;
    cout<<"\nIntroduzca el angulo de trayectoria: ";
    cin>>ang;
    ang=ang*pi/180;
    cout<<"\nIntroduzca la velocidad inicial: ";
    cin>>vi;
    for (t=0;t<=10;t++)
    {
        x=vi*t*cos(ang);
        y=vi*t*sin(ang)+0.5*g*pow(t,2);
        cout<<"t= "<<"\tx= "<<x<<"\ty= "<<y<<"\n";
    }
    return 0;
}
```

52

Es de notar en el código anterior la inclusión de la librería llamada **cmath**. Esta biblioteca es una colección de funciones matemáticas que permite calcular, entre otras cosas: senos, cosenos tangentes, potencias, raíces. Observe la manera en que tales funciones son invocadas a lo largo del código.

Ejercicios 4.3. Realizar:

- El programa que permita realizar el siguiente cálculo: $S = \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{50}$
- El programa que calcule la suma de los números consecutivos desde 11 hasta 50.

4.6. Ejercicios Complementarios

A continuación, presentamos una breve lista de ejercicios sugeridos para complementar el aprendizaje de los tópicos vistos en ésta unidad.

- Realizar un programa que lea tres números enteros y decida si se han dado en orden o no.
- Realizar un programa que decida cuál es el mayor de tres números enteros dados.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

- III. Realizar un programa que determine si el carácter asociado a un código introducido es numérico, literal, de puntuación, especial o no imprimible.
IV. El valor de e^x se puede aproximar mediante la suma:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

Escriba el programa que permita aproximar la suma mediante el cálculo de n sumandos, dado un valor de x distinto de cero.

53



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

5. Arreglos y Archivos

En ésta unidad el estudiante:

- Aplica arreglos y archivos en programas.

54

5.1. Arreglos

Un arreglo es una especie de variable capaz de almacenar varios datos. Cuando se crea una variable “común” se está reservando una localidad de memoria para contener un único dato en ella.

Cuando se crea un arreglo se asocia el nombre dado a un conjunto consecutivo de localidades de memoria, tantas como datos pueda contener el arreglo.

Por lo general se usan existen dos tipos de arreglos: Vectoriales (o tipo columna, o lista) y Matriciales (o tipo tabla, o rectangular), aunque no son los únicos tipos de arreglos.

Los datos almacenados en un arreglo se encuentran direccionados, de tal modo que la computadora puede discriminar qué dato hay en cada localidad de memoria. La dirección se determina asociando un valor entero positivo (o por pares, o triadas, etc., de números, dependiendo del tipo de arreglo) al nombre dado al arreglo.

Variable	Valor almacenado	Localidad en memoria
Arreglo[0]	1550	FFFB
Arreglo[1]	130	FFFC
Arreglo[2]	45	FFFD
Arreglo[3]	90	FFFE
Arreglo[4]	76	FFFF

5.1.1. Arreglos Vectoriales

Los arreglos vectoriales son aquellos que podemos relacionar con una columna o lista de datos, de tal modo que, para ubicar un dato almacenado en el arreglo, sólo es necesario asociar un número al nombre del arreglo, para distinguir el número de fila del arreglo.

Aunque los seres humanos acostumbramos contar desde uno, en el lenguaje C++ los datos se contabilizan desde cero.

Nombre de Arreglo:
Dato

Dato[0]
Dato[1]
Dato[2]
Dato[3]

Por la forma de declarar un arreglo vectorial podemos decir que ésta puede ser:

- Estática, se declara en un sólo paso el tipo del arreglo y su tamaño
`Tipo_Dato Nombre_Arreglo[Número_Filas];`
- Dinámica, se declara el arreglo una vez indicado el valor del tamaño (con una variable auxiliar), en una instrucción anterior.
`Tamaño=Valor;`
`Tipo_Dato Nombre_Arreglo[Tamaño];`

También podemos notar que no existen arreglos de tipos de datos mezclados, es decir, todos los datos de un arreglo son del mismo tipo.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



A continuación, presentamos ejemplos de declaración estática y dinámica de arreglos tipo vector.

Ejemplo 5.1. Declaración estática de arreglos: introducción y salida de datos.

```
int main(int argc, char *argv[])
{
    cout<<"Programa que muestra un vector de declaracion estatica"<<endl;
    cout<<"\n\nEl arreglo es de 3 filas:";

    int i;
    int Vec[2]; // Aqui se crea el arreglo con un tamaño fijo
    for (i=0;i<3;i++)
    {
        cout<<"\nIntroduzca V["<<i<<"] = ";
        cin>>Vec[i];
    };
    cout<<"\n\nLos datos introducidos fueron: "<<endl;
    for (i=0;i<3;i++)
    {
        cout<<"\nV["<<i<<"] = "<<Vec[i];
    }
    cout<<"\n\n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

55

Ejemplo 5.2. Declaración dinámica del arreglo presentado en el ejemplo 5.1.

```
int main(int argc, char *argv[])
{
    cout<<"Programa que muestra un vector de declaracion dinamica"<<endl;
    int m,i;
    cout<<"\nIntroduzca cantidad de datos"<<endl;
    cin>>m;
    int Vec[m]; // Aqui se crea el arreglo con el tamaño indicado por el usuario
    for (i=0;i<m;i++)
    {
        cout<<"\nIntroduzca V["<<i<<"] = ";
        cin>>Vec[i];
    };
    cout<<"\n\nLos datos introducidos fueron: "<<endl;
    for (i=0;i<m;i++)
    {
        cout<<"\nV["<<i<<"] = "<<Vec[i];
    }
    cout<<"\n\n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

También es posible la inicialización de arreglos, es decir, el almacenamiento de valores en las variables. Lo anterior se puede realizar de las siguientes maneras:

- Indicando cada elemento, por ejemplo, para 3 elementos:

```
int arreglo_vector[] = {dato1,dato2,dato3};  
int arreglo_vector[3] = {dato1,dato2,dato3};
```
- Colocando en cada espacio el mismo dato, para el mismo ejemplo anterior:

```
int arreglo_vector[3] = {dato};
```

56

Ejercicios 5.1. Practique la creación de arreglos en los siguientes problemas.

- i. Realizar el programa que permita realizar el producto de un escalar por un vector de orden 3.
- ii. Realice un programa que permita opcionalmente sumar y restar dos vectores de orden 3.
- iii. Realice un programa que permita opcionalmente sumar y restar dos vectores de cualquier orden.
- iv. Realizar un programa que permita obtener el producto punto de dos vectores.
- v. Realizar un programa que permita calcular la magnitud del producto cruz de dos vectores en el plano XY

5.1.2. Arreglos Matriciales

Los arreglos matriciales son en realidad arreglos de arreglos, es decir suponen la agrupación de varios arreglos de tipo fila, para formar columnas. De tal forma que para direccionarlos es necesario utilizar un par de números.

Nombre del Arreglo: Dato				
	Col. 0	Col. 1	Col. 2	Col. 3
Fil. 0	Dato[0][0]	Dato[0][1]	Dato[0][2]	Dato[0][3]
Fil. 1	Dato[1][0]	Dato[1][1]	Dato[1][2]	Dato[1][3]
Fil. 2	Dato[2][0]	Dato[2][1]	Dato[2][2]	Dato[2][3]
Fil. 3	Dato[3][0]	Dato[3][1]	Dato[3][2]	Dato[3][3]

Igualmente se pueden declarar de forma estática o dinámica. La sintaxis correspondiente es:

Tipo_Dato Nombre_Matriz[No_Filas][No_Columnas];

Ejemplo 5.3. Introducción y salida de datos en un arreglo matricial.

```
cout<<"Usos basicos de una matriz";  
int i,j,matriz[3][3];  
cout<<"\nIntroduzca los datos de una matriz de 3 por 3";  
for (i=0;i<3;i++)  
{  
    cout<<"\n\nDatos de la fila "<<i+1<<":"<<endl;  
    for (j=0;j<3;j++)
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

```
{
    cout<<"Columna ["<<j+1<<"]= ";
    cin>>matriz[i][j];
}
};

cout<<"\n\nMatriz introducida: \n";
for (i=0;i<3;i++)
{
    cout<<"\n";
    for (j=0;j<3;j++)
    {
        cout<<"\tM["<<i+1<<","<<j+1<<"]= "<<matriz[i][j];
    };
}
cout<<"\n\n";
```

Igualmente es posible inicializar matrices, por ejemplo para un arreglo de 3 por 3:

```
static tipo_dato ArregloMatriz[3][3]=
{
    {dato00,dato01,dato02},
    {dato10,dato12,dato13},
    {dato20,dato21,dato22}
};
```

En este caso la palabra reservada static garantiza que la matriz creada no modifica su forma a lo largo de la ejecución del programa, siempre que se declare y use en la misma función o procedimiento donde fue creado el arreglo.

Ejercicios 5.2. Ejercite la creación de arreglos matriciales con los siguientes ejercicios.

- Realice el programa que permita introducir matrices de cualquier orden.
- Realice el programa que permita calcular opcionalmente la suma, la resta, el producto de matrices (A por B o B por A) y el determinante de matrices de tercer orden. Note que el producto y el determinante sólo son aplicables en ciertos casos.
- Realice el programa que permita calcular el producto de matrices de cualquier orden.
- Realice el programa en el que, almacenados una serie de números enteros no repetidos, sea posible localizar la fila y columna donde se localice uno en específico.

5.1.3. Arreglos de cadenas Char

En el caso de una cadena de caracteres cada literal ocupará un espacio-elemento del arreglo, incluso un espacio vacío ocupa un lugar del arreglo. Además, al preparar un arreglo



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

para cadenas char, es importante tomar en cuenta el carácter no imprimible de “fin de línea”, que vale como un carácter más. También es posible introducir el carácter de fin de línea mediante el modificador “\0”. Ejemplos:

```
char texto[13] = "Hola a todos";
char texto2[5] = {'h', 'o', 'l', 'a', '\0'};
```

58

Relacionado con la creación de arreglos tipo cadena tenemos a función miembro **get()**, que funciona de manera similar al operador de flujo ‘>>’, con dos diferencias: primero, la función *get* es capaz de incluir los caracteres de espacio en blanco, mientras que el operador los excluye (opción por definición). Segundo es menos propenso a causar salidas de flujo vacías (como cout). Una variación en la especificación de la función *get* permite indicar hasta cuantos caracteres deben leerse, contando el delimitador de cadena. Citamos dos formas de usarlo:

cin.get() cin.get(char variable)	Extrae solo un carácter. Lo devuelve tal como está (primera forma) o puede devolver su valor numérico (segunda forma).
cin.get(char variable[n+1], streamsize n) cin.get(char variable[n+1], streamsize n, char delim)	Extrae una secuencia de caracteres y los almacena en el arreglo <i>variable</i> . Este arreglo debe tomar en cuenta el carácter delimitador que contará como un elemento más. Este carácter delimitador se puede indicar, siendo por lo general el del salto de línea ('\n').

Para introducir caracteres desde teclado será mejor emplear la función modificadora del flujo *cin*: **getline()**, porque permite manipular mejor los caracteres y además permite declarar con qué carácter el usuario terminará la cadena. Su sintaxis es:

```
cin.getline(nombre_cadena, longitud_de_cadena, carácter_de_fin);
```

Por lo general se elige como carácter de fin de cadena al Enter ('\n').

El stream *ignore*, extrae una secuencia de caracteres de la entrada y los descarta, tiene tres formas de indicarse:

cin.ignore();	Lee y descarta sólo un carácter.
cin.ignore(n);	Lee y descarta los siguientes n caracteres.
cin.ignore(n, 'carácter');	Lee y descarta los siguientes <i>n</i> caracteres o hasta encontrar el carácter especificado, lo que ocurra primero.



Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

**Ejemplo 5.4. Programa para codificación simple de texto.**

```

const int tam_cad=20;
char cadena[tam_cad];
int clave;
cout<<"Programa que 'codifica' una cadena de texto" << endl;
cout<<"\nIntroduzca el texto a cifrar (20 caracteres max): ";
cin.getline(cadena,tam_cad,'\'\n\'');
cout<<"\n\nNúmero de Código Llave (no más de 3 dígitos): ";
cin>>clave;
for (int i=0; (i<tam_cad) && (cadena[i]!='\'0\'); i++)
{
    cadena[i]+=clave;
}
cout<<"\n\nEl texto cifrado es: " << endl;
cout<<cadena << endl;
cin.ignore();
cin.get();

```

59

Ejercicios 5.3. Resuelva lo que se solicita.

- Realice el programa que decodifique un texto codificado por el método anterior.
- Realizar el programa que decodifique al menos 10 ejemplos de cadenas para un texto cuyo código llave se desconoce.

5.2. Archivos

Para manejar archivos debemos revisar el concepto de **flujo**: dispositivo que consume o produce información. Hasta ahora se han usado flujos estándar: cin, cout.

Los flujos estándar están en la biblioteca iostream. Cuando se usa la función cin.get() se invoca a una función miembro del flujo cin.

Desde el punto de vista de las estructuras el flujo es un objeto, dado que contiene variables y funciones que pueden ser invocadas (un concepto cercano a la programación orientada a objetos, donde estaríamos hablando de clases).

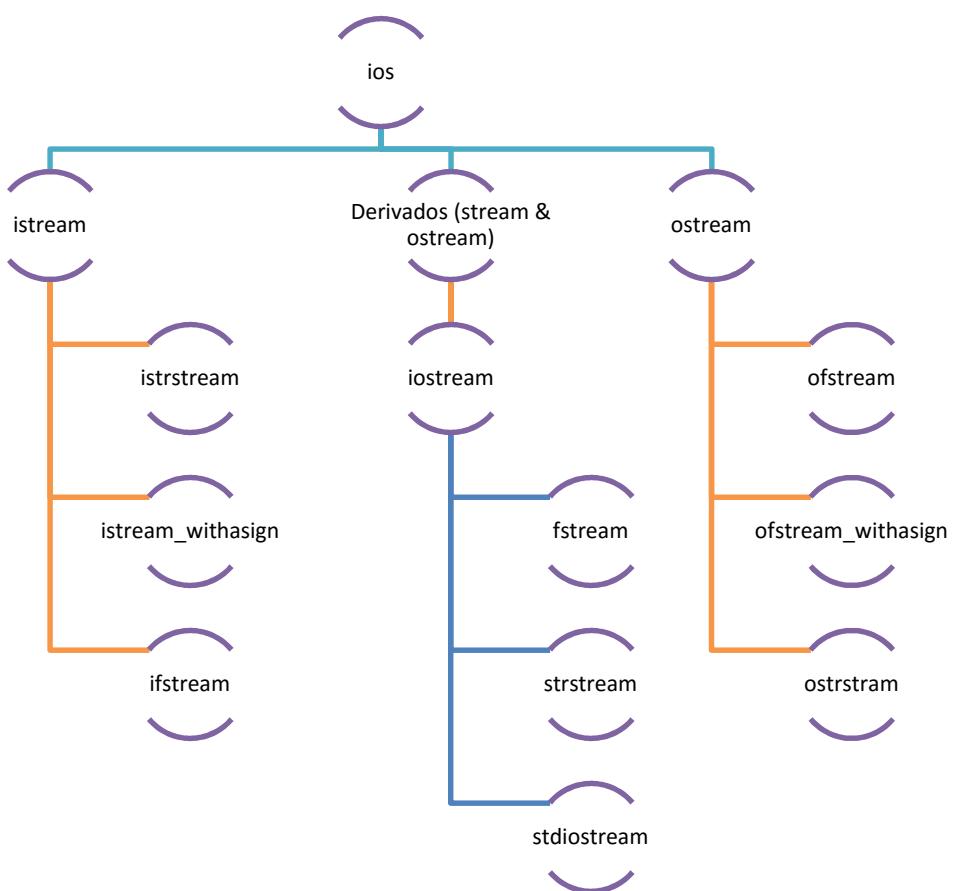
Las estructuras para la entrada y salida de datos tienen un orden jerárquico, para las que existe **herencia**, que significa que una de orden inferior obtiene las mismas variables y funciones que las de orden mayor, además de que se pueden agregar más.

Una categorización para la entrada y salida de datos se presenta en la figura presentada a continuación.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



60

Para trabajar con archivos necesitaremos trabajarlos justamente como flujos, para lo cual será necesario incluir la librería **fstream** y según la utilización que queramos dar a este fichero (lectura o escritura) deberemos declarar el tipo de flujo. Entonces, para crear un archivo de salida declaramos una variable de tipo **ofstream**, la cual forma parte de la librería comentada.

En el ejemplo presentado a continuación, declaramos a “archivo” como variable tipo **ofstream**, y posteriormente utilizamos su función miembro **open** para asociarla a un archivo. Note que la variable “archivo” instancia al archivo llamado “myfile” y que se ubica en la ruta: “D:\Trabajo”, “D” es la unidad y “Trabajo” la carpeta donde está el archivo. Observe que las diagonales se escriben de forma invertida en el código. Otro detalle es, que, debido a los escudos de protección implementados por las nuevas versiones de Microsoft Windows, la carpeta de destino no debe contenerse en una unidad de sistema (típicamente “C”), sugerimos en su lugar usar una memoria flash que haga las veces de unidad de destino.

Ejemplo 5.5. Almacenando texto en un archivo de salida.

```
#include <cstdlib>
#include <iostream>
```



“HACIA LA EXCELENCIA, CON
CALIDAD HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



CACEI



CACEB A.C.

A COMEXA



ITSZ

ESTUDIANTES DE
GRADO EN
SISTEMAS COMPUTACIONALES



```
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Creacion de archivos de salida\n";
    ofstream archivo;
    archivo.open("D:/Trabajo/myfile.txt");
    archivo<<"Hola desde este archivo\n";
    archivo<<"Primer texto complementado\n";
    archivo.close();
    cout<<"\nPrograma terminado\n\n";
    return 0;
}
```

También se puede asociar el archivo en cuestión directamente en la declaración, de la siguiente manera:

```
ofstream archivo("miarchivo.txt");
```

Tanto la primera como la segunda forma admiten un segundo argumento que especifica el modo de apertura de un archivo. Estos modos de apertura se encuentran declarados en la librería iostream. La tabla siguiente enlista los modos disponibles.

ios::app	Se escribe al final del archivo
ios::out	El archivo se abre para escritura
ios::trunc	Si el archivo existe se elimina su contenido
ios::in	El archivo se abre para lectura, no se modifica el archivo original.
ios::binary	El archivo se abre en modo binario

Si deseásemos agregar más texto al archivo creado en el ejemplo 5.5, el código correspondiente sería el que sigue.

Ejemplo 5.6. Agregando texto al programa del ejemplo 5.5.

```
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Agregando texto a myfile.txt\n";
    ofstream archivo("D:/Trabajo/myfile.txt", ios::app);
```





```

archivo<<"Nuevo texto agregado\n";
archivo<<"Archivo modificado\n";
archivo.close();
cout<<"\nPrograma terminado\n\n";
return 0;
}

```

62

El método para abrir un archivo en modo lectura es muy similar, pero en este caso utilizaremos **ifstream**.

Para tener el control del fichero, aparte de conocer los modos de apertura de un archivo, debemos de conocer el delimitador. En el caso de los archivos el delimitador es el “fin de archivo” (EOF, End Of File).

Ejemplo 5.7. Programa que permite visualizar el contenido de un archivo de texto.

```

#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char *argv[])
{
    cout<<"Leyendo texto de myfile.txt\n";
    cout<<"myfile contiene actualmente: \n\n";
    char caracter;
    ifstream archivo("D:/Trabajo/myfile.txt", ios::in);
    while (!archivo.eof())
    {
        archivo.get(caracter);
        cout<<caracter;
    }
    archivo.close();
    cout<<"\n\n*** Programa terminado ***\n\n";
    return 0;
}

```

Un problema del código del ejemplo 5.7, es que si el archivo a mostrar no existiese la computadora entraría en un ciclo infinito (¿por qué?). Para controlar este particular podemos incluir una condición de verdad.

Ejemplo 5.8. Modificación del programa del ejemplo 5.8, para verificar la existencia del archivo de apertura.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```

cout<<"Programa que lee texto del archivo myfile2.txt, siempre que exista\n\n";
char caracter;
ifstream archivo("myfile2.txt", ios::in);
if (archivo)
{
    cout<<"myfile2 contiene actualmente: \n\n";
    while (!archivo.eof())
    {
        archivo.get(caracter);
        cout<<caracter;
    }
    archivo.close();
}
else
{
    cerr<<"Imposible mostrar, el archivo no existe\n"<

```

63

Ejercicios 5.4. Desarrollar:

- i. Un programa que incluya un menú con operaciones básicas de matrices: A+B, A-B, B-A, A*B, B*A, y cuyos resultados se almacenen opcionalmente en un archivo.
- ii. Un programa que lea nombre completo de un listado n de alumnos y las calificaciones de cuatro materias de cada uno, cada tipo de dato debe almacenarse en su propio archivo.
- iii. Un programa que despliegue la información introducida por el programa del ejercicio anterior.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

6. Módulos

En esta unidad el estudiante:

- Aplica funciones y procedimientos en programas.

64

6.1. Concepto de módulo

Un módulo es una porción de un programa que permite realizar una o varias de las tareas que éste debe realizar. Por lo general un módulo recibe como entrada la salida que a su vez proporciona otro módulo, o, en su defecto, las entradas directas al sistema, si es que se trata del módulo principal. Posteriormente el módulo proporcionará una salida que igualmente puede ser utilizada por otro módulo, o simplemente contribuir a la salida final de todo el programa.

Los módulos suelen estar organizados jerárquicamente, de tal forma que existe uno principal que es el encargado de llamar a aquellos que son de un nivel inferior.

Las características que debería cumplir un módulo son:

- **Tamaño relativamente pequeño.** Esta cualidad permite aislar el impacto que pueda tener la realización de modificaciones en un programa, sea para corregir un error, o por un rediseño del algoritmo correspondiente.
- **Independencia modular.** Implica que para desarrollar un módulo no es necesario conocer los detalles internos de los demás módulos. Esto a su vez permite:
 - Características de caja negra.
 - Aislamiento de los detalles por encapsulamiento.

Programar mediante módulos acelera la programación al poder realizarse en equipo y en paralelo, además de que contribuye a la reutilización de software.

6.2. Funciones

Una función es un bloque de instrucciones al que se le asigna un nombre. La característica principal de una función es que sólo puede devolver un solo valor. De hecho en la biblioteca **cmath** se encuentran funciones que hemos usado `sin()`, `cos()`, `pow()`. En estos casos sólo nos ocupó la manera en que se usan, dándole datos (argumentos) y devolviéndonos valores (salidas), sin preocuparnos de la manera en que estaban construidas.

Al hablar de funciones debemos de hablar de prototipos. El prototipo de una función es el esquema estándar bajo el cual opera dicha función, es como su formulario. Por obvias razones la función debe estar definida o al menos declarada antes de poder hacer uso de ella.

El prototipo de la función se realiza abajo del espacio de nombres con la siguiente sintaxis:

`Tipo_Dato Nomb_Func(Tipo_Dato Arg1, Tipo_Dato Arg2,..., Tipo_Dato ArgN);`

La definición (bloque de instrucciones) de la función se realiza después de la función principal `main`, con la siguiente sintaxis:



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Tipo_Dato Nomb_Func(Tipo_Dato Arg1, Tipo_Dato Arg2,..., Tipo_Dato ArgN)
 {
 Instrucciones;
 }

65

Ejemplo 6.1. Programa que suma dos valores mediante una llamada a función.

```
#include <cstdlib>
#include <iostream>

using namespace std;
int suma(int x, int y); //Prototipo de la función suma
int main(int argc, char *argv[])
{
    cout<<"Ejemplo de uso de funciones\n";
    cout<<"Cálculo de la suma de dos funciones \n\n";
    int a, b;
    cout<<"Introduzca un valor numérico, a= ";
    cin>>a;
    cout<<"\nIntroduzca otro valor numérico, b= ";
    cin>>b;
    cout<<"\n Resultado: a+b= "<<suma(a,b);
    cout<<"\n*** Programa terminado ***\n";
    return 0;
}
int suma(int x,int y) //Definición de la función
{
    return x+y;
}
```

En el ejemplo 6.1 observe como se invoca a la función “suma” desde la función principal, al tiempo que se le pasan los argumentos necesarios. La palabra reservada **return** devuelve el valor calculado en la función, termina con la ejecución de éste y devuelve el control a la función principal en el espacio siguiente en el que fue invocada. También se puede invocar a la función de tal forma que almacene el valor resultante en una variable contenedora:

variable=función(arg1,arg2,...,argN);

Cuando se pasan datos (argumentos), éstos deben cumplir las características declaradas en la definición de las funciones. Note que al declarar y definir la función los parámetros no necesariamente deben llamarse igual, sin embargo, sí deben ser del mismo tipo.

Una función puede regresar cualquier tipo de valor excepto tablas u otras funciones.

Ejercicios 6.1. Para practicar la creación de funciones:

- Realice un programa que dados los datos cartesianos de dos vectores de n-ésimo



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



- orden se realice el producto punto de los dos, usando para realizar el producto entre elementos una función.
- ii. Realice un programa que calcule el factorial de un número mediante una función.

6.3. Procedimientos

Los procedimientos se parecen a las funciones en el sentido de que también realizan tareas repetitivas, sin embargo, son más versátiles dado que verdaderamente representan miniprogramas dentro de un programa. Otra diferencia es que los procedimientos no devuelven valores (no hacen uso del **return**) sino que procesan los argumentos que se le entregan y los resultados se pasan a variables de destino previamente creadas.

Sintaxis de declaración (después del espacio de nombres):

```
void nombr_proc(tipo_dato arg1, tipo_dato arg2,...,tipo_dato argN);
```

Sintaxis de definición (después de función main):

```
void nombr_proc(tipo_dato arg1, tipo_dato arg2,...,tipo_dato argN)
{
    Instrucciones;
}
```

Cabe aclarar que en este caso los datos procesados quedan guardados en los argumentos declarados para tal fin en el programa principal al invocar el procedimiento.

Ejemplo 6.2. Programa que calcula el producto de un escalar por un vector de orden n.

```
using namespace std;
void lee_vec(int arreglox[], int orden);
void esc_vec(int arregloc[], int orden);
int main(int argc, char *argv[])
{
    int n,a;
    cout<<"Programa que realiza el producto de un vector por un escalar \n\n"<<endl;
    cout<<"Introduzca el orden (n) del vector= ";
    cin>>n;
    int vectorA[n];
    cout<<"\n***** Introduzca los valores del vector *****\n";
    lee_vec(vectorA,n);
    cout<<"\n\nIntroduzca el valor del escalar a= ";
    cin>>a;
    for (int i=0;i<n;i++)
    {
        vectorA[i]=vectorA[i]*a;
    }
}
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```

cout<<"\n***** Los nuevos valores del vector son *****\n";
esc_vec(vectorA,n);
system("PAUSE");
return EXIT_SUCCESS;
}
void lee_vec(int arreglox[],int orden)
{
    for (int i=0;i<orden;i++)
    {
        cout<<"\nvector["<<i+1<<"]= ";
        cin>>arreglox[i];
    }
}
void esc_vec(int arregloy[],int orden)
{
    for (int i=0;i<orden;i++)
    {
        cout<<"vector["<<i+1<<"]= "<<arregloy[i]<<endl;
    }
}

```

Ejercicios 6.2. Modifique el programa que realiza opcionalmente la suma, producto, multiplicación y determinante de matrices, para que use procedimientos para la lectura y escritura de matrices. Programe también procedimientos para realizar los cálculos.

6.4. Macros

Se consideran instrucciones de pre-proceso ya que se ejecutan al comienzo de la compilación. Se declaran sobre el espacio de nombres. Justo como si fueran una de las bibliotecas del lenguaje:

```
#define Nombre_Macro(parámetros) instrucciones
```

Es importante recalcar que las macros ocupan mucha más memoria que la de las funciones comunes, dado que reflejan un nivel de proceso “en tiempo real”, por lo que nuestros programas serán más sensibles a los fallos.

Ejemplo 6.3. Macro que prueba si un número es mayor que otro, mostrando el que es mayor.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```
#include <cstdlib>
#include <iostream>
#define mayor(a,b) (a>b)? a : b
using namespace std;

int main(int argc, char *argv[])
{
    int a,b;
    cout<<"Programa ejemplo de uso de macros\n\n";
    cout<<"Se comparan dos números\n";
    cout<<"Teclee el primer número: ";
    cin>>a;
    cout<<"\nTeclee un segundo número (diferente): ";
    cin>>b;
    cout<<"\nEl mayor de esos numeros es: "<<(mayor(a,b))<<endl<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Podemos ver que las macros al igual que las funciones pueden tener parámetros, sin embargo, tienen que escribirse con cuidado. Por ejemplo, al hacer una macro de la siguiente manera:

```
#define curva(a) 1+2*a+a*a 4
```

Si en nuestro código colocamos una instrucción curva(1+3) se sustituye por

```
1+2*1+3+1+3*1+3
```

Contrario a lo que esperaríamos si invocáramos curva(4), obtendríamos 13 en lugar de 25. Habría que definir la macro como $1+2*(a)+(a)*(a)$ para que los resultados fueran coincidentes..

El uso de macros hace más difícil la depuración de un programa, porque en caso de haber errores, el compilador nos avisará que hay errores, pero sólo en la implementación de la macro, sin realizar comprobaciones de tipo de datos ni conversiones.

6.5. Recursividad

La recursividad se presenta cuando una función se invoca a sí misma. A diferencia de las iteraciones (bucles), las funciones recursivas consumen muchos recursos de memoria y tiempo.

Una función recursiva se programa simplemente para resolver los casos más sencillos, cuando se llama a una función con un caso más complicado, se divide el problema en dos partes, la parte que se resuelve inmediatamente y la que necesita de más pasos, ésta última se manda de nuevo a la función, que a su vez la divide de nuevo, y así sucesivamente hasta que se llegue al caso base. Cuando se llega al final de la serie de llamadas, va recorriendo el camino de regreso, hasta que, por fin, presenta el resultado.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx





Ejemplo 6.4. Programa que calcula el factorial de un número mediante una función recursiva.

```
using namespace std;
long factorial(long numero);
int main(int argc, char *argv[])
{
    long numero;
    cout<<"Programa ejemplo de la recursividad";
    cout<<"\n\nCalculo del factorial";
    cout<<"\nProporcione un numero entero positivo: ";
    cin>>numero;
    cout<<"\n\nEl factorial es: "<<factorial(numero)<<endl<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
long factorial(long numero)
{
    if (numero<=1)
    {
        return 1;
    }
    else
    {
        return numero*factorial(numero-1);
    }
}
```

69

Consideramos importante hacer notar que la recursividad suele consumir más memoria de la necesaria, por lo que es peligroso abusar de este tipo de programación.

Ejercicios 6.3. Use funciones recursivas para obtener:

- La suma de los primeros n números enteros positivos pares.
- La suma de los cuadrados de los primeros n números enteros positivos.
- La suma de los primeros n términos de:

$$S = \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \cdots + \frac{1}{2^{n-1}}$$



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

7. Graficación¹

En esta unidad el estudiante:

- Utiliza herramientas básicas de graficación.

70

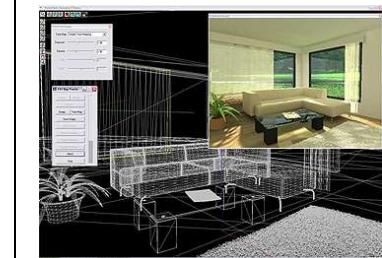
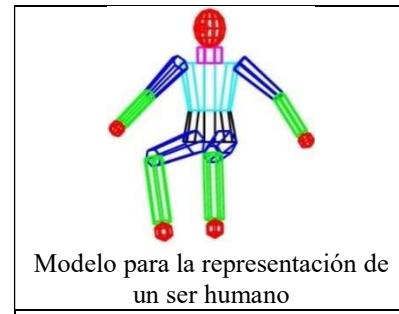
7.1. Fundamentos

Comprenderemos a la graficación por computadora, como el uso que se le da a ésta última en la construcción, modificación, despliegue y manipulación de modelos geométricos e imágenes de objetos.

El problema a resolver en la graficación por computadora es que, dado un objeto, sea posible generar un modelo geométrico del mismo para obtener una imagen representativa.

Sus áreas de estudio incluyen:

- **Modelado:** Evidentemente por modelos nos referimos a aquellas expresiones matemáticas que nos permitan reproducir la forma de los objetos que forman una escena. Este es el caso de estudio de las empresas de software que desarrollan programas para diseñadores como pueden ser: arquitectos, ingenieros civiles, etc.
- **Render:** Es un proceso en el que, mediante cálculos y algoritmos computacionales, se busca dibujar de manera realista un objeto tridimensional, aplicando color, texturas y efectos de iluminación. En particular podemos relacionarlo con algunas de las opciones que incluyen programas como Autodesk AutoCAD o D'Assault SolidWorks para mejorar la apariencia de las entidades tridimensionales que se diseñan en ellos.

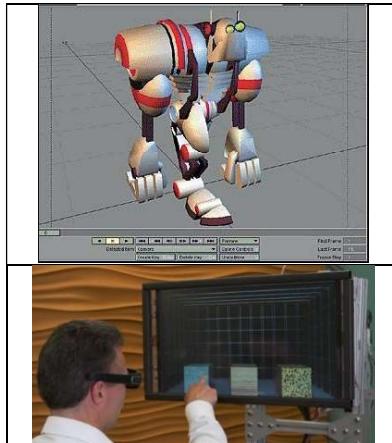


¹ Tema 1. Elementos de las gráficas por computadoras. (2010, Abril 12). Recuperado Octubre 27, 2013, desde OpenCourseWare de la Universidad Anáhuac México Norte Web site:
http://educommons.anahuac.mx:8080/eduCommons/computacion-y-sistemas/programacion-de-graficas-computacionales/tema-1-1/aprendizaje_motivacion



Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

- **Animación:** Es el control por computadora que se realiza sobre una secuencia de imágenes u objetos a lo largo de un intervalo del tiempo.



71

- **Interacción:** Se refiere a la creación de las interfaces de usuario, lo cual se debe realizar mediante un estudio en Hardware y Software.

Hoy en día es evidente que las gráficas por computadora son usadas en una amplia variedad de aplicaciones: científicas, arquitectónicas, artísticas, cinematográficas, etc. Esto se debe a:

- La visualización de datos complejos de forma sencilla.
- La interacción en tiempo real.
- Una nueva forma de desarrollar y un nuevo enfoque para la resolución de problemas.
- El Hardware y el Software de nuestros días es más accesible.

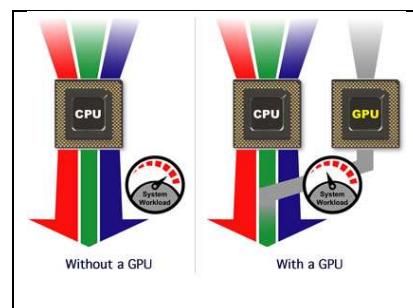
El Diseño Asistido por Computadora (CAD) es una consecuencia del estudio de los gráficos computacionales.

7.2. Elementos de un ambiente gráfico

Cuando se trata de realizar gráficos en una computadora se ven involucrados las partes formales de la computadora: el Hardware y el Software. El hardware porque es la que se encargará del proceso pesado, consistente en cálculos matemáticos e interpretación de los códigos necesarios. El software porque la eficiencia de las aplicaciones permitirá una gestión eficiente de los recursos de la computadora permitiendo un mayor o menor éxito en el cumplimiento de los objetivos.

Con respecto al hardware las partes que se ven más comprometidas en la tarea de graficación son:

- Adaptador gráfico, o, más precisamente el chip o placa dedicada específicamente a los gráficos, generalmente llamado GPU (Graphics Processing Unit).
- Monitor, porque dependiendo tanto de su velocidad de barrido como de su matriz de pixeles permite una mayor o menor calidad de la representación gráfica.
- Dispositivos de entrada, los medios involucrados en la representación gráfica externa a la computadora.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

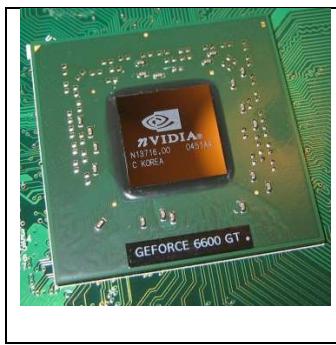
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Los conceptos relacionados con el software son:

- La Interfaz Gráfica de Usuario (GUI, Graphical User Interface), que es el conjunto de imágenes y objetos gráficos usados en la interacción usuario-sistema operativo-máquina, para representar la información y las acciones disponibles.
- La Interfaz de Programación de Aplicaciones (API, Application Programming Interface) es el conjunto de funciones y procedimientos que cierta biblioteca ofrece para ser utilizado por otro software, dicho de otro modo, supone la capacidad de comunicación entre distintos componentes de software, entre los que se establece una serie de llamadas a bibliotecas.
- Aplicaciones, que se refiere al programa como tal que se encargará de la gestión de los demás componentes para la realización de los gráficos.

7.2.1. Adaptador gráfico

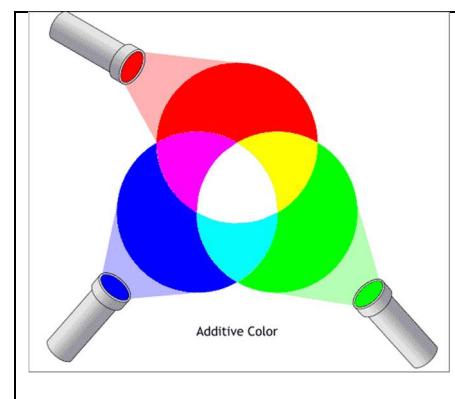
Las tarjetas o chips gráficos nacieron de la necesidad de liberar al microprocesador del uso de memoria RAM y de los cálculos relacionados con la representación visual de objetos, sean estos de 2D o de 3D (aunque recientemente un fabricante (AMD) ha hecho una simbiosis de un integrado gráfico con un microprocesador, aduciendo una mejora en la eficiencia del intercambio de información entre ambos componentes).



7.2.1.1. Modelos de color

Un aspecto relacionado con el adaptador gráfico, es la representación del color, en los que se puede hablar de dos **modelos**, uno basado en la fenomenología óptica (luz) y otra basada en sustancias químicas (pigmentos). El primero se usa en la tecnología de monitores y televisores y el segundo con aquello que se relacione con la impresión.

Para los que se basan en luz tenemos:

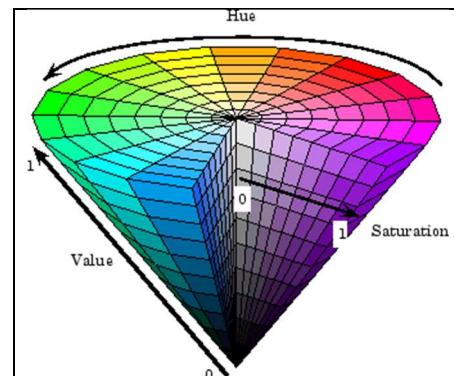


- El modelo de color **RGB** es un sistema de codificación que establece variaciones aditivas de los tres colores básicos que utiliza: Red-Rojo, Green-Verde, Blue-Azul, para producir los restantes.



Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

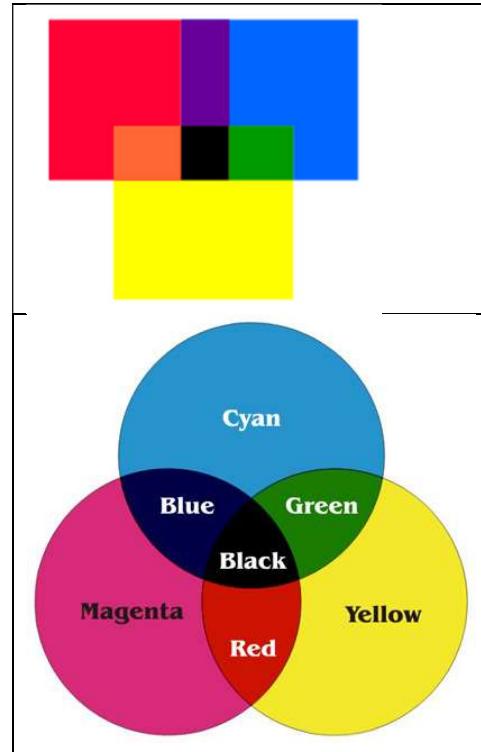
- El HSV (Hue-Matiz, Saturation-Saturación, Value-Valor). Es una transformación no lineal del espacio de color RGB, y se usa como progresión de color. Se le puede representar por un cono de colores invertido, en el que la cara circular representa la elección de un matiz, una línea desde el centro sobre la cara representa el nivel de saturación, y la altura (invertida del cono) representa el valor del color.



73

Los modelos basados en pigmentos son:

- RYB (Red-Rojo, Yellow-Amarillo, Blue-Azul), es un modelo de síntesis sustractiva (al contrario de la RGB, que es aditiva). Con la evolución fotografía se ha demostrado que es un sistema impreciso, por lo que no suele usarse en aplicaciones informáticas, aunque se le sigue utilizando en la teoría impartida en la educación artística.
- CMYK (Cyan-cian, Magenta, Yellow-Amarillo, blacK-Negro) es una corrección del modelo RYB, que se considera tecnológicamente obsoleto. Es de tipo sustractivo, basándose en la absorción de la luz: el color que presenta un objeto corresponde a la parte de la luz que incide sobre éste y que no es absorbida. El cian se considera un filtro del color rojo, puesto que absorbe este color, lo mismo pasa al magenta respecto del verde, y al amarillo respecto del azul.



Relacionada con los modelos, tenemos a la **profundidad de color** que se refiere a la cantidad de bits que se utilizan para codificar cada variante, cada n bits se tienen 2^n variantes de color. En particular el código RGB utiliza 8 bits.

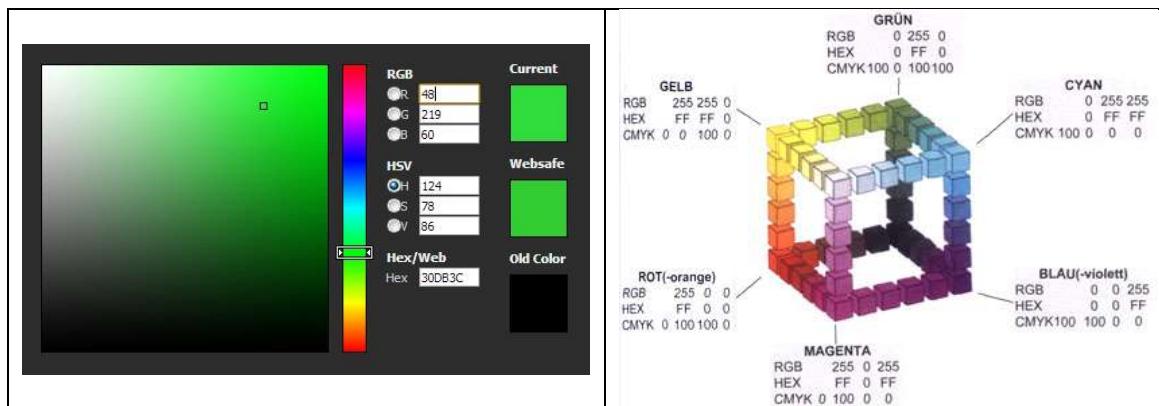
Profundidad	Número de colores
1	2
2	4
4	16
8	256
16	65,536
24	16,777,216
32	4,294,967,296



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

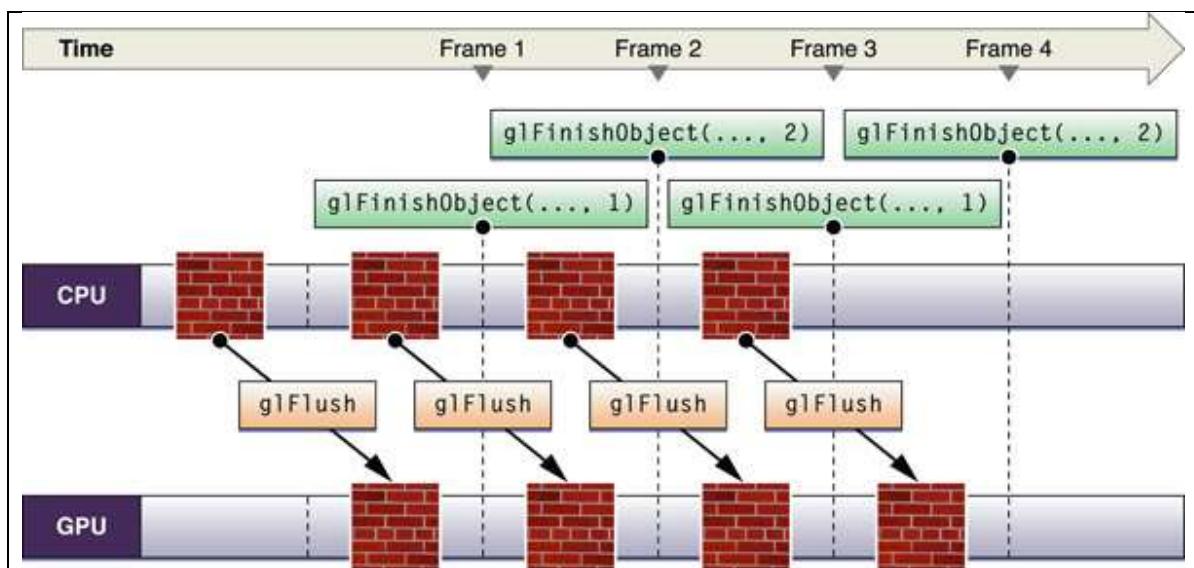
Los modelos de color son aplicados regularmente de manera sencilla mediante software de diseño gráfico, no así cuando se involucra un tratamiento más orientado a la programación en el que se hace necesaria una interpretación adecuada de los códigos usados según el lenguaje de programación.



74

7.2.1.2. El Framebuffer y el monitor

El framebuffer (o gestor de marco) representa la fracción de memoria que corresponde a la gestión de cada uno de los pixeles de un monitor. Esta fracción de memoria puede ser de tipo física o virtual; física es cuando el sistema cuenta con un dispositivo específico de memoria alterna al de la RAM; virtual es cuando el sistema operativo gestiona una fracción de la RAM para acceder a los dispositivos de representación gráfica como los monitores. Esto evidentemente puede acelerar o ralentizar el proceso de una computadora.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

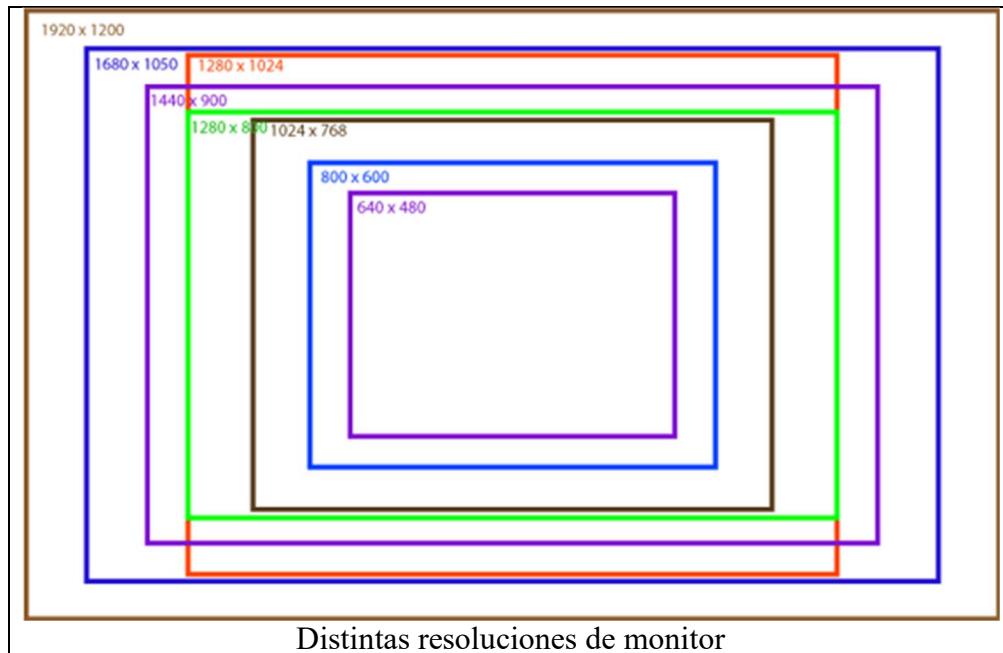
La aparición del framebuffer en la arquitectura de una computadora obedece a la necesidad de reconfigurar automáticamente el hardware para mostrar diferentes resoluciones, profundidades de color, diseños de memoria y tasas de refresco.

Así entonces se involucran otros tópicos llamados: Resolución (tamaño de la matriz de pixeles a usar), Memoria (Compartida o Integrada), Buffer Simple o Doble (también llamado múltiple), que se refiere al uso de más de un solo buffer para el almacenamiento de un bloque de datos.

En particular el buffer simple se usa cuando se despliegan imágenes estáticas, mientras que las de doble buffer se utilizan cuando se despliegan imágenes animadas. El siguiente gráfico ilustra el proceso mediante el cual se van refreshcando las imágenes en una pantalla, cada frame representa un evento en un instante de tiempo distinto, mientras que el CPU va procesando las órdenes para, primero, trazar el gráfico en su “lienzo” (buffer), y posteriormente, enviarlo al GPU para el despliegue de gráficos.

Ahora debe ser evidente que el framebuffer guarda una estrecha relación con el monitor, que es la salida de datos por defecto.

La amplia variedad de monitores que han existido desde su aparición como componente por defecto de un sistema computador (1970) se refleja en los avances que en cuanto a tecnología ha tenido la humanidad. En particular éste desarrollo se nota en la cantidad de pixeles que forman la matriz de la pantalla, el medio de desplegado (TRC, tubo de rayos catódicos; LCD, pantalla de cristal líquido; LED, matriz de diodos emisores de luz) y la frecuencia de refresco (que es el número de imágenes que se despliegan por minuto).



Para lo que nos ocupa los parámetros que debemos tomar en cuenta son

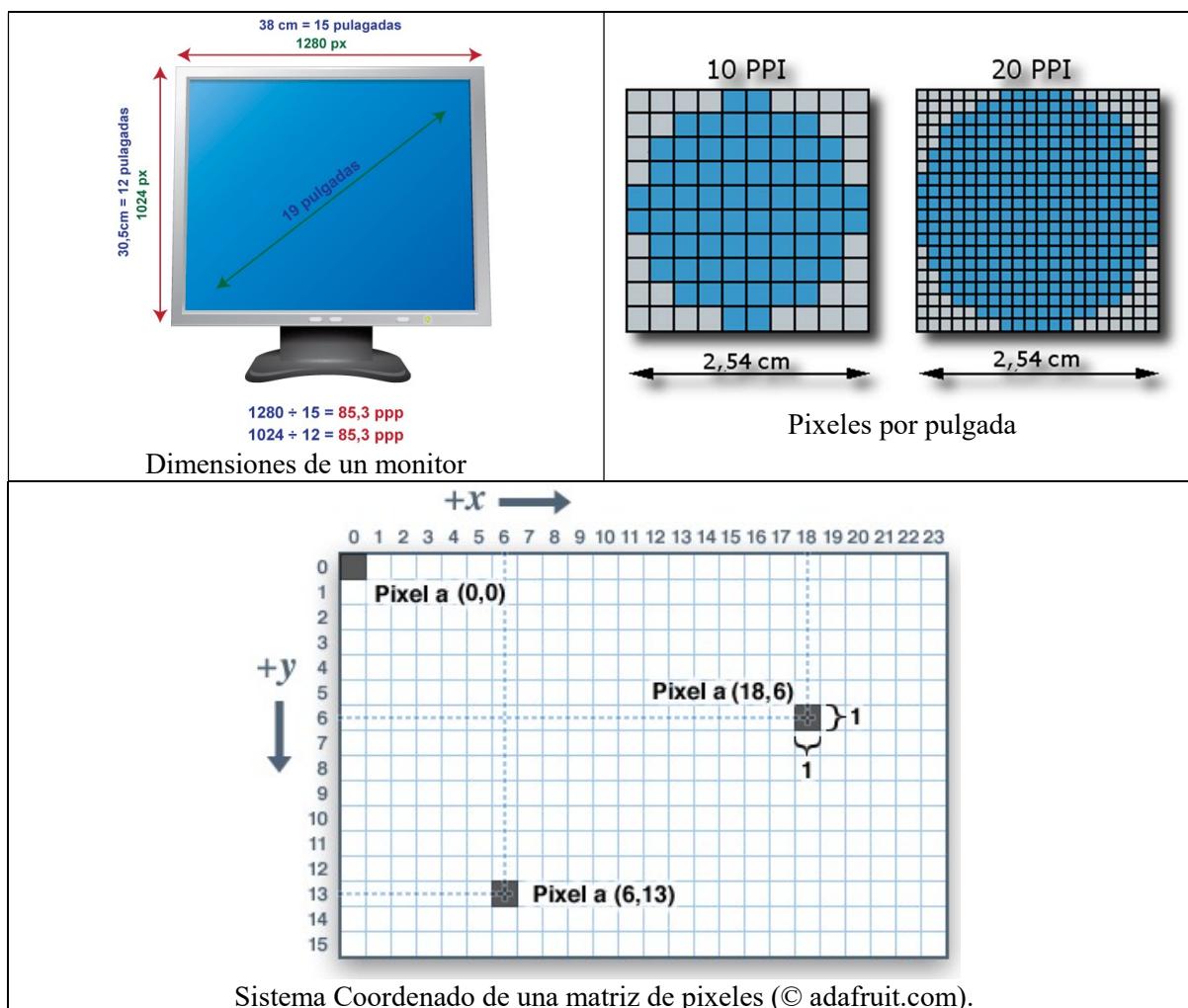


"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



- Las posibles resoluciones estándar disponibles, la cual hoy en día es bastante amplia: 800×600 , 1024×768 , 1280×1024 , etc.
- La dimensión de la pantalla, generalmente medida en pulgadas y en forma diagonal de una esquina a otra diagonalmente opuesta. Aquí existe un parámetro relevante cuando se trata de la edición de video, que es el de la relación de proporción horizontal-vertical, por ejemplo: 4:3, que significa por cada 4 pixeles horizontales existen 3 en forma vertical.
- El número de pixeles por pulgada (PPI-Pixels per inch).
- La distribución de la matriz de pixeles mediante un sistema coordenado.



7.2.2. Software: API's y GUI's

En la actualidad las instrucciones que damos a una computadora se entregan de manera indirecta, mediante el uso de interfaces gráficas (ventanas, íconos, botones, punteros, etc.)



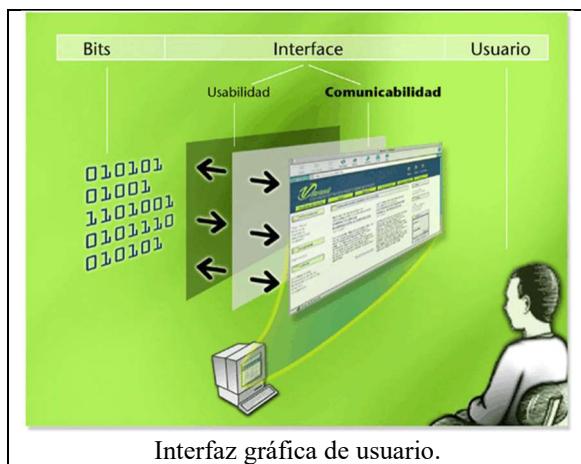
"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

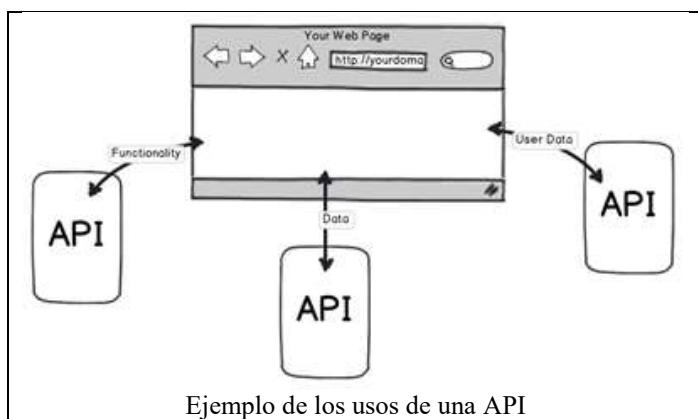


para facilitar la intercomunicación entre el usuario y la computadora. Surgen como una evolución de las interfaces en línea de comandos (consola) que se usaban para operar los primeros sistemas operativos. Como ejemplos de interfaz gráfica de usuario (GUI, por sus siglas en inglés), cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, Aqua.

Una GUI puede estar basada en estaciones de trabajo independientes, lo que implica el uso del kernel propio de la computadora, o bien ser sólo una subestación conectada a un servidor que se encarga de gestionar los servicios de interacción con un usuario.



Ahora bien, como ya se había dicho, el propósito de una API es la de proporcionar un conjunto de funciones de uso general (en este caso gráficas), que los programadores pueden aprovechar para evitar un trabajo más extenso. Pueden estar basados en una implementación basada en el sistema operativo, encargada de proporcionar las funcionalidades, para, por ejemplo, desplegar información en pantalla, borrarla, y refrescar la información desplegada. También se pueden implementar a través de una aplicación específica (usada por el sistema operativo) que ofrece otras tareas que pueden ser más o menos simplificadas, pero con diversos niveles de flexibilidad.



Como ejemplos de API's tenemos:

- OpenGL
- Direct3D
- Java 3D
- Open Inventor
- X3D (Extensible 3D)

Así entonces, para realizar graficación a nivel de software requerimos de: una GUI adecuada, elegir una API para llamar a las funciones, y acaso utilizar o desarrollar una aplicación que realice la gestión de la de tales funciones para desplegar información en el monitor a través de una GPU.

78

7.3. Librerías de OpenGL para graficación 2D

Para los objetivos que se persiguen en este curso, nos centraremos en las herramientas informáticas que nos permitan la graficación en dos dimensiones. De forma más precisa, estas gráficas se realizan a través de un conjunto de funciones que permiten la creación de entidades gráficas básicas llamadas *Primitivas 2D*, para lo cual debemos elegir una API adecuada, que en este caso es OpenGL.

La API denominada Open Graphics Library (OpenGL), es un conjunto de librerías escrito originalmente en C, para la producción de gráficos en 2D y 3D. Se le ha usado desde su creación en 1992 (por Silicon Graphics Inc.) en aplicaciones de CAD, realidad virtual, representación científica, visualización de información y simulación de vuelo.

Entre las librerías disponibles podemos citar las tres más utilizadas:

- **gl**, que define a la librería principal, contiene funciones para la creación de primitivas 2D (puntos, líneas, polígonos), también permite controlar los colores, además de manejar transformaciones geométricas. Debe incluirse en la cabecera de librerías mediante:

#include <GL/gl.h>

- **glu**, contiene funciones de más alto nivel que permiten la creación de primitivas 3D (cilindros, discos). También permite definir curvas y superficies NURBS (Non-Uniform Rational B-Splines, B-splines racionales no uniformes), y el manejo de texturas. Para incluirlo en la cabecera deberemos usar:

#include <GL/glu.h>

- **glut**, contiene funciones utilitarias para el manejo de ventanas y su interacción por medio de teclado y ratón. Para incluirlo en la cabecera usar:

#include <GL/glut.h>

Los gráficos se consiguen mediante la creación de algoritmos que hacen un uso extensivo de las bibliotecas comentadas. En nuestro caso nos centraremos en el uso de las bibliotecas *gl* y *glu*.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

7.4. El IDE y la API OpenGL

Debido a la amplia variedad de entornos de desarrollo para el estándar del C++, resulta imposible describir en cada una, la manera de instalar la biblioteca OpenGL. Otro inconveniente de la instalación lo presenta el hecho de que los sistemas operativos de Microsoft han complicado este fin al establecer mayores restricciones de usuario, por lo que resulta en una acción que debe hacerse con cuidado, de preferencia con la guía de un instructor.

Ésta sección describe brevemente la instalación de OpenGL para el compilador MinGW que acompaña al IDE DevC++ y su evolución wx-DevC++ sobre sistemas operativos Microsoft Windows, está basado en las ideas colocadas en el artículo de internet “*How to Install Dev-C++ and the GLUT Libraries for Compiling OpenGL Programs with ANSI C*”².

Las imágenes son tomadas del sitio web y hacen referencia al sistema operativo Windows XP, aunque las instrucciones son igualmente válidas para las versiones 7 y 8, en los que la única diferencia es el lugar donde se ubica la carpeta del IDE que entonces era simplemente *Archivos de Programa*, pero con la aparición de los sistemas operativos de 64 bits esta carpeta se llama ahora *Archivos de Programa (x86)*.

En el mismo sitio se ofrece un link de descarga³ para obtener una versión funcional de la API basada en la extensión GLUT.

Una vez descargado observará que el archivo es un comprimido en formato zip, al descomprimirlo en una carpeta localizable (aquí *temp*) verá que a su vez contiene una carpeta llamada GLUTMingw32, dentro hallará otras carpetas donde estarán contenidos los archivos necesarios, que solamente son tres.



Primero deberá buscar el archivo **glut.h** en *temp\glutming\GLUTMingw32\include\GL* y lo copiará en la carpeta *include\GL* que se encuentra ubicada donde se instaló Dev-Cpp o wxDevCpp, que probablemente sea: C:\Archivos de programa(x86)\Dev-Cpp\include\GL

² No hay ninguna fuente en el documento actual.

³ <http://chortle.ccsu.edu/Bloodshed/glutming.zip>



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



GL

File Edit View Favorites Tools Help

Address C:\DD\temp\GLUTMingw32\include\GL

Folders	Name	Size	Date Modified
temp	fgl.h	61 KB	2/8/1999 1:11 PM
Assembly Lang	fglu.h	8 KB	2/8/1999 1:11 PM
GLUTMingw32	fglut.h	11 KB	2/8/1999 1:11 PM
include	glcman.h	6 KB	9/25/2000 4:03 PM
GL	glut.h	27 KB	3/4/2001 10:33 AM
mui	glutfr90.n	5 KB	2/16/1999 10:55 AM
lib	tube.h	9 KB	2/8/1999 1:11 PM

Lugar de donde se copia a glut.h

Equipo > OS (C:) > Archivos de programa (x86) > Dev-Cpp > include > GL

Organizar Abrir Grabar Nueva carpeta

Favoritos	Nombre	Fecha de modificación	Tipo	Tamaño
Descargas	freeglut.h	09/05/2013 03:16 a...	C Header File	1 KB
Escritorio	freeglut_ext.h	09/05/2013 03:16 a...	C Header File	9 KB
Shared Space	freeglut_std.h	09/05/2013 03:16 a...	C Header File	26 KB
Sitios recientes	glut.h	09/05/2013 03:43 a...	C Header File	31 KB

Ubicación final del archivo glut.h

A continuación, localice y copie el archivo **libglut32.a** de la carpeta *temp\glutming\GLUTMingw32\lib*. Lo pegará en la carpeta lib ubicada dentro de Dev-Cpp, que probablemente será: C:\Archivos de programa(x86)\Dev-Cpp\lib. Es importante aclarar que cualquier archivo que existiese en las carpetas de destino debe de ser sobrescrito, aunque la versión del archivo en la carpeta de destino sea más reciente, porque de otro modo el compilador encontrará incongruencias que impedirán que los programas se ejecuten correctamente.

lib

File Edit View Favorites Tools Help

Address C:\temp\glutming\GLUTMingw32\lib

Folders	Name	Type	Date Modified
GLUTMingw32	libgle.a	A File	10/5/2000 10:29 PM
include	libglut.a	A File	10/5/2000 10:29 PM
GL	libglut32.a	A File	10/5/2000 9:53 PM
mui	libmurmur.a	A File	10/5/2000 10:29 PM
lib			

Localizando a libglut32.a





Equipo > OS (C:) > Archivos de programa (x86) > Dev-Cpp > lib >

Organizar	Abrir	Grabar	Nueva carpeta	
★ Favoritos	Nombre	Fecha de modifica...	Tipo	Tamaño
Descargas	libfreeglut.a	09/05/2013 03:16 a...	Archivo A	115 KB
Escritorio	libfreeglut_static.a	09/05/2013 03:16 a...	Archivo A	366 KB
Shared Space	libglut32.a	09/05/2013 03:43 a...	Archivo A	343 KB
Sitios recientes	libkbool.a			284 KB
	libtinyxml.a			91 KB
	...			

Pegando a libglut32.a en su destino final

El último archivo es **glut32.dll**, lo encontrará en *temp\glutming\GLUTMingw32*, deberá pegarlo en *C:\Windows\System32*.

Edgar Hernández G > Descargas > glutming > GLUTMingw32 >

Organizar	Abrir con...	Compartir con	Grabar	Nueva carpeta	
★ Favoritos	Nombre	Fecha de modifica...	Tipo	Tamaño	
Descargas	include	09/05/2013 03:46 a...	Carpetas de archivos		
Escritorio	lib	09/05/2013 03:46 a...	Carpetas de archivos		
Shared Space	glut32.dll	09/05/2013 03:46 a...	Extensión de la apl...	216 KB	
Sitios recientes	readme.txt	09/05/2013 03:46 a...	Documento de tex...	1 KB	

Copiando a glut32.dll

Equipo > OS (C:) > Windows > System32 >

Organizar	Abrir con...	Grabar	Nueva carpeta	
★ Favoritos	Nombre	Fecha de modifica...	Tipo	Tamaño
Descargas	glu32.dll	14/07/2009 03:40 a...	Extensión de la apl...	162 KB
Escritorio	glut32.dll	09/05/2013 03:46 a...	Extensión de la apl...	216 KB
Shared Space	gpapi.dll	14/07/2009 03:40 a...	Extensión de la apl...	95 KB

Colocando a glut32.dll

La carpeta System32 puede contener otras bibliotecas relacionadas con la API, que son **glu32.dll** y **opengl32.dll**, que generalmente son incluidos por defecto con el sistema operativo.

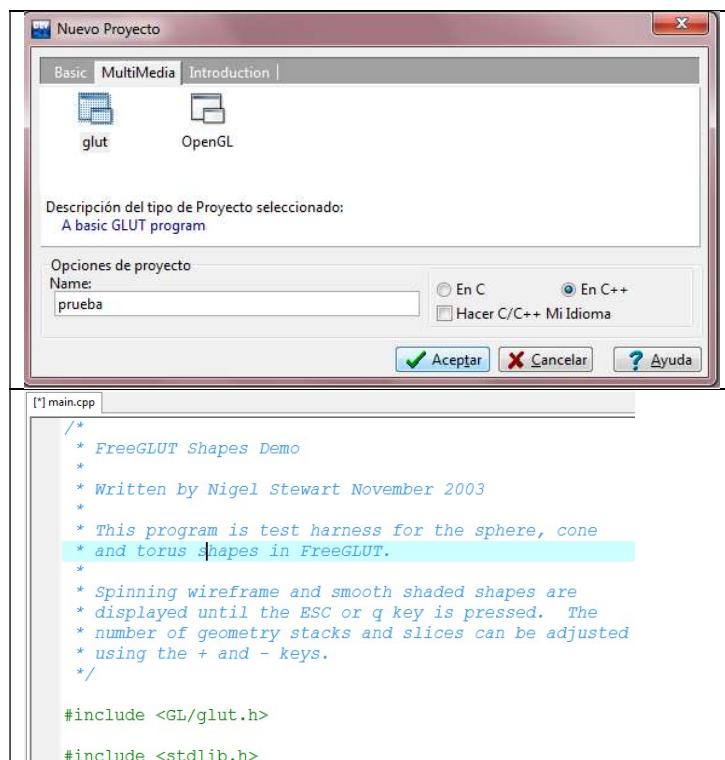
Una alternativa a los pasos aquí descritos es usar una extensión alternativa llamada **freeglut** que se actualiza periódicamente y posee versiones para 32 y 64 bits, se le supone compatible con **glut**, con algunas pequeñas diferencias. Puede hallar más información relativa en: <http://freeglut.sourceforge.net/>.



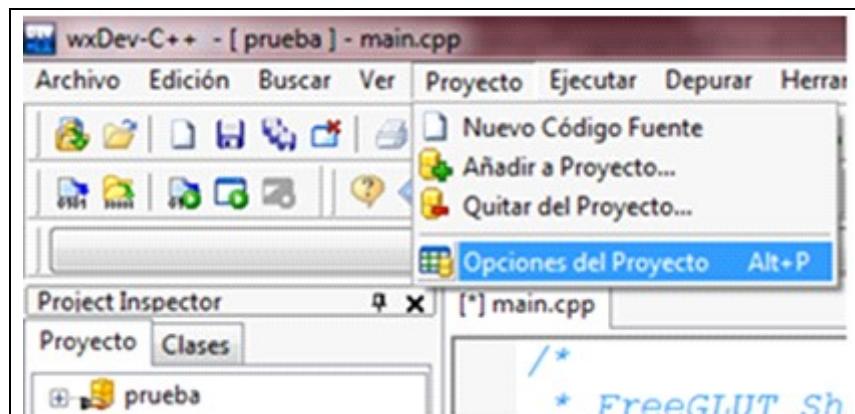


Un último paso es probar que el sistema es capaz de compilar y ejecutar códigos en OpenGL, para ello abra el IDE wxDev-C++ y cree un nuevo proyecto, cuando deba elegir qué tipo de proyecto, elija *Multimedia* y ahí seleccione *glut*. Luego guarde el archivo del proyecto como siempre.

El sistema precargará un código de ejemplo de la librería para el uso de la API.



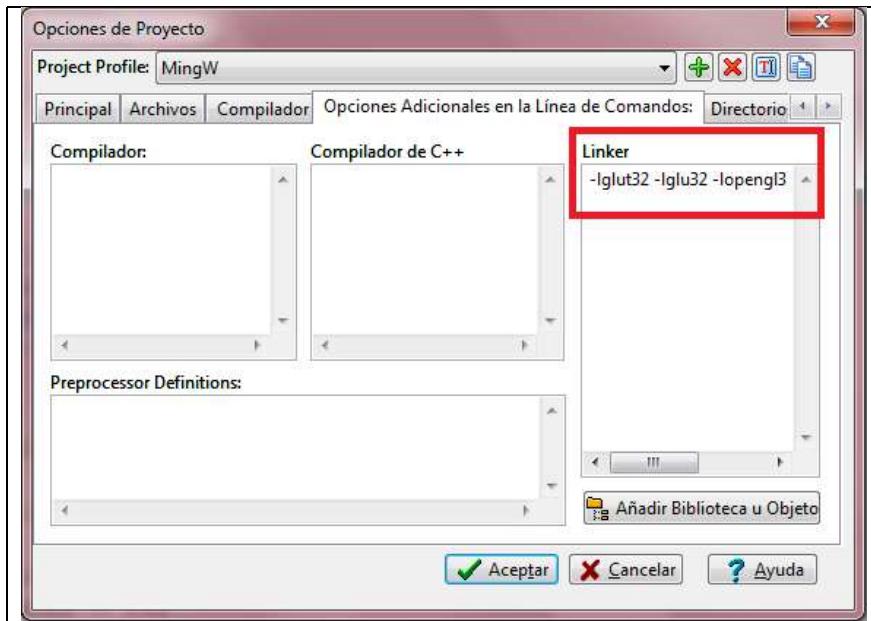
A continuación, deberá configurar el compilador para que sea capaz de comprender las llamadas a las funciones, para ello vaya a la barra de menús y active Proyecto>Opciones del Proyecto. Ahí vaya a la ficha “Opciones adicionales en la línea de comandos”, y en el recuadro ‘Linker’ debe revisar que esté escrito el siguiente texto: **-lglut32 -lglu32 -lopengl32 -lwinmm -lm -lgdi32**. Este paso deberá repetirlo para todo proyecto que utilice la API para hacer graficación.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

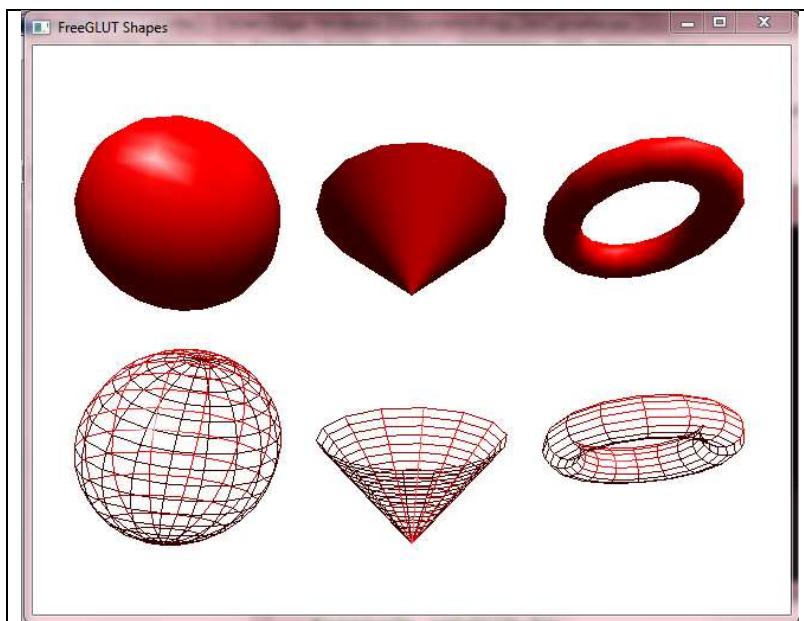
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx





83

De estar escrito, puede proceder a compilar el código, para posteriormente ejecutarlo, el resultado debería ser como se muestra en la siguiente imagen.



7.5. Generando una ventana

Si vamos a generar gráficos, éstos deben producirse en el marco de una ventana, cuya construcción depende en parte del sistema operativo en uso y de la API, esto a su vez puede hacerlo más o menos complejo. En este caso supondremos el uso del sistema operativo



"HACIA LA EXCELENCIA, CON
CALIDAD HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Windows y el IDE wxDev-C++, aunque más adelante presentaremos ejemplos en otro IDE (NetBeans) y corriendo sobre GNU/Linux.

Así, una especie de programa “Hola Mundo” para la graficación consiste en generar una ventana, cuyo código se muestra a continuación.

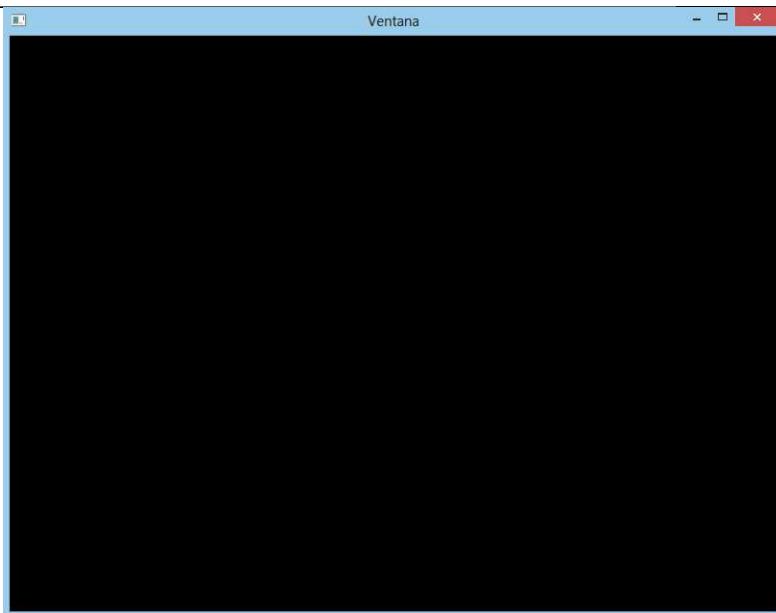
84

Ejemplo 7.1. Código para la generación de una ventana en C++ y OpenGL.

```
#include <GL/glut.h>

void dibuja(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("Ventana");
    glutDisplayFunc(dibuja);
    glutMainLoop();
    return 0;
}
```



Resultado de la ejecución

El código anterior presenta sólo instrucciones de inicialización de gráficos cuya descripción se presenta de manera breve a continuación:



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



- `#include <GL/glut.h>` carga el conjunto de librerías de la API GLUT⁴. GLUT ya incluye a `gl.h`, `glu.h` y `glx.h`.
- `glClear(parámetro)` es una función que “limpia” el buffer donde se va a dibujar. En este caso el parámetro predefinido (indexado) `GL_COLOR_BUFFER_BIT` le indica a glut que se debe limpiar con el color definido actualmente (en el ejemplo, inicialmente el color es el negro). Una alternativa es `glClearColor(máscara)` que permite limpiar el buffer con un tono específico de color, sobre éste se hablará más adelante.
- `glFlush()` es una función que obliga al hardware a “pintar” sin esperar a que otras instrucciones terminan de ejecutarse. Existe una función llamada `glFinish()` que prepara al sistema para dibujar pero ésta tarea sólo se hace cuando el sistema ha terminado otras ejecuciones.
- `glutInit(int *argc, char **argv)` inicializa GLUT y procesa cualquier argumento de la línea de comandos; `glutInit()` debe invocarse antes de cualquier rutina GLUT.
- `glutInitDisplayMode(unsigned int modo)` especifica si se usa un modelo RGBA o un color indexado. Puede especificarse si se quiere usar una ventana buffer simple o doble. Para nuestros fines bastará con el simple. Si trabaja en el modo de color indexado, querrá cargar ciertos colores dentro del mapeado de color; use `glutSetColor()` para hacerlo. Esta sentencia también puede indicar que la ventana tenga un fondo asociado, plantilla y/o acumulación de buffer. Por ejemplo, si desea una ventana con doble búfer, el modelo de color RGBA, y un búfer de fondo debe invocar `glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH)`.
- `glutInitWindowPosition(int x, int y)` especifica la localización en pantalla desde la esquina superior izquierda de la ventana gráfica.
- `glutInitWindowSize(int ancho, int alto)` especifica el tamaño en pixeles de la ventana.
- `int glutCreateWindow(char *string)` crea una ventana de contexto OpenGL. Se retorna un identificador único para la nueva ventana. Nota: la ventana sólo se despliega hasta que se invoca la función `glutMainLoop()`.

OpenGL con GLUT opera mediante llamadas a funciones que responden a la ocurrencia de eventos, como pueden ser: dar clic en un botón del mouse, presionar una tecla, etc., lo que en el argot de la programación es conocido como *callbacks*. Uno de esos callbacks es `glutDisplayFunc(void (*func)(void))`, ésta se encarga de invocar a funciones de trazado. Cada vez que GLUT determina que el contenido de la ventana necesita ser redesplegado, la función callback registrada por `glutDisplayFunc()` es ejecutada. Por lo tanto, todas nuestras rutinas para dibujar deben estar incluidas en un procedimiento que será invocado por este callback.

⁴ Nota: en Windows a veces se requiere declarar en el IDE la inclusión de la biblioteca `windows.h`, antes de declarar la biblioteca `glut.h`.

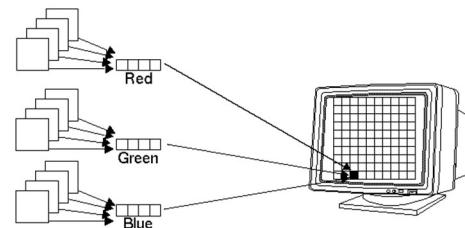


Si su programa cambia el contenido de la ventana, algunas veces tendrá que invocar a **glutPostRedisplay(void)**, el cual obliga a **glutMainLoop()** a llamar al siguiente desplegado callback registrado en la siguiente oportunidad.

7.6. Preparación del área de graficado

En OpenGL se pueden controlar cuatro clases de búfer: el buffer de color, que es usado normalmente para dibujar; el búfer de profundidad o *zbuffer*; el *stencil buffer*, usado para la generación de sombras, y el búfer de acumulación, que proporciona soporte para efectos especiales como el difuminado dinámico y la profundidad de campo. Como a nosotros sólo nos interesa la graficación, nos enfocaremos en algunas de las funciones que nos permite el buffer de color.

También debemos saber que una cierta cantidad de información de color se almacena directamente en cada pixel a través de lo que se llaman *Bitplanes*. En ellos existen dos modos de almacenamiento de los valores de color en el modelo RGBA, que pueden ser: en el propio bitplane como hardware o bien mediante valores indexados en una tabla de colores. En éste último caso, los valores para cada variación de color son de tipo flotante y van desde 0.0 hasta 1.0.



Esquema de un bitplane

Ahora bien, en una computadora la imagen desplegada no es eliminada de la memoria hasta que se ejecute una instrucción que renueve su contenido. Por esta razón si se desea generar una nueva imagen en un fondo limpio, entonces será necesario redefinir el color de fondo, ya sea el mismo que se estaba usando u otro diferente, lo que supone “limpiar” el contenido de la ventana. Para ésta labor en OpenGL se han implementado varias funciones de limpieza. De entre ellas tenemos a:

glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);	Establece un color de limpieza para el búfer de color.
glClear(GLbitfield mask);	Limpia el búfer especificado

Entonces para indicar el búfer tenemos:

- Buffer de color: **GL_COLOR_BUFFER_BIT**.
- Buffer de fondo: **GL_COLOR_BUFFER_DEPTH**.
- Buffer de acumulación: **GL_COLOR_BUFFER_ACCUM**.
- Buffer de patrón: **GL_COLOR_BUFFER_STENCIL**.

Por ejemplo, para establecer el color de limpieza en negro y ejecutar la acción los comandos serían:



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



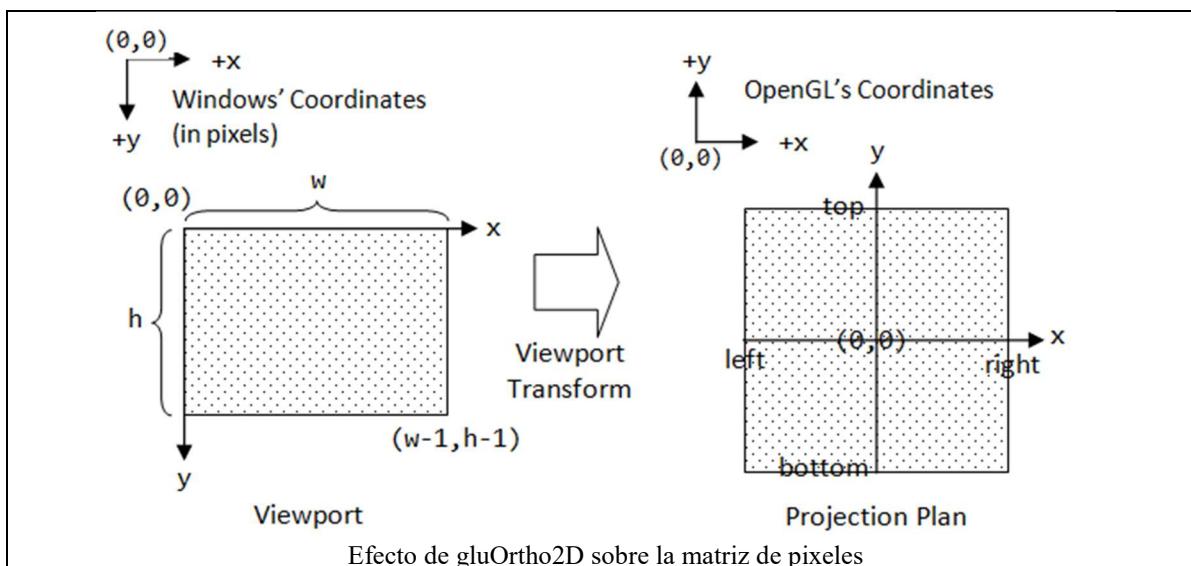
```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
```

glClear puede admitir más de un búfer si se aplica el símbolo relacional OR: “|”.

87

Para hacer trazos en la ventana creada se requiere que existan ubicaciones (posiciones) en ella, lo cual es posible dada la disposición matricial de los pixeles. Los comandos fundamentales para tal fin son:

- **void glMatrixMode(GLenum mode);** especifica la matriz actual para realizar la composición. En openGL las operaciones de rotación, translación, escalado, etc. se realizan a través de matrices de transformación. Dependiendo de lo que estemos tratando, hay tres tipos de matriz (que son las tres posibles señales o banderas (flags) que puede llevar de parámetro la función): matriz de proyección (GL_PROJECTION), matriz de modelo (GL_MODELVIEW) y matriz de textura (GL_TEXTURE). Con esta función indicamos a cuál de estas tres deben afectar las operaciones.
- **glLoadIdentity();** se carga como matriz actual la matriz identidad, lo cual equivale a resetear la matriz.
- **gluOrtho2D(izquierda, derecha, abajo, arriba);** permite establecer una ventana de dibujo 2D donde los parámetros indican las rectas de recorte. Al ejecutar se crea una matriz de proyección de coordenadas 2D y la multiplica por la matriz de proyección actual. Si no existen otras transformaciones acumuladas en la matriz de proyección actual, la nueva matriz transforma los objetos 2D de forma que los puntos (izq, abajo) y (der,arriba) aparezcan respectivamente en (-1,-1) y (1,1).



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



En estas condiciones podemos establecer la siguiente serie de pasos para preparar una ventana de graficación:

En primer término, incluir la librería glut (y, en su caso, la biblioteca *windows.h*)..

En segundo término, preparar un procedimiento de inicialización de lienzo donde se declare: el color de limpiado, el modo de uso de la matriz de pixeles, inicialización de la matriz de pixeles, el tamaño del área de dibujo dentro de la ventana. Opcionalmente: ordenar que se dibuje inmediatamente (*glFlush*). A continuación se deben realizar los siguientes pasos:

- I. Preparar un procedimiento que pinte y/o dibuje.
- II. Inicializar glut, se hace en la función principal main.
- III. Inicializar el modo de desplegado (en main).
- IV. Inicializar el tamaño de ventana (en main).
- V. Crear la ventana (en main).
- VI. Llamar consecutivamente al procedimiento de inicialización, y luego desplegar la función de pintura o dibujo.
- VII. Ciclar los callbacks de glut.

Ejemplo 7.2. Código para la inicialización de una ventana en color de fondo:

- El color elegido es un tono gris-azul claro.
- Se elige una matriz de proyección.
- Se utiliza todo el ancho de pixeles de la ventana creada para el sistema coordenado.
- Se pinta de manera inmediata mediante el procedimiento pinta.

```
#include <cstdlib>
#include <iostream>
#include <GL/glut.h>
using namespace std;
void inicializa(void)
{
    glClearColor(0.8,0.8,0.9,0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,500.0,0.0,500.0);
}
void pinta(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}
```



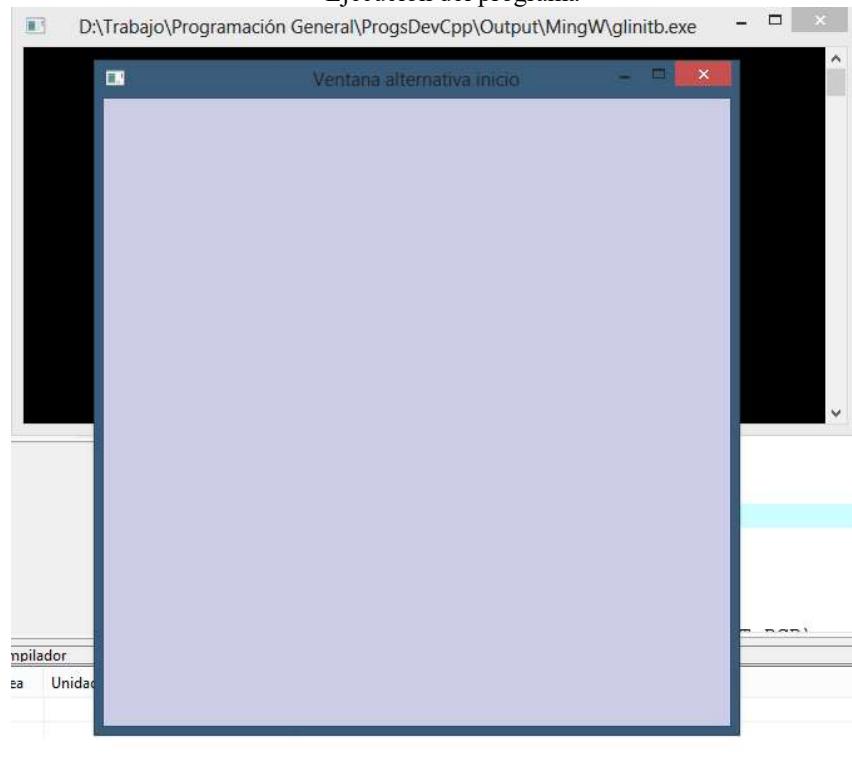


```

int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Ventana alternativa inicio");
    inicializa();
    glutDisplayFunc(pinta);
    glutMainLoop();
    return 0;
}

```

Ejecución del programa



7.7. Primitivas 2D

Llamamos primitivas 2D a aquel conjunto de callbacks que nos permiten trazar entidades básicas sobre nuestro lienzo. Las primitivas consisten de rutinas que permiten trazar puntos, líneas y polígonos. De hecho, en OpenGL se dibuja fundamentalmente con polígonos.

Toda primitiva se dibuja mediante **vértices**, que no necesariamente se refieren a esquinas, sino a puntos que se dibujan por coordenadas en la matriz de pixeles, o bien, son puntos de inicio y de llegada de aquellas primitivas que se dibujan mediante líneas. El área disponible para el trazado no necesariamente es toda la ventana creada, sino que está definida por las



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



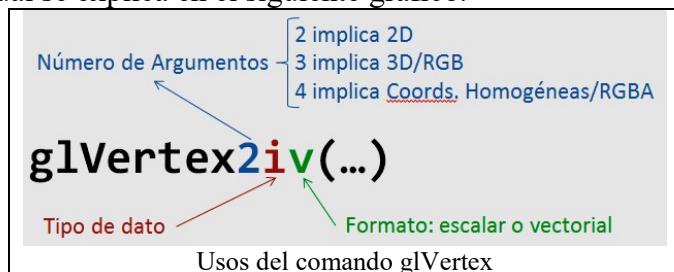
dimensiones declaradas en `glOrtho2D`, por lo que debemos de tener claro en qué área queremos trazar.

Para que un trazo se aloje en el buffer, es necesario llamar a las primitivas dentro de un par de sentencias que encierran el conjunto los vértices:

```
glBegin(GL_primitiva);      /*Inicio del proceso*/
    vértice1;
    vértice2;
    ...
glEnd();                  /*Fin del proceso*/
```

Se debe crear un juego de estas instrucciones si se van a trazar varias primitivas, uno por cada una.

Los vértices se declaran mediante el comando `glVertex` cuya escritura involucra diversos modos de uso, lo cual se explica en el siguiente gráfico.



Los tipos de datos disponibles para el comando son:

Letra	Tipo de Dato	Tipo en C	Definición en OpenGL
s	Entero de 16 bits	short	GLshort
i	Entero de 32 bits	int o long	GLint
f	Punto flotante de 32 bits	float	GLfloat
d	Punto flotante de 64 bits	double	GLdouble

Ejemplos de uso del comando son:

<code>glVertex2i(25,250);</code>	Declara un vértice de tipo 2D, con datos en formato entero, y ubicado de manera cartesiana en la matriz de pixeles con coordenadas en (25,250)
<code>int vectorp[]={32,43}; glBegin(GL_primitiva); ... glVertex2iv(vectorp); ... glEnd();</code>	Se declara por fuera del proceso de trazado un arreglo bidimensional llamado <i>vectorp</i> . A continuación, dentro del proceso es declarado un vértice en formato vectorial donde es usado como argumento el arreglo previamente creado.

Las primitivas tienen atributos que se “configuran” antes del proceso de trazado mediante instrucciones específicas. Algunas de estas características se presentan en la tabla siguiente.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Instrucción	Ejemplo
glColor: Establece el color para la primitiva. Si se trazan varias primitivas de distintos colores, entonces se debe establecer todas las veces necesarias antes del proceso de trazado de cada primitiva. Como se muestra en los ejemplos de la derecha tiene una estructura similar a la de la instrucción <code>glVertex</code> .	<code>glColor3f(0.5,0.6,0.7);</code> <code>glColor3fv(vector_color);</code>
glPointSize: Declara el tamaño de cada punto de trazado, por supuesto sólo aplica para primitivas que se trazan punto por punto. La sentencia tiene la estructura: <code>glPointSize(GLfloat tamaño);</code>	<code>glPointSize(3.0);</code>
glLineWidth: Aplicable a aquellas primitivas que se trazan mediante líneas, esta instrucción permite controlar el ancho de las mismas. La sentencia tiene la estructura: <code>glLineWidth(GLfloat ancho);</code>	<code>glLineWidth(2.0);</code>

7.7.1. Trazos por puntos

Para dibujar puntos en pantalla invocaremos a la función **GL_POINTS**. Para este callback cada vértice especificado corresponderá a un punto.

Ejemplo 7.3. Código ejemplo para el trazado de puntos. Es de notar la manera en que se puede ciclar el trazado para obtener múltiples puntos. Aquí se utiliza el IDE Code::Blocks.

```
#include <windows.h>
#include <GL/glut.h>

void inicializa(void)
{
    glClearColor(0.8,0.8,0.9,0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,500.0,0.0,500.0);
}

void dibuja(void)
{
    float negro[]={0.0,0.0,0.0};
    int vecpoint[]={375,375};
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3fv(negro);           // Se declara el color de la primitiva
    glPointSize(10);             // Se declara el tamaño del pincel
    glBegin(GL_POINTS);         // Se invoca el trazado de puntos
        glVertex2i(125,125);    // Se dibuja un punto por coordenada
        glVertex2iv(vecpoint);  // Se dibuja un punto por vector
        for(int i=0;i<=25;i++) // |se dibujan diversos puntos mediante ciclos
        {
            glVertex2i(20*i,250);
            glVertex2i(250,20*i);
        }
    glEnd();
    glFlush();
}
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



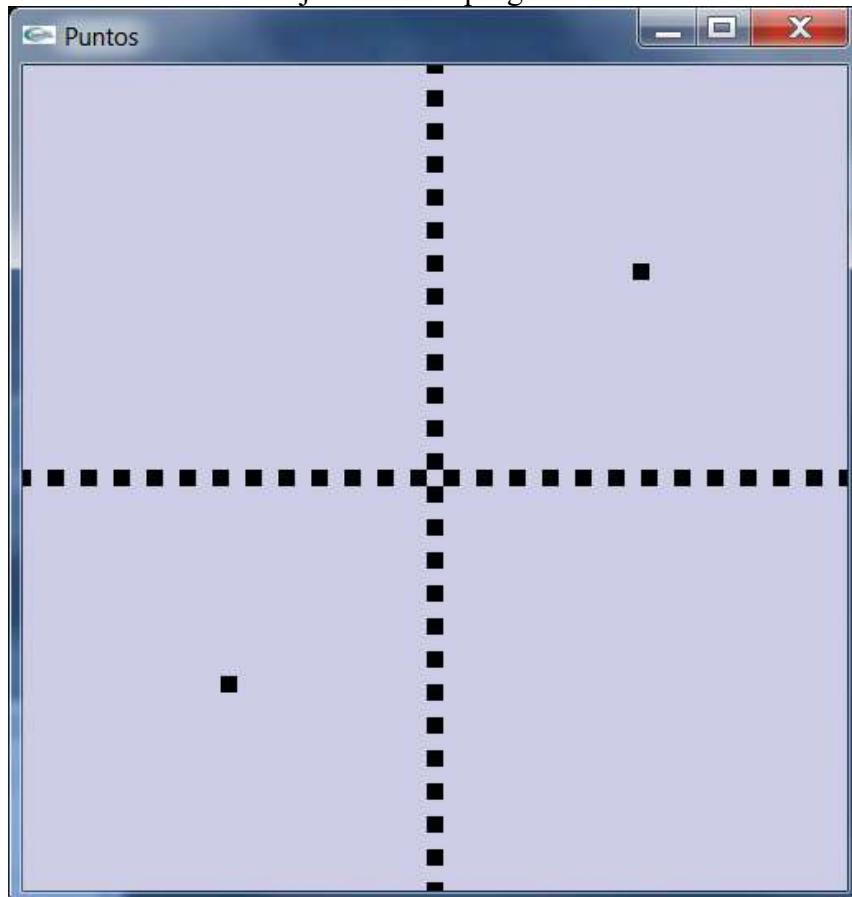
```

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Puntos");
    inicializa();
    glutDisplayFunc(dibuja);
    glutMainLoop();
    return 0;
}

```

92

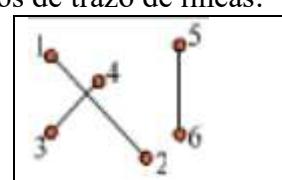
Ejecución del programa



7.7.2. Líneas

Contamos con tres comandos distintos para la misma cantidad de modos de trazo de líneas:

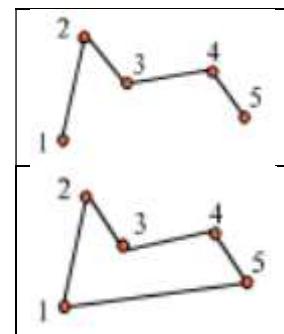
- **GL_LINES** permite trazar una recta individual. Se traza por pares de vértices, uno para el inicio y otro para el fin de trazo. Si se indicasen vértices impares el último será ignorado.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

- **GL_LINE_STRIP** traza varias rectas interconectadas, una detrás de otra.
- **GL_LINE_LOOP** traza un conjunto de líneas interconectadas pero asumiendo que estas formarán una figura geométrica cerrada, dado que el último vértice especificado será conectado con el primero.



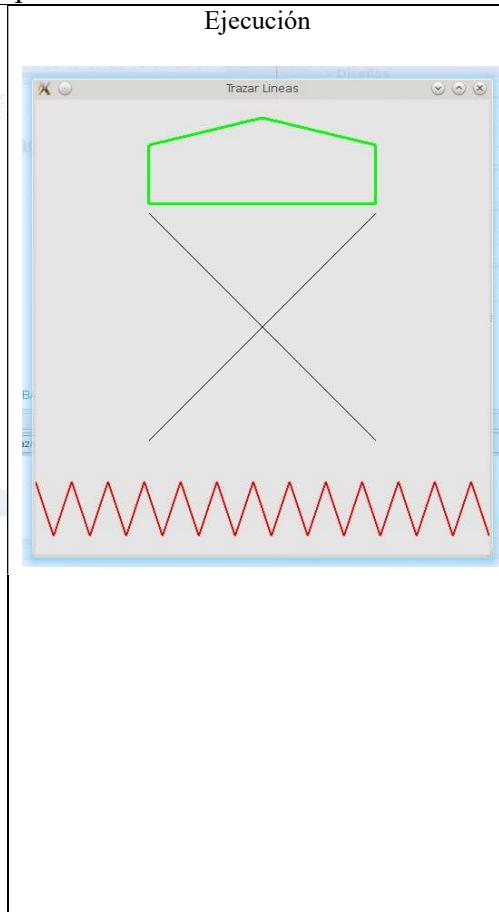
93

Debe ser evidente que en los primeros dos casos es necesario un mínimo de dos vértices, mientras que en el tercero (GL_LINE_LOOP) se requieren al menos tres puntos.

A continuación, se presenta un código que muestra distintos usos de algunas de las primitivas de línea. Sólo se incluye el código del procedimiento que hace el trazado. Para su correcto funcionamiento es necesario agregar también la biblioteca **cmath**.

Ejemplo 7.4. Código que muestra distintos usos de las primitivas de línea.

```
void dibuja(void)
{
    float negro[]={0.0,0.0,0.0,0.0};
    int i, vecpoint[]={375,375},vecpoint2[]={125,375};
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3fv(negro);
    glLineWidth(1);
    glBegin(GL_LINES);
        glVertex2i(125,125);
        glVertex2iv(vecpoint);
        glVertex2iv(vecpoint2);
        glVertex2i(375,125);
    glEnd();
    glColor3f(1.0,0.0,0.0);
    glLineWidth(2);
    glBegin(GL_LINE_STRIP);
        for(i=0;i<=25;i++)
    {
        glVertex2i(i*20,50+pow(-1,i)*30);
    };
    glEnd();
    glColor3f(0.0,1.0,0.0);
    glLineWidth(3);
    glBegin(GL_LINE_LOOP);
        glVertex2i(125,385);
        glVertex2i(375,385);
        glVertex2i(375,450);
        glVertex2i(250,480);
        glVertex2i(125,450);
    glEnd();
    glFlush();
}
```



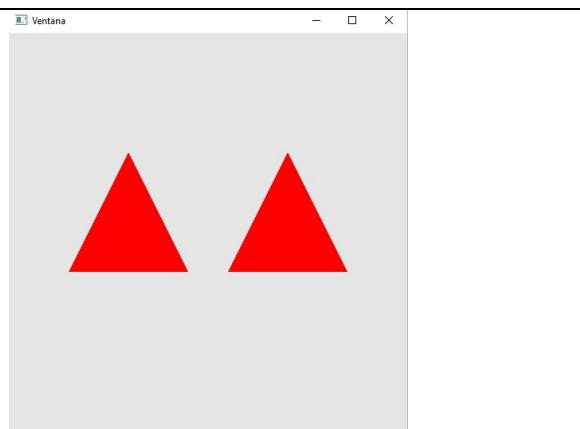
"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

7.7.3. Polígonos

Los polígonos son primitivas que permiten crear superficies cerradas rellenas de uno o varios colores. Éstas primitivas son usadas en graficación 3D para formar objetos “semisólidos”, dado que las figuras creadas están huecas.

Los polígonos definidos como primitivas son triángulos y rectángulos. Así la primitiva que dibuja triángulos (`GL_TRIANGLES`) es el polígono más simple, la cual requiere de vértices que van de tres en tres, en el siguiente ejemplo se dibujan dos triángulos (sólo se muestra el código que crea las primitivas)⁵.



```

void dibuja()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_TRIANGLES);
        glVertex2f(0.5,-1.0);
        glVertex2f(2.0,2.0);
        glVertex2f(3.5,-1.0);
        glVertex2f(-0.5,-1.0);
        glVertex2f(-2.0,2.0);
        glVertex2f(-3.5,-1.0);
    glEnd();
    glFlush();
}

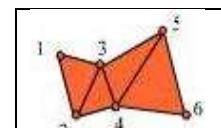
```

El dibujo de los triángulos en el ejemplo anterior no realiza de la misma manera, de acuerdo a la declaración de los vértices, vemos que se usó un sentido horario en el triángulo derecho y un sentido antihorario en el triángulo izquierdo. Cualquier polígono cuyos vértices se declaran en sentido horario se consideran positivos, mientras que si se declaran en sentido antihorario entonces son negativos. En OpenGL se considera que los polígonos negativos tienen un encare frontal, así en nuestro ejemplo, el triángulo izquierdo nos muestra su cara frontal.

Por defecto OpenGL muestra ambas caras, pero se pueden mostrar las caras frontales si se incluye la llamada `glEnable(GL_CULL_FACE)` en la inicialización de la ventana. Para mostrar caras frontales o traseras se invierte el comportamiento de OpenGL, llamando a la función `glFrontFace` que acepta a los parámetros `GL_CW`, para considerar los polígonos positivos con un encare frontal, o bien, `GL_CCW`, para considerar los polígonos negativos como los del encare frontal.

Lo mismo que la primitiva de líneas, la primitiva de triángulos tenemos otras dos variantes:

- **GL_TRIANGLE_STRIP**, dibuja superficies triangulares interconectadas.



⁵ No hay ninguna fuente en el documento actual.

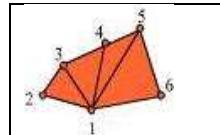


“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

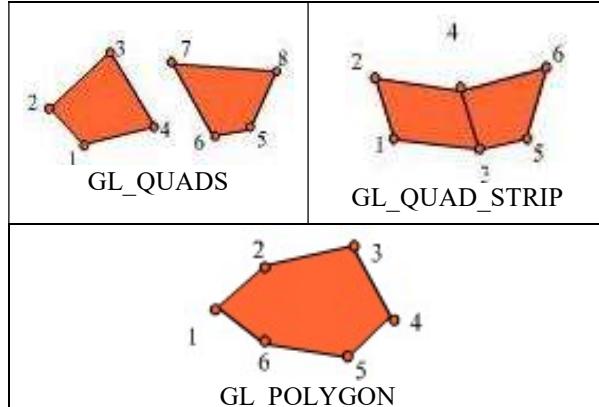


- **GL_TRIANGLE_FAN**, genera polígonos en torno de un punto central, el cual es constituido por el vértice inicial.



La otra primitiva poligonal es **GL_QUADS** que dibuja rectángulos, funciona de manera equivalente a la primitiva de triángulos. Una variante de la misma primitiva es **GL_QUAD_STRIP** que genera superficies rectangulares encadenadas.

Añadida a las anteriores tenemos a **GL_POLYGON** que genera superficies poligonales en función de la cantidad de coordenadas indicada, siempre que se indiquen más de dos coordenadas.

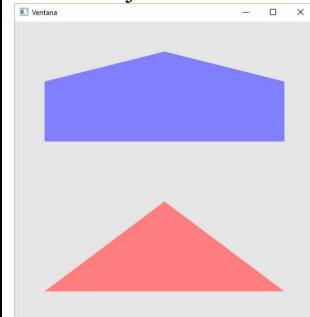


A continuación presentamos un procedimiento de ejemplo para el trazado de polígonos. Observe las maneras en que se declaran los vértices.

Ejemplo 7.5. Creando polígonos.

```
void dibuja()
{
    int coords1[3][2]={{50,50},{450,50},{250,200}};
    int coords2[5][2]={{50,300},{450,300},{450,400},{250,450},{50,400}};
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.5,0.5);
    glBegin(GL_TRIANGLES);
        glVertex2iv(coords1[0]);
        glVertex2iv(coords1[1]);
        glVertex2iv(coords1[2]);
    glEnd();
    glColor3f(0.5,0.5,1.0);
    glBegin(GL_POLYGON);
        for (int i=0;i<5;i++)
        {
            glVertex2iv(coords2[i]);
        }
    glEnd();
    glFlush();
}
```

Ejecución



7.8. Modelo de sombreado⁶

Llamamos modelo de sombreado al método que usa OpenGL para colorear los polígonos. Esta condición se especifica con la función **glShadeModel**, que tiene como parámetros a **GL_FLAT**, que provoca que se rellene la superficie con el color que se define en el último parámetro; si es **GL_SMOOTH**, el relleno de color se interpolará de cada uno de los

⁶ No hay ninguna fuente en el documento actual.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

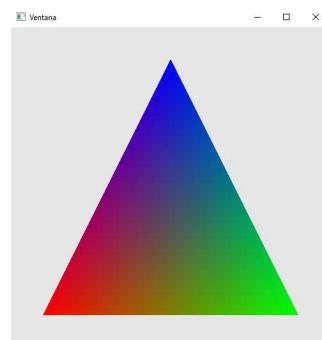
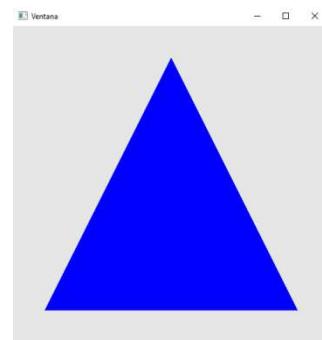


colores activos en la definición de cada vértice. En el siguiente ejemplo se muestran éstas características.

Ejemplo 7.6. Módulos mostrando el efecto de usar glShadeModel: en primer término se muestra al módulo usando como argumento a GL_FLAT, lo que provoca que el triángulo se ilumine de azul, ya que ese es el último color indicado en el último vértice; en segundo término se usa el argumento GL_SMOOTH provocando la interpolación entre colores rojo, verde y azul.

```
void dibuja()
{
    int coords[3][2]={{50,50},{450,50},{250,450}};
    glClear(GL_COLOR_BUFFER_BIT);
    glShadeModel(GL_FLAT);
    glBegin(GL_TRIANGLES);
        glColor3f(1.0,0.0,0.0);
        glVertex2iv(coords[0]);
        glColor3f(0.0,1.0,0.0);
        glVertex2iv(coords[1]);
        glColor3f(0.0,0.0,1.0);
        glVertex2iv(coords[2]);
    glEnd();
    glFlush();
}

void dibuja()
{
    int coords[3][2]={{50,50},{450,50},{250,450}};
    glClear(GL_COLOR_BUFFER_BIT);
    glShadeModel(GL_SMOOTH);
    glBegin(GL_TRIANGLES);
        glColor3f(1.0,0.0,0.0);
        glVertex2iv(coords[0]);
        glColor3f(0.0,1.0,0.0);
        glVertex2iv(coords[1]);
        glColor3f(0.0,0.0,1.0);
        glVertex2iv(coords[2]);
    glEnd();
    glFlush();
}
```



7.9. Sentencias útiles

Dos sentencias que pueden facilitar el dibujo de objetos son:

- Una llamada predefinida a partir de otras primitivas es **glRect** permite dibujar rectángulos eficientemente, indicando para ello dos vértices esquina diagonalmente opuestos. Cada comando requiere de cuatro argumentos organizados en dos pares consecutivos de coordenadas (x,y), o bien dos arreglos apuntando a esas coordenadas. En el caso de coordenadas 3D el rectángulo se dibuja en el plano $z=0$. La instrucción como tal tiene variantes para cada tipo de datos usados y/o si se proveen como escalares o vectoriales. Para escalares usar: **glRectd**, para tipos



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



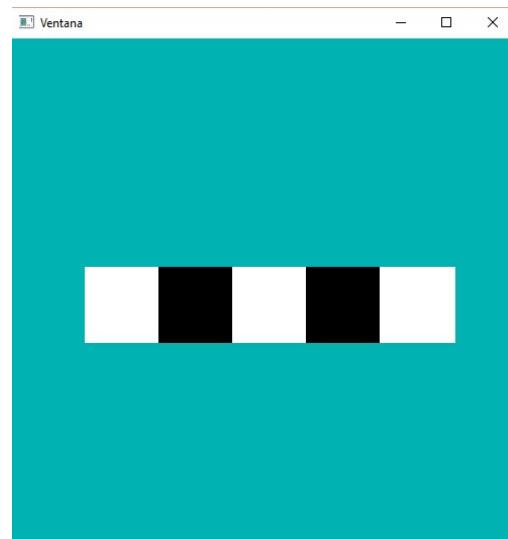
doubles; **glRectf**, para tipos flotantes; **glRects**, para tipos short; **glRecti**, para tipos enteros. Si se usan vectorialmente se debe agregar una ‘v’ al comando, por ejemplo, para vector fotante: **glRectfv**.

- **GLboolean** es un tipo de dato definido en OpenGL para asignar un valor lógico a una variable, las cuales pueden ser: **GL_TRUE**, cuando el valor es verdadero; y, **GL_FALSE**, cuando es falsa.

En el código que se muestra en el ejemplo siguiente puede observar cómo se usan de manera ingeniosa las variables de tipo booleano para decidir de qué color se trazan cuatro rectángulos.

Ejemplo 7.7. Procedimiento ejemplo de uso de tipos de datos booleanos en OpenGL y del comando Rect en su modalidad entera. Cada rectángulo se traza gracias al ciclo *for* en el cual cada vez que se traza un rectángulo se cambia la condición de verdad de la variable ‘colorBlanco’ (originalmente inicializada en ‘true’), provocando que cada cuadro sea de un color distinto.

```
void dibuja()
{
    GLboolean colorBlanco=GL_TRUE;
    float blanco[]={1.0,1.0,1.0};
    float negro[]={0.0,0.0,0.0};
    glClear(GL_COLOR_BUFFER_BIT);
    for (int i=1;i<=5;i++)
    {
        if (colorBlanco)
        {
            glColor3fv(blanco);
        }
        else
        {
            glColor3fv(negro);
        }
        glRecti(i*73,200,75+i*73,275);
        colorBlanco=!colorBlanco;
    }
    glFlush();
}
```



7.10. Ejercicios sugeridos

Realice las siguientes actividades:

- I. Realice el programa que, en una ventana de 500×500 pixeles dibuje sólo el perímetro de una circunferencia de 200 pixeles de radio. Sugerencia: considere la ecuación del círculo centrado en las coordenadas (h,k), que es: $(x - h)^2 + (y - k)^2 = r^2$.
- II. Con base en el ejercicio anterior, realice el procedimiento que permita inscribir en el círculo una superficie hexagonal regular. Sugerencia: use una transformación polar.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

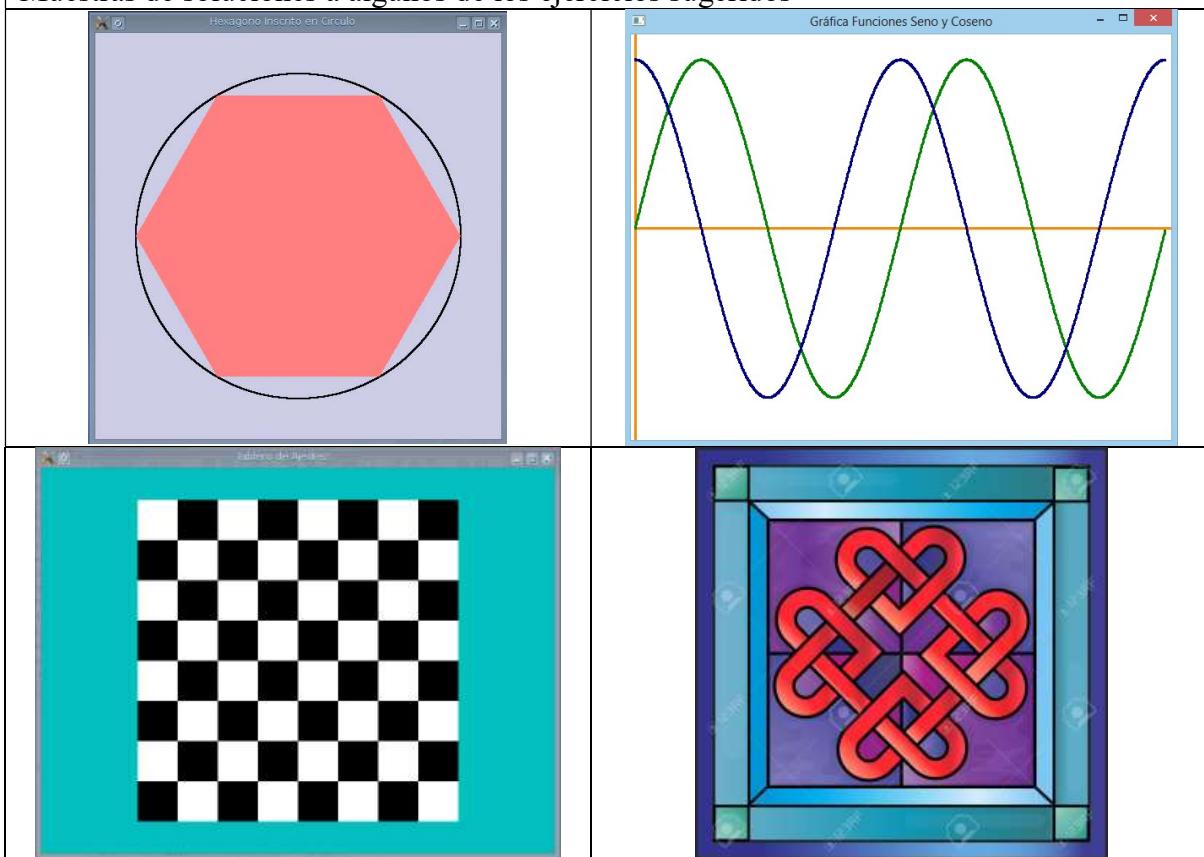
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



- III. Crear un programa que realice la gráfica de las funciones Seno y Coseno, cada uno en distinto tono, en una ventana de 640×480 pixeles. Se deben dibujar al menos dos periodos y representar una rejilla en unidades decimales.
- IV. Usar la idea del ejemplo 7.7 y dibujar un tablero de ajedrez en una ventana de 640×480 pixeles.
- V. Buscar un modelo de vitral geométrico en internet y dibujarlo en una ventana de 500×500 pixeles.

98

Muestras de soluciones a algunos de los ejercicios sugeridos



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

8. Puertos

En esta unidad el estudiante:

- Diseña e implementa programas con manejo de puertos.

8.1. Comentarios previos

De acuerdo con los lineamientos solicitados en el temario de ésta materia, en la última unidad debe proponerse al estudiante la realización de una práctica-proyecto que permita visualizar la manipulación de dispositivos externos a través de los puertos de una computadora.

Debido a los rápidos cambios tecnológicos y a la disponibilidad de equipo en los últimos siete años, ésta actividad se ha realizado de dos distintas formas: de 2011 a 2014 se implementó una práctica tendente a manipular el puerto paralelo de una PC; en los últimos dos años, 2015 y 2016 se optó por elegir la implementación de computadoras embebidas Raspberry Pi que facilitan enormemente el direccionamiento de puertos.

En este contexto ofrecemos en primer término una conceptualización muy básica de la manipulación de puertos en una computadora; en segundo término, ofrecemos, a manera de antecedente, una práctica implementada sobre Visual C# para el encendido de leds a través del puerto paralelo; por último, mostramos dos sencillas prácticas que igualmente permiten encender leds mediante el uso del Raspberry Pi y el lenguaje de programación Python.

8.2. Arquitectura de puertos de una computadora

Los puertos se podrían entender como las terminales a través de las cuales se comunican los periféricos con el CPU, aunque no son exactamente lo mismo. Las terminales físicas usan distintas normas (forma de la conexión) para distinguir los periféricos que se pueden conectar en ellos.

Hasta antes del 2006, los puertos de entrada y salida (E/S o I/O) más relevantes en la arquitectura de una computadora (para el mercado mexicano) eran el puerto serial también llamado COM y el puerto paralelo o LPT; el primero usado en la comunicación entre dispositivos computadores o con dispositivos periféricos como el mouse; el segundo era en el que comúnmente se conectaban las impresoras.



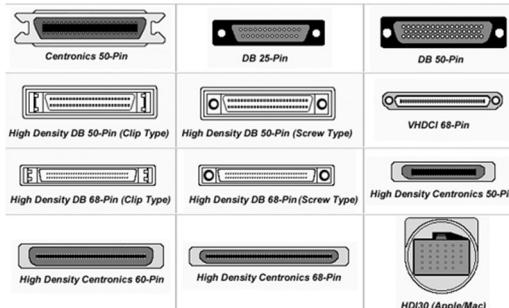
"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



FireWire o IEEE1394

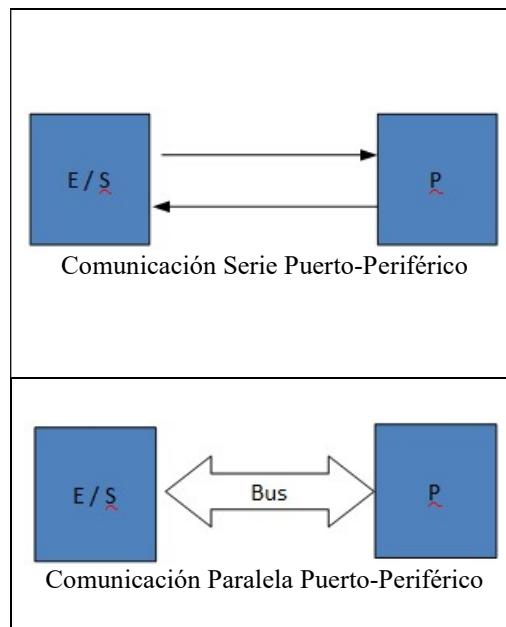
External SCSI Connectors



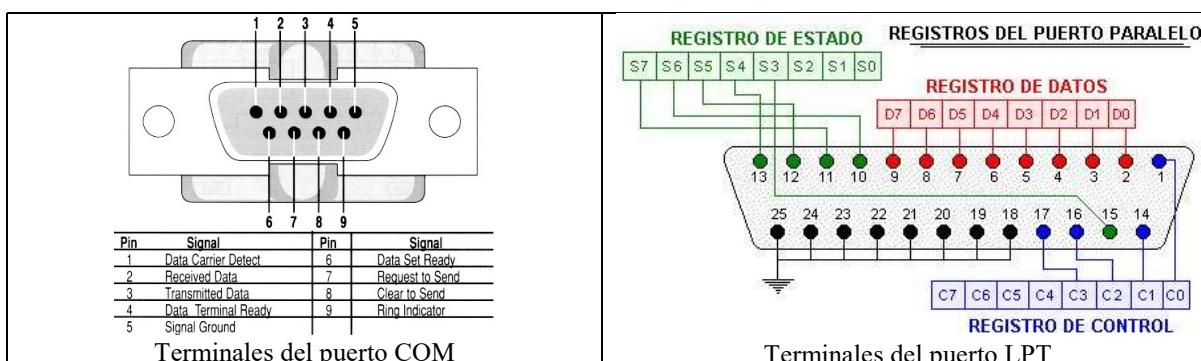
100

El puerto COM se denomina serial debido a que la comunicación con el periférico se realiza bit a bit, es decir, para formar una palabra en el destino se debe enviar un bit por ciclo de comunicación, mientras que en el destino existe un buffer a manera de registro que va almacenando uno a uno cada bit, hasta que se forma la palabra completa. La comunicación puede parecer lenta, pero a cambio no se requiere más que de un hilo para la comunicación.

El puerto LPT en cambio, envía la información en forma de palabra completa cada vez que hay un ciclo de comunicación, cada bit viaja en su propio hilo, de ésta manera la información viaja rápidamente, pero en cambio tiene la limitante de requerir de manejar un cable compuesto de tantos hilos como bits deban viajar.



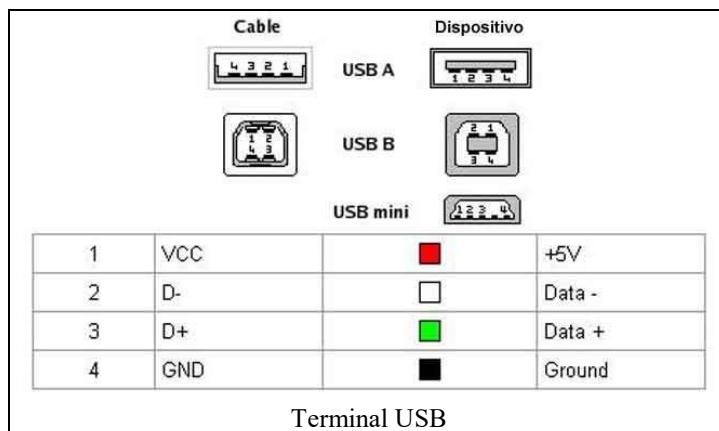
Los puertos: paralelo y serial, utilizaban terminales de la norma SCSI en su formato DB tal como se puede apreciar en la figura siguiente.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Actualmente el puerto serie y el puerto paralelo han caído en desuso e incluso se les considera obsoletos ante el auge de las tecnologías que utilizan el puerto universal serie (USB), implementada desde 1996 en computadoras IBM, y generalizadas como estándar a inicios de la década 2000. Define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.



Este bus fue desarrollado un grupo de empresas del sector que buscaban unificar la forma de conectar periféricos a sus equipos, por aquella época poco compatibles entre sí, entre las que estaban Intel, Microsoft, IBM, Compaq, DEC, NEC y Nortel. La primera especificación completa 1.0 se publicó en 1996, pero en 1998 con la especificación 1.1 comenzó a usarse de forma masiva⁷. Actualmente se utilizan las especificaciones 2.0 y 3.0. Cada especificación supone una mejora en la velocidad de transferencia de datos como se muestra en la siguiente tabla⁸

Versión de puerto	Velocidad máxima en Megabits por segundo	Velocidad máxima en (MegaBytes/segundo)
USB 1.0 (Low Speed)	1.5 Mbps	187.5 KB/s
USB 1.1 (Full Speed)	12 Mbps	1.5 MB/s
USB 2.0 (Hi-Speed)	480 Mbps	60 MB/s
USB 3.0 (Super Speed)	3200 Mbps / 3.2 Gbps	400 MB/s

La finalidad de los puertos es permitir a los periféricos el acceso al bus del sistema, de este modo cada módulo de E/S implementa la lógica necesaria para permitir la comunicación entre el periférico y el bus. Ésta manera de comunicar periféricos con el CPU se debe, entre

⁷ Wikipedia.org. (28/05/2016). *Universal Serial Bus*. Obtenido de:

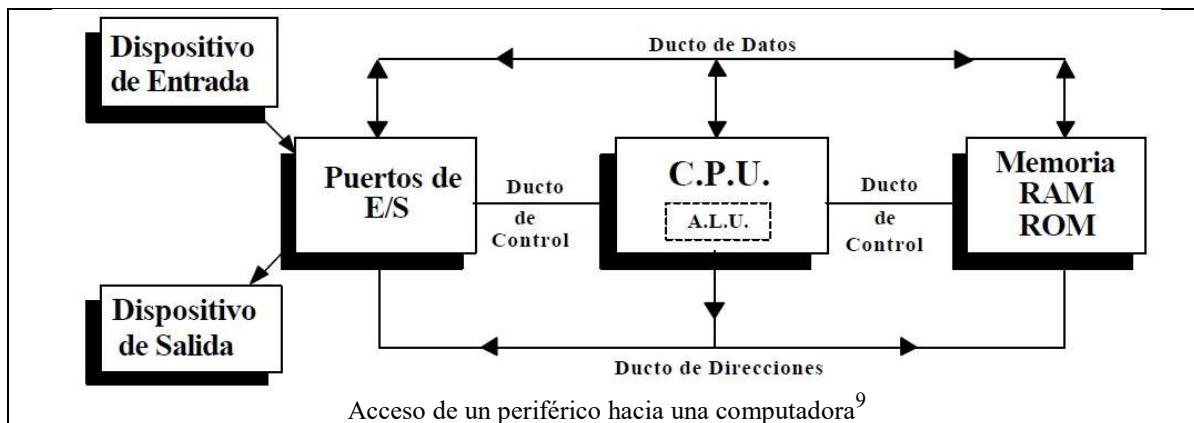
https://es.wikipedia.org/wiki/Universal_Serial_Bus

⁸ InformaticaModerna.com (30/05/2016). *El puerto USB 1.0/2.0 y 3.0*. Obtenido de:

http://www.informaticamoderna.com/El Puerto_USB.htm

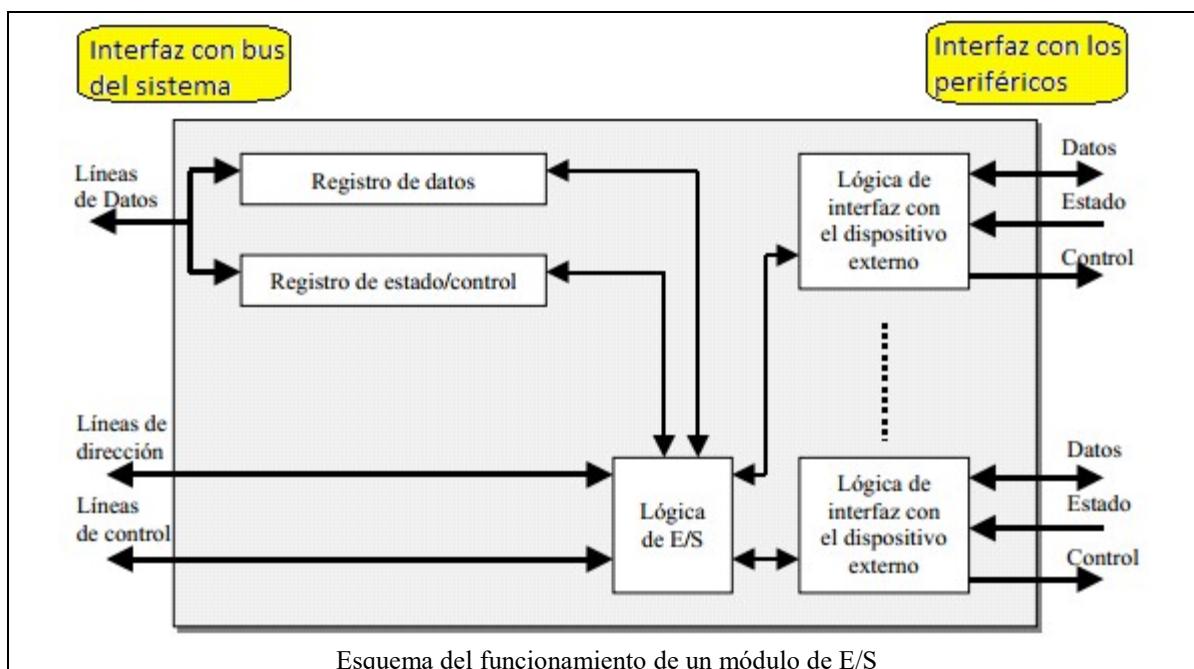


otras cosas a la mucho menor velocidad en que se transfieren datos (comparada con el mismo microprocesador y la memoria), y a los distintos tamaños de palabra.



102

El módulo de E/S consta fundamentalmente de dos interfaces: uno interno para la comunicación CPU-módulo y otro externo para la comunicación módulo-periférico. De este modo sus funciones principales son: control y temporización, comunicación con el CPU, comunicación con los dispositivos, almacenamiento temporal de datos y detección de errores.



⁹ Fernanda Guevara Bello. (31/05/2016). *Arquitectura de una computadora*. Obtenido de: <http://componentesprincipalesdecomputadora2.blogspot.mx/>



El proceso de comunicación entre el CPU y un periférico implica:

- i. Descodificar las órdenes dadas por el CPU al módulo de E/S, las cuales se envían usando el bus de control.
- ii. Usar el bus de datos el CPU y los módulos de E/S para intercambio de datos.
- iii. Determinar el estado (BUSY, READY), y, en su caso, informar posibles situaciones de error.
- iv. Reconocer la dirección del dispositivo de E/S. El sistema asigna una dirección única (cuál si fuera un registro) para cada uno de los periféricos que controla.

Dado que cada módulo requiere de su propio dispositivo de hardware, cada uno se diferencia de otro mediante una lógica de registros de direccionamiento, incluso el propio puerto ocupa su propia dirección en el marco de los componentes de una computadora.

Dirección		Descripción	Dirección		Descripción
Desde	Hasta		Desde	Hasta	
000	00F	Control DMA (Acceso directo a memoria)	2E8	2EF	Puerto COM4
020	02F	Control de Interrupciones maestro	2F8	2FF	Puerto COM2
030	03F	Control de Interrupciones esclavo	370	377	Control Disco Flexible
040	043	Temporizador	378	37F	Puerto LPT2
060	060	Teclado	3B0	3BB	Adaptador video (mono)
061	061	Altavoz	3BC	3BF	Puerto LPT1
170	17F	Primer disco duro	3E0	3EF	Puerto COM3
200	20F	Puerto de juegos	3F8	3FF	Puerto COM1
278	27F	Puerto LPT3	220	22F	Tarjeta de sonido

Tabla de direcciones típica en una antigua computadora x86

Un boceto de algoritmo que implementa la transferencia de datos es entonces:

- I. El CPU revisa el estado del dispositivo preguntando al módulo de E/S a través del cual está conectado.
- II. El módulo de E/S indica el estado del dispositivo.
- III. Si el dispositivo está funcionando y preparado para transmitir, el CPU solicita la transferencia del dato mediante una orden al módulo de E/S.
- IV. El módulo de E/S obtiene un dato del dispositivo externo.
- V. Los datos se trasfieren desde el módulo de E/S a la CPU.

8.3. Práctica propuesta 1: Control de LED's por puerto paralelo

En ésta práctica buscamos mostrar cómo es posible controlar el encendido de LED's mediante el puerto paralelo de una computadora de escritorio (Desktop). Se implementa un



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

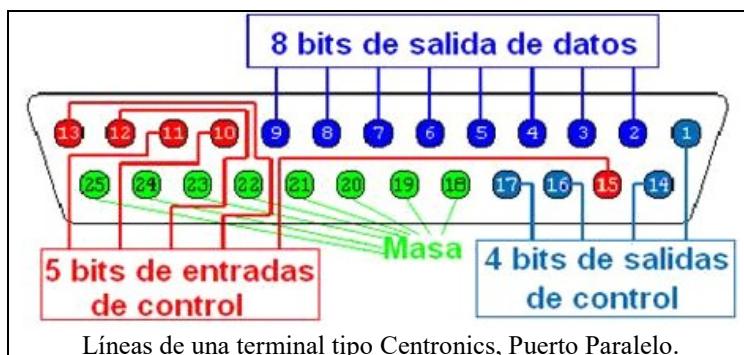
programa en Microsoft Visual C#, por la facilidad en el desarrollo de una aplicación visual e interactiva en sistemas operativos Windows desde XP hasta 8.1 y se basa casi fielmente en las ideas aportadas por Ángel Acaymo M. G., en su proyecto “*Interfaz puerto paralelo LPT*”¹⁰.

Entrando en materia, el puerto paralelo tipo Centronics (IEEE 1284, formato de terminal DB25), es el puerto que hasta antes de la entrada del USB, era usado como terminal de la impresora, aunque igualmente podían conectarse por este medio escáneres, interfaces de red Ethernet a 10Mb, unidades ZIP, e incluso comunicar paralelamente dos computadores.

El puerto paralelo cuenta con 17 líneas de señal y 8 líneas de tierra. Las líneas de señal se clasifican en tres grupos:

- 4 líneas de control, que servían fundamentalmente para el intercambio de mensajes desde el PC a la impresora.
- 5 líneas de estado, indicaban al PC el estado de la impresora (falta de papel, impresora ocupada, error en la impresora).
- 8 líneas de datos, empleadas para el envío bidireccional de información entre la PC y el dispositivo externo.

Cada una de estas líneas se referencian internamente de modo independiente mediante un registro, cada uno llamado del mismo modo en que se llama a las líneas: control, estado y datos. Las líneas de datos cuentan con un retenedor que mantiene el último valor que fue escrito en ellas.



En cuanto a las características eléctricas del puerto tenemos: tensión de nivel alto de 3.3 a 5 V; 0V para tensiones de nivel bajo; corriente de salida máxima de 2.6 mA, y, corriente de entrada máxima de 24 mA.

Las direcciones base asignadas para el puerto LPT1 y LPT2 eran típicamente 0x378 y 0x278 respectivamente. El registro de control (bidireccional) posee la dirección 0x037A, el

¹⁰ Ángel Acaymo M.G. (2010) *Interfaz Puerto Paralelo LPT*. Obtenido de:
<http://es.slideshare.net/Metaconta2/interfaz-puerto-paralelo-lpt>



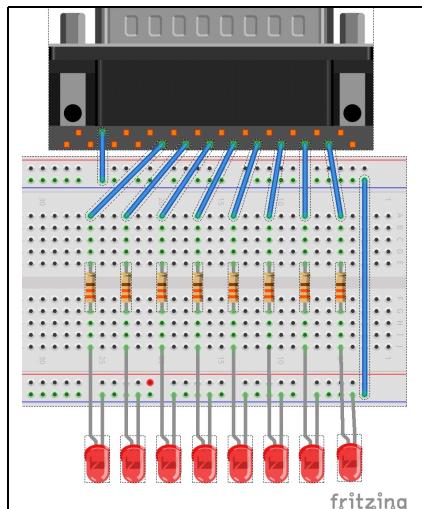
“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

registro de estado (de entrada, solamente) posee la dirección 0x379, mientras que el registro de datos (bidireccional) es 0x378.

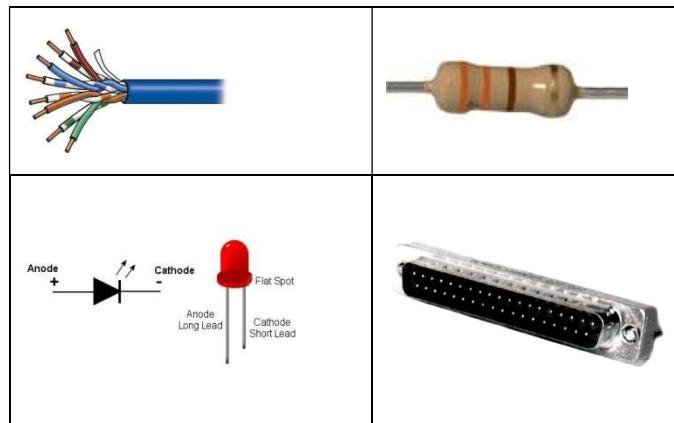
8.3.1. Pasos Previos

Las terminales que usaremos del puerto son justamente los 8 bits de datos y cualquiera de las terminales de tierra o masa. Armaremos un circuito como el que se muestra en la siguiente figura.



De este modo el material requerido es:

- Computadora personal con puerto paralelo.
- Protoboard.
- Cable UTP, 1.5 m.
- Resistencias de 330 Ohms, 8 unidades.
- LED's sencillos, no importa el color, 8 unidades.
- Terminal macho y concha para terminal DB25.
- Cautín y soldadura.



Lo primero es soldar las terminales del puerto DB25 a los hilos del cable UTP, según se muestra en el gráfico mostrado más hacia arriba en ésta página, tomando en consideración qué color de hilo usamos en cada pin para poder ubicar al bit menos significativo. Opcionalmente puede cubrir el DB25 con la concha. Luego se hace el arreglo de LED's y resistencias mostrado. Dado que el puerto paralelo puede dañarse (y con él la placa madre) con algún voltaje y corriente inadecuado, es importante que cuente con la asesoría de su profesor.



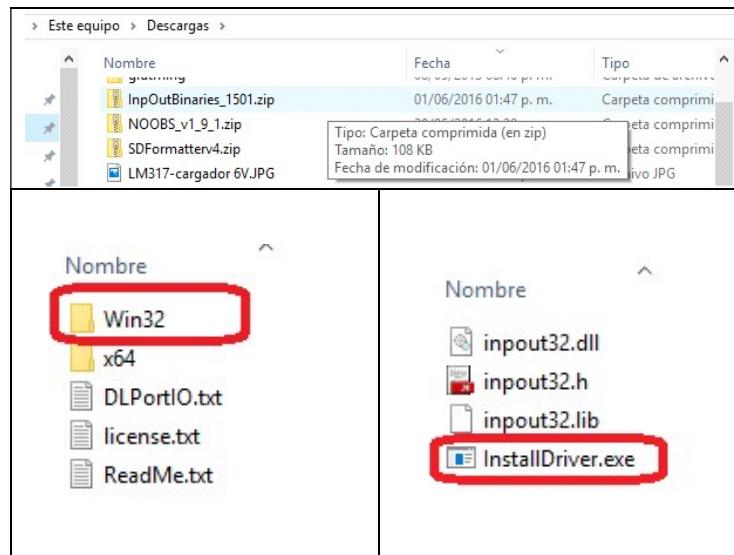
"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel./Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

Para manipular el puerto tendremos el inconveniente de que Windows (desde la versión XP hasta el 10) es que virtualizan el hardware sobre el cual operan, de tal modo que no es posible mandar datos directamente a las direcciones correspondientes porque son interceptadas y bloqueadas, por razones de seguridad. De este modo el uso del editor (CodeBlocks o Dev-Cpp) y el del compilador (MinGW) sobre un entorno Windows, requeriría de ciertas adaptaciones que serían largas de explicar e implementar. Para resolver ésta problemática proponemos experimentar con el IDE que nos proporciona el Microsoft Visual Studio Versión Express 2010, enfocándonos en el uso del lenguaje Visual C#, que con algunas diferencias es similar al que hemos venido usando. Aunado a lo anterior, para evitar las intercepciones del sistema, deberemos descargar e instalar la biblioteca **InpOut32.dll** de highrez del sitio:

<http://www.highrez.co.uk/Downloads/InpOut32/default.htm>

En esa página buscar el link de descarga para [Binaries only - x86 & x64 DLLs and libs](#). Una vez descargado hay que descomprimir la carpeta (imagen abajo a la izquierda). Entrar a la carpeta Win32, y ahí ejecutar el programa “InstallDriver(.exe)” (imagen abajo a la derecha). Aunque también se puede usar la que viene en la carpeta x64, preferimos la versión de 32 bits por compatibilidad, ya que ésta funciona incluso en computadoras de 64 bits.



Otro paso previo es verificar las direcciones vaya al administrador de dispositivos de su PC (Clic derecho en equipo>Propiedades>Administrador de dispositivos>Puertos Serie y Paralelo.

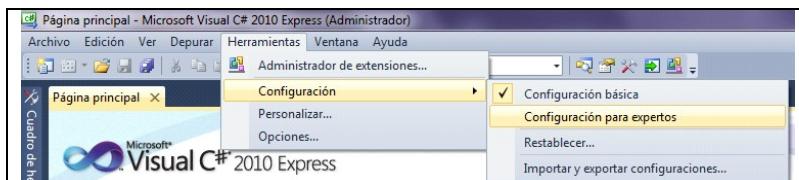
8.3.2. Construcción del entorno gráfico

Ahora pasamos a la construcción del entorno gráfico y de programación de nuestra práctica, para lo cual, si va a usar VC# 2010, antes de crear el proyecto configuraremos la plantilla de programación para expertos en: Herramientas>Configuración>Configuración para expertos, ésta acción en versiones anteriores (2005, 2008) no es necesario.



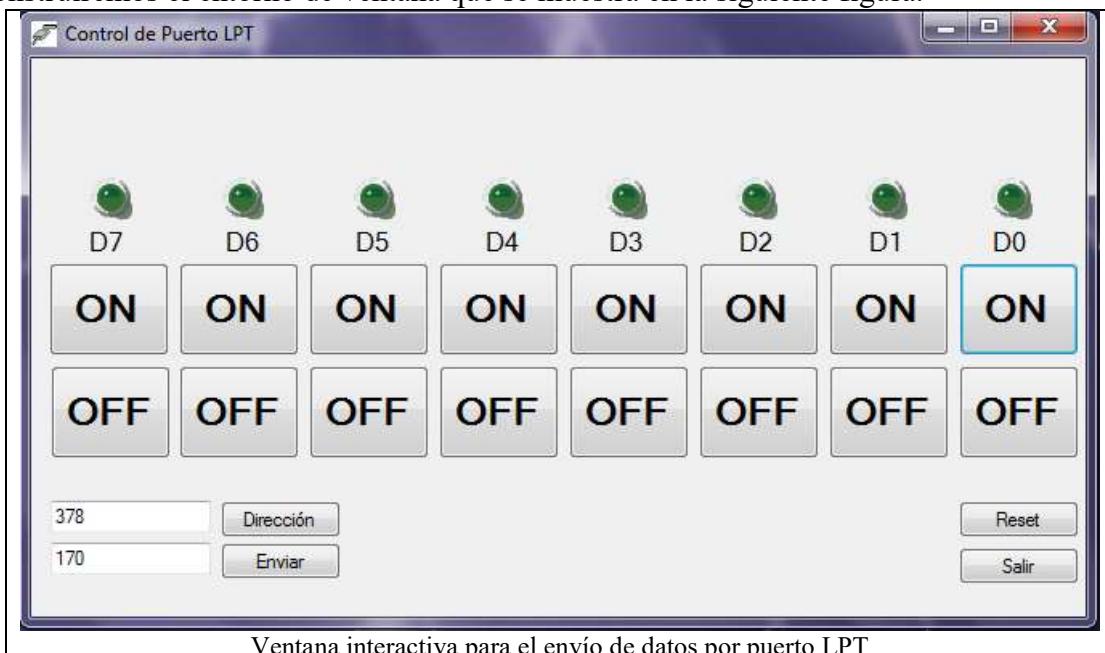
“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



107

Construiremos el entorno de ventana que se muestra en la siguiente figura.



Así entonces, con la guía de su instructor, construya la ventana principal creando un proyecto de WindowsForms llamado: Control_Puerto_Paralelo. (Incluya los guiones bajos). En el panel de propiedades del WindowsForms configure:

- Size: 673 ancho, 385 alto.
- StartPosition: CenterScreen.
- Text: Control de Puerto LPT.

Luego, agregue los 8 botones superiores con las siguientes propiedades:

- (Name): button_Dx_ON.
- Font: Microsoft Sans Serif, Negrita, 18 pts.
- Text: ON.
- Anchor: Top.
- Location: 578; 96 (opcional para primer botón).
- Size: 75; 58 (propuesto)..



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

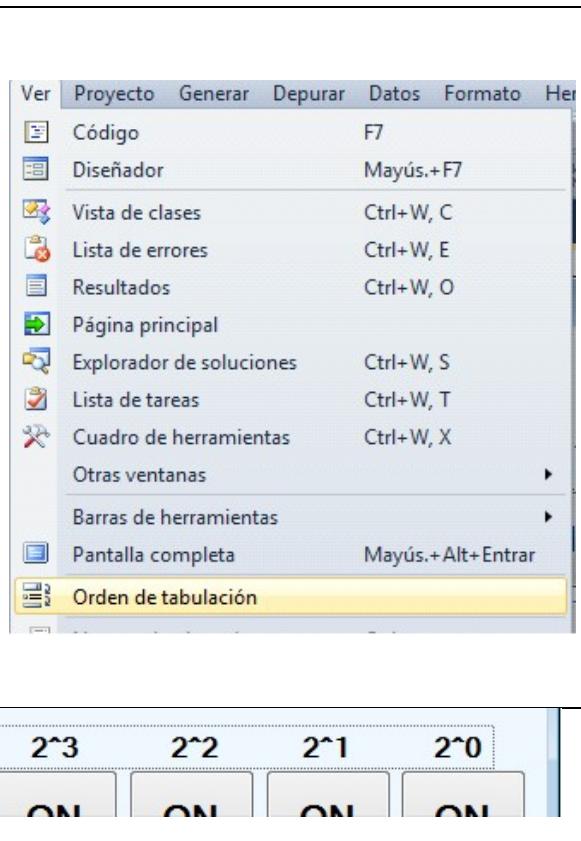
En el caso de la propiedad (Name), éste valor, debe ir cambiando de botón a botón, variando a x en valores de 0 a 7. Por ejemplo, para el primer botón debería ser *button_D0_ON*, el siguiente sería *button_D1_ON*, y así consecutivamente.

Luego seleccione todos los botones incrustados y agregue una copia en la parte inferior de éstos (son los botones en cuyo texto aparecerá un OFF), cambie las propiedades:

- (Name): button_Dx_OFF (de Nuevo la ‘x’ representa valores de 0 a 7).
- Text: OFF.

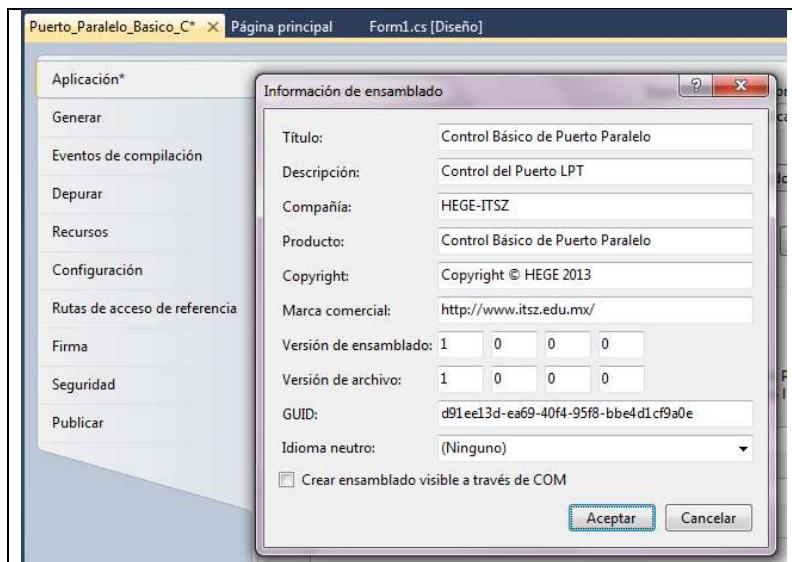
Un paso opcional es indicar el modo de uso de los botones cuando el usuario no tenga mouse. Para ello seleccione todos los botones y vaya al menú Ver>Orden de Tabulación, y seleccione el adecuado. En este punto se puede depurar (compilar y ejecutar, tecla F5 o dar clic en la barra de herramientas “Iniciar Depuración”) el proyecto para ver cómo se comportan los botones con la tecla Tab (figura a la derecha).

Luego, sobre los botones agregue un label, en donde a espacios adecuados colocará indicadores de potencias de base 2, de izquierda a derecha: 2^7 , 2^6 , ..., 2^0 . Se propone usar Font: Microsoft Sans Serif, 12 puntos, negrita (figura de abajo).



Como siguiente paso, editaremos información relativa a la aplicación que estamos creando, para ello iremos a Proyecto>Propiedades del Proyecto y luego en la ficha General, en la opción Información de ensamblado. Llenamos los espacios según se nos ejemplifica en la siguiente figura. Considere cambiar la información propuesta según se lo indique su instructor.

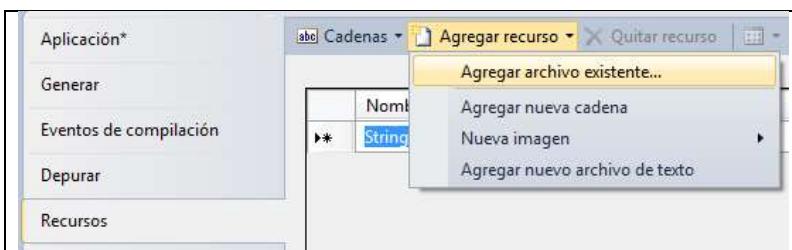
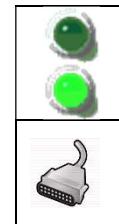




Para el siguiente paso requeriremos imágenes e íconos. Necesitamos la imagen de un led encendido y la de un led apagado, los cuales los puede hallar por internet, éstos nos servirán para ilustrar en la pantalla del monitor las acciones que el usuario realice en el puerto.

Un ícono en forma de puerto servirá para dar realce a la barra de título de nuestra aplicación.

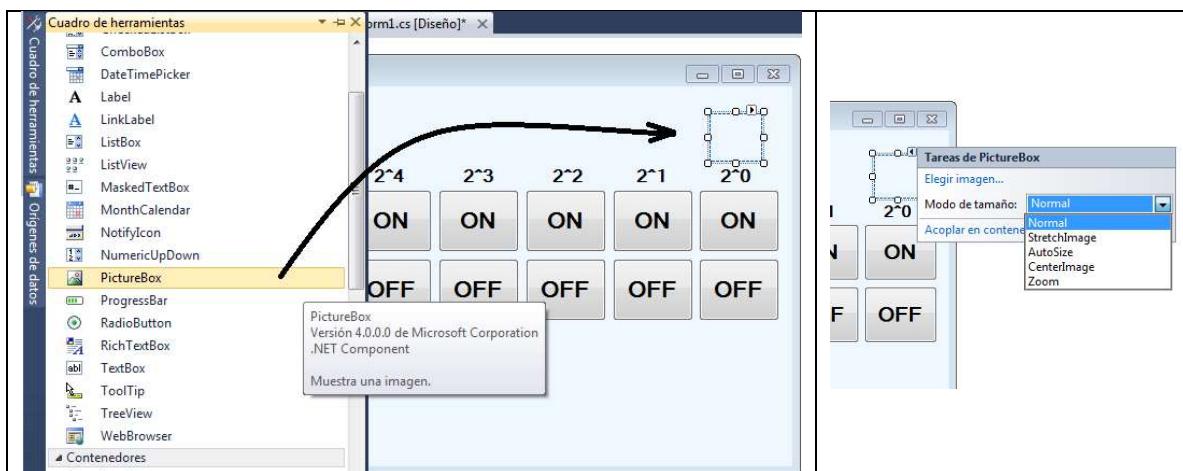
Para añadir los recursos mencionados, activamos la ficha *Recursos*, y en el panel derecho, damos clic en *Agregar recurso*. Navegaremos hasta hallar los íconos e imágenes necesarias. Si ha agregado algún ícono con anterioridad, el mismo estará disponible en la ficha *Aplicación* en la sección *Ícono*, dando clic en el menú desplegable.



Guarde todo y cierre la pestaña. Para que también aparezca el ícono en el Formulario, vaya a la propiedad *Icon* del *WindowsForm*, navegue hasta donde se halla el recurso y selecciónelo.

Para colocar las imágenes agregaremos objetos *PictureBox* desde el *Cuadro de herramientas*. Note sobre el cuadro una pequeña flecha, dé clic sobre ella para navegar y seleccionar la imagen que se va contener, en nuestro caso, un led apagado. Luego puede activar la opción *autoSize* para que el tamaño se adapte por sí sólo.

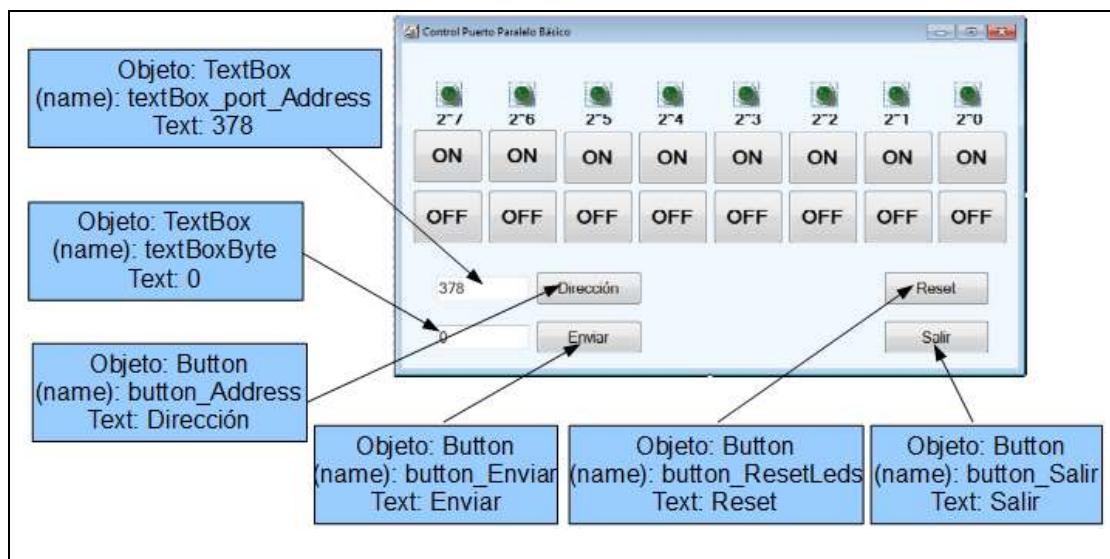




Cambiamos las propiedades de cada *PictureBox* a:

- *(name)*: PictureBox_Dx, la última x es un número entre 0 y 7 según corresponda.
- *SizeMode*: AutoSize.
- *Anchor*: Top.

En la siguiente imagen mostramos la vista previa del arreglo con la inclusión de algunos objetos más, los cuales se detallan en la misma.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

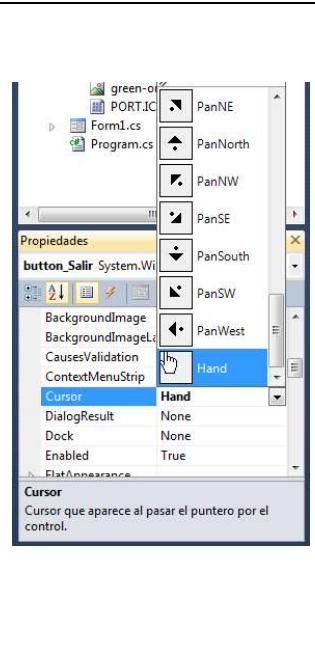
Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Podemos hacer que cuando el puntero se coloque sobre los botones se convierta en una mano, para ello, seleccione todos los botones, y cambie la propiedad *Cursor* a *Hand* (imagen a la derecha).

Como última fase de este proceso copiaremos la biblioteca *inpout32.dll* en la misma carpeta donde se encuentre el ejecutable de nuestro proyecto, generalmente esa dirección es: *Documents>Visual Studio 2010> Projects> NombreProyecto> NombreProyecto> bin> Debug* (imagen inferior).

Debug				
Nombre	Fecha de modifica...	Tipo	Tamaño	
<i>inpout32.dll</i>	20/01/2011 12:07 a...	Extensión de la apl...	96 KB	
Puerto_Paralelo_Basico_C	26/05/2013 06:41 ...	Aplicación	18 KB	
Puerto_Paralelo_Basico_C.pdb	26/05/2013 06:41 ...	Archivo PDB	34 KB	
Puerto_Paralelo_Basico_C.vshost	26/05/2013 09:09 ...	Aplicación	12 KB	
Puerto_Paralelo_Basico_C.vshost.exe.man...	18/03/2010 12:39 a...	Archivo MANIFEST	1 KB	
WindowsFormsApplication1.vshost	26/05/2013 03:39 a...	Aplicación	12 KB	
WindowsFormsApplication1.vshost.exe....	17/03/2010 09:39 ...	Archivo MANIFEST	1 KB	



8.3.3. Codificación

Creamos la clase para importar la biblioteca *inpout* al proyecto. Para ello vaya al menú *Proyecto> Agregar clase*.

De las plantillas disponibles elegimos Clase y damos clic en el botón Agregar. Le daremos por nombre *PortInterop*.

Hechos los pasos anteriores se abrirá una plantilla de programación con el nombre *PortInterop.cs* en donde de forma predeterminada se agregarán líneas de código.

Comenzamos el proceso de programación declarando uso de las rutinas de la clase *Portinterop* (observar en el encabezado la entrada ‘using’) agregando: *using System.Runtime.InteropServices*.

```
PortInterop.cs X Form1.cs [Diseño]
Puerto_Paralelo_Basico_C.PortInterop
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Puerto_Paralelo_Basico_C
{
    class PortInterop
    {
    }
}
```





Posteriormente importaremos la biblioteca *inpout32* dentro de la clase *PortInterop*, escribiendo las instrucciones:

```
[DllImport("inpout32.dll", EntryPoint = "Out32")]
public static extern void Output(int adress, int value);
[DllImport("inpout32.dll", EntryPoint = "Inp32")]
public static extern void Input(int adress);
```

112

```
PortInterop.cs* X Form1.cs [Diseño]
Puerto_Paralelo_Basico_C.PortInterop
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;
namespace Puerto_Paralelo_Basico_C
{
    class PortInterop
    {
        [DllImport("inpout32.dll", EntryPoint = "Out32")]
        public static extern void Output(int adress, int value);
        [DllImport("inpout32.dll", EntryPoint = "Inp32")]
        public static extern void Input(int adress);
    }
}
```

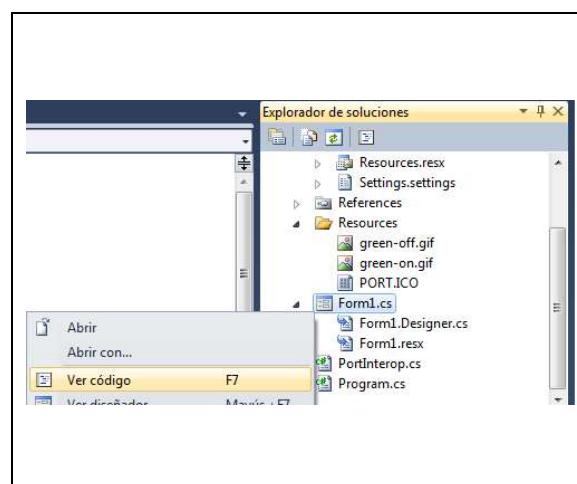
A continuación, incorporaremos procedimientos para llamar a la clase *PortInterop*. Abrimos el código del formulario dando clic derecho a sobre él en el árbol de soluciones, su nombre típicamente es: *Form1.cs*, consulte con su docente de ser necesario.

Declaramos el uso de las propiedades del programa (entre las que se halla la clase *PortInterop*) después de los demás elementos de sistema:

```
using Puerto_Paralelo_Basico_C.Properties;
```

Creamos algunas variables utilitarias globales en la carga de la clase del Formulario:

```
public int i = 0, j = 0, adress = 888;
public int D7, D6, D5, D4, D3, D2, D1, D0;
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



Agregamos un procedimiento que prepare los comandos de control de puerto (imagen inferior). Al mismo tiempo agregaremos un procedimiento que inicialice los *pictureBox* con los LEDs apagados.

Dentro de la rutina de reseteo se invoca al procedimiento *Alerta_1*, cuyo código se lista en la siguiente diapositiva.

```

Form1.cs*  Form1.cs [Diseño]*

WindowsFormsApplication1.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Puerto_Paralelo_Basico_C.Properties;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public int i = 0, j = 0, adress = 888;
        public int D7, D6, D5, D4, D3, D2, D1, D0;
        public Form1()
        {
            InitializeComponent();
        }
    }
}

public partial class Form1 : Form
{
    public int i = 0, j = 0, adress = 888;
    public int D7, D6, D5, D4, D3, D2, D1, D0;
    public Form1()
    {
        InitializeComponent();
        Reset_LEDs();
    }

    private void Reset_LEDs()
    {
        try // Intentamos controlar los posibles errores por omisión (excepción)
        {
            Puerto_Paralelo_Basico_C.PortInterop.Output(adress, 0);
            Puerto_Paralelo_Basico_C.PortInterop.Input(adress);
        }
        catch (DllNotFoundException) //Capturamos el error por ausencia de biblioteca
        {
            Alerta_1();
        }
        pictureBox_D0.Image = Resources.green_off; //Se cargan los gifs LED apagados
        pictureBox_D1.Image = Resources.green_off;
        pictureBox_D2.Image = Resources.green_off;
        pictureBox_D3.Image = Resources.green_off;
        pictureBox_D4.Image = Resources.green_off;
        pictureBox_D5.Image = Resources.green_off;
        pictureBox_D6.Image = Resources.green_off;
        pictureBox_D7.Image = Resources.green_off;
    }
}

```

Procedemos a incluir el módulo *Alerta_1* cuya finalidad es enviar mensajes de error al usuario, labor que hace implementando un *MessageBox*. El código en cuestión es demasiado largo, por lo que lo mostramos en forma ajustada:



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



{

 MessageBox.Show(@"No se encuentra la biblioteca 'input32.dll', debería encontrarse en la misma carpeta del ejecutable del proyecto.", "Aviso:",
 MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

114

La siguiente imagen muestra su ubicación en el programa.

```

    pictureBox_D7.Image = Resources.green_off;
}
static void Alerta_1()
{
    MessageBox.Show(@"No se encuentra la biblioteca 'input32.dll', debería encontrarse en la misma carpeta del ejecutable del proyecto")
}

```

Creamos un procedimiento llamado Opciones, el cual permite cargar los datos que se envían al puerto a la vez de que se intercambian los objetos de imagen para simular el envío de datos (encendido o apagado de LED's) en la pantalla. Como se puede apreciar en la imagen inferior, el módulo se compone de bloques if-else, uno por cada bit (o led) *D1*, *D2*, ..., *D7*.

```

    MessageBox.Show(@"No se encuentra la biblioteca input32.dll , debería encontrarse en la misma carpeta del ejecutable del proyecto");
}

//region es una instrucción utilitaria que permite segmentar por campos el código
#region Opciones
public void opciones() //Las sentencias en if deben repetirse para cada LED (Dx), x=0,1,...,7
{
    int value = 0;
    if (D0 == 1) //Si D0 se ha puesto a uno (ON) hacer:
    {
        value += (int)Math.Pow(2, 0); //suma el contenido de value con 2^0
        LoadNewPict_D0(); //se llama a un procedimiento para actualizar la imagen y simular la acción
    }
    else
    {
        LoadOldPict_D0(); //con otro procedimiento se deja la imagen anterior a la acción
        value += 0; //Se suma cero a value, es decir no cambia su valor
    }

    if (D1 == 1) //Si D1 se ha puesto a uno (ON) hacer:
    {
        value += (int)Math.Pow(2, 1); //suma el contenido de value con 2^1
        LoadNewPict_D1(); //se llama a un procedimiento para actualizar la imagen y simular la acción
    }
    else
    {
        LoadOldPict_D1(); //con otro procedimiento se deja la imagen anterior a la acción
        value += 0; //etc...
    }
}

```

Hacia el final del módulo agregamos el bloque intentará enviar datos al puerto y configuraremos la excepción que puede ocurrir por la falta de la biblioteca necesaria para el proceso.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```

else
{
    LoadOldPict_D7();
    value += 0;
}
//intentamos enviar el dato al puerto y cachamos la posible excepción
try
{
    Puerto_Paralelo_Basico_C.PortInterop.Output(888, value);
}
catch (DllNotFoundException)
{
    Alerta_1();
}
//marcamos el fin de la región en la siguiente línea
#endifregion

```

Agregamos los procedimientos que permitirán a nuestro programa intercambiar las imágenes de encendido y apagado. Mostramos sólo los primeros, los demás nuevamente se hacen para cada Dx, x=0, 1, ..., 7.

```

#region Procedimientos para intercambio de imágenes LED
private void LoadNewPict_D0() //Procedimiento que carga gif LED encendido para D0
{
    pictureBox_D0.Image = Resources.green_on;
}

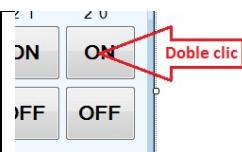
private void LoadOldPict_D0() //Procedimiento que carga gif LED apagado para D0
{
    pictureBox_D0.Image = Resources.green_off;
}

private void LoadNewPict_D1() //gif LED encendido para D1
{
    pictureBox_D1.Image = Resources.green_on;
}

private void LoadOldPict_D1() //gif LED apagado para D1
{
    pictureBox_D1.Image = Resources.green_off;
}

```

Para codificar los botones iremos a la vista de formulario y daremos doble clic sobre el botón a programar y a continuación se cambiará la vista al código donde se nos colocará en el espacio donde deberán ir las instrucciones.





Mostramos los códigos para el ON y OFF del bit 0, códigos similares deben insertarse para cada botón ON y OFF para el bit o LED correspondiente.

```
private void button_D0_ON_Click(object sender, EventArgs e) // este código se inserta automáticamente
{
    D0 = 1;           //Se pone a 1 el bit 0 del puerto
    opciones();       //Se invoca el procedimiento de opciones para encender el gif LED y enviar el dato
}

private void button_D0_OFF_Click(object sender, EventArgs e)
{
    D0 = 0;           //Se pone a 0 el bit 0 del puerto
    opciones();       //Se intercambia el gif LED por el de apagado.
}
```

116

El *TextBox* de la dirección sólo puede tener dos valores en decimal 888 (correspondiente a la dirección 378H) o 632 (278H). Programamos el botón Dirección con tales opciones.

```
private void button_Address_Click(object sender, EventArgs e)
{
    if (textBox_port_address.Text == "378")
    {
        adress = 888;
    }
    else
    {
        adress = 632;
    }
}
```

El botón ‘Enviar’ nos permitirá enviar valores entre 0 y 255 que se pueden escribir en el *TextBox* byte. Agregamos el código correspondiente para tal fin.

```
private void button_Enviar_Click(object sender, EventArgs e)
{
    try
    {
        Puerto_Paralelo_Basico_C.PortInterop.Output(adress, Int32.Parse(textBox_byte.Text));
    }
    catch (DllNotFoundException)
    {
        Alerta_1();
    }
}
```

Codificamos el botón Reset inicializando las variables a 0.



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```
private void button_Reset_Leds_Click(object sender, EventArgs e)
{
    i = 0; j = 0;
    Reset_LEDs();
    D0 = 0; D1 = 0; D2 = 0; D3 = 0; D4 = 0; D5 = 0; D6 = 0; D7 = 0;
}
```

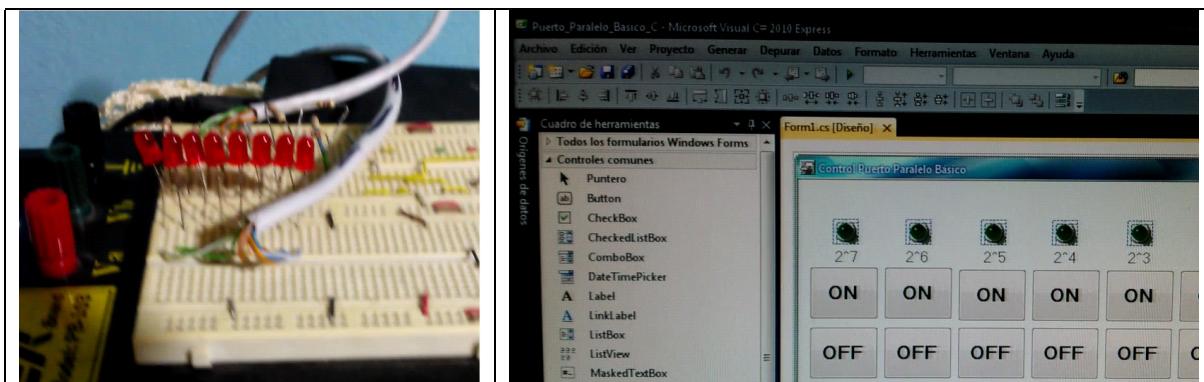
117

Por último, programamos el botón que nos permitirá salir de la aplicación, agregando las siguientes instrucciones.

```
private void button_Salir_Click(object sender, EventArgs e)
{
    System.Windows.Forms.Application.Exit();
}
```

8.3.4. Prueba del programa

Para presentar la práctica, se montan los LED's según se mencionó al principio de ésta práctica en un protoboard y se conecta la terminal DB25 al puerto paralelo de una computadora (en nuestro caso, las pruebas se hicieron en una PC ensamblada AMD Athlon X2 64 Bits con Windows 7 Ultimate).



Ejecutamos el programa y podremos probar a dar clic a los botones de 'ON', verificar que en la ventana los LED involucrados se iluminan por efecto del cambio de imagen gif, al tiempo que en el protoboard ocurre algo similar encendiendo el LED correspondiente.



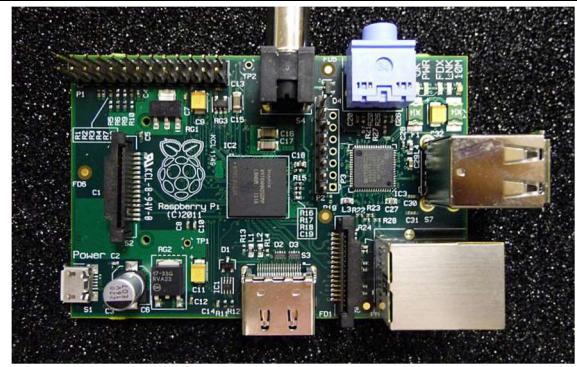
"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx

8.4. Práctica Propuesta 2: Encendido de LED's en un Raspberry Pi

El Raspberry Pi es una computadora embebida en una placa reducida (o de placa única) de bajo coste desarrollado en el Reino Unido con el objetivo de estimular la enseñanza de las ciencias de la Computación en las escuelas.

La fundación Raspberry Pi mantiene el control de los derechos de la arquitectura, pero permite su uso libre a nivel educativo y particular.¹¹



118

Su diseño llamado ‘System-on-a-chip’ (BCM2835, BCM2836, BCM2837) posee procesadores ARM11, ARM Cortex A7 y ARMv8, que van de los 700MHz hasta los 1.2 GHz, con núcleo simple a cuádruple, siendo los tres primeros procesadores de 32 bits y él último de 64. La RAM implementada es de 256MB, 512MB y de 1GB, dependiendo de la versión de la tarjeta. Posee puertos de bajo nivel de diversa norma: GPIO, SPI, I²C, UART. El sistema operativo que promueve la fundación es un derivado de GNU/Linux Debian llamado Raspbian, pero soporta derivados de otras distribuciones: Pidora de Fedora, Arch Linux ARM de Arch Linux. Emplea como disco duro tarjetas de memoria SD (Raspberry Pi B) y microSD (Raspberry Pi B+ y posteriores).

En el ámbito que nos atañe que es la programación, podemos decir que se promueve el uso de los lenguajes de programación de filosofía libre, fundamentalmente Python, aunque soporta la programación en C, Tiny BASIC, Perl, Ruby, y recientemente, mediante un acuerdo con Oracle se incluye una máquina virtual para programar en Java.

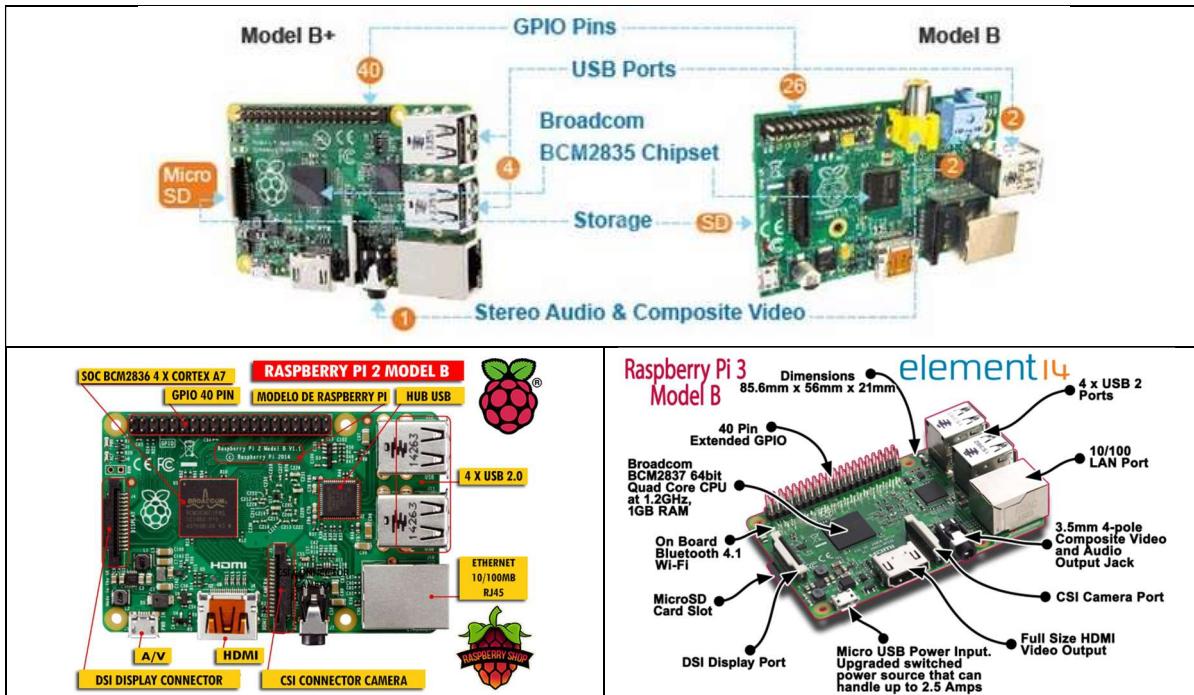


8.4.1. Estructura básica de la Raspberry Pi

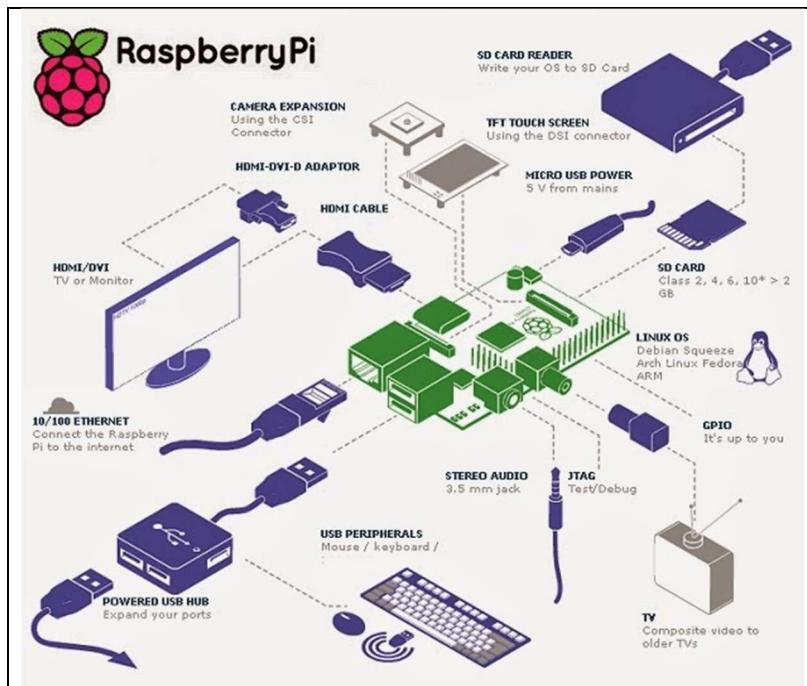
En el mercado mexicano se ofertan actualmente (junio de 2016) los modelos B, B+, Pi 2 y Pi 3, cuyas diferencias internas implican la distinta capacidad en RAM, velocidad de procesamiento, núcleos del microprocesador y la arquitectura de 32 o 64 bits. Aparte de éstas diferencias tenemos las que conciernen al hardware que se puede apreciar en cada tarjeta.

¹¹ Wikipedia.org. (2016). *Raspberry Pi*. Obtenido de: https://es.wikipedia.org/wiki/Raspberry_Pi





Para un funcionamiento básico de la Raspberry Pi requeriremos de algunos dispositivos externos que se pueden ofertan en Kit's armados por las tiendas, o bien adquirirlos por separado.



En nuestro caso requeriremos de un adaptador de voltaje de 120VAC a 5VDC con al menos 2A de corriente; en la mayoría de los casos emplearemos un adaptador de HDMI a VGA, que nos permita utilizar los monitores más ampliamente distribuidos; un teclado; un mouse y opcionalmente tarjeta de red inalámbrica USB. Si elegimos los modelos anteriores al B+, deberemos agregar también un Hub USB con alimentación propia a 2A de corriente.

120

Evidentemente debemos contar con una tarjeta SD o microSD dependiendo del modelo de Raspberry disponible, sobre él instalaremos Raspbian que es el que usaremos. Las instrucciones para la descarga del software y su instalación las puede hallar en:

<https://www.raspberrypi.org/downloads/>

8.4.2. Pasos Previos

Un requerimiento necesario al usar Raspberry's es conocer los comandos básico usados en Raspbian desde la terminal (el equivalente de una consola de Windows o línea de sistema), la cual se puede encontrar en la barra de tareas del sistema o buscar en el menú de programas como *LXTerminal*. La terminal de Raspbian, como en casi toda distribución de Linux, es importante porque permite realizar tareas especializadas como la instalación y compilación de aplicaciones

En la ventana que se abre verá un fondo comúnmente negro con algún texto similar al que se muestra en la siguiente figura.

```
pi@raspberrypi ~ $
```

La línea de caracteres mostrados es llamada ‘prompt’ del terminal y provee de cierta información:

- ‘pi’ es el nombre del usuario.
- ‘raspberrypi’ es el nombre de la computadora.
- El símbolo ‘~’ es el directorio en el que se encuentra actualmente, home.
- El símbolo ‘\$’ es el indicador prompt de la terminal, todo comando se escribirá a la derecha de este símbolo. Éste símbolo cambia según se tengan privilegios '#', o no se tengan privilegios '\$'.

En la línea de comandos y con guía de su docente practicará la siguiente lista de comandos:

Comando	Acción
ls	Enlista las carpetas y archivos en la carpeta actual
cd	Accede a una carpeta
mkdir	Crea una carpeta
rm	Borra un archivo
rm -rf	Borra una carpeta



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



df -h	Espacio libre e los dispositivos de almacenamiento
uname -a	Información del sistema operativo
sudo	Ejecución de comandos con privilegios
reboot	Reinicia a computadora
halt	Apaga la computadora

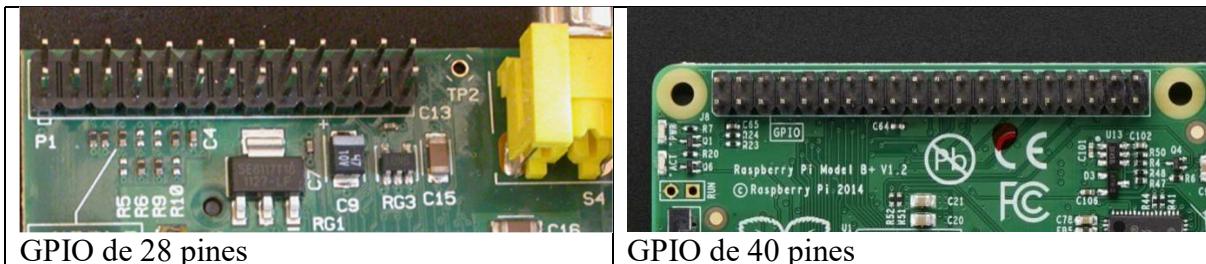
Otra acción relevante es la actualización del sistema operativo y de las aplicaciones instaladas en torno a él, para lo cual será necesario contar con acceso a internet y realizar los pasos siguientes:

- | | | |
|------|-------------------------------|----------------------------|
| i. | Actualizamos repositorios: | \$ sudo apt-get -y update |
| ii. | Actualizamos los programas: | \$ sudo apt-get -y upgrade |
| iii. | Actualizar el kernel: | \$ sudo rpi-update |
| iv. | Verificamos la actualización: | \$ uname -r |

Una nota pertinente es el hecho de que la práctica a realizar involucra el uso del lenguaje de programación Python, cuyo entendimiento no debería ser complicado, toda vez que se cuenta de la experiencia adquirida a lo largo de éste curso, pero que sin embargo, tal vez merezca una introducción previa que debería ser impartida por el docente. Involucrar esa actividad en este documento sale de los fines que persigue éste, por lo que no se incluye al momento de producir ésta versión, pensándose agregarlo como un anexo en un futuro próximo.

8.4.3. Terminales GPIO

Las terminales de entrada/salida de propósito general (GPIO, General Purpose Inputs/Outputs) del Raspberry permiten a éste interactuar con el exterior. Existen dos versiones de éstas terminales siendo de 28 pines para los modelos A y B, mientras que el modelo B+ y el Pi 2 tienen cuarenta pines.



Cuentan con dos clases de nomenclatura: RasPi y BMC2835, los cuales son usados en justo esos términos por los programas creados, por lo que es importante elegir adecuadamente cuál usar, ya que esto se refleja en la o las bibliotecas requeridas para el control del puerto.



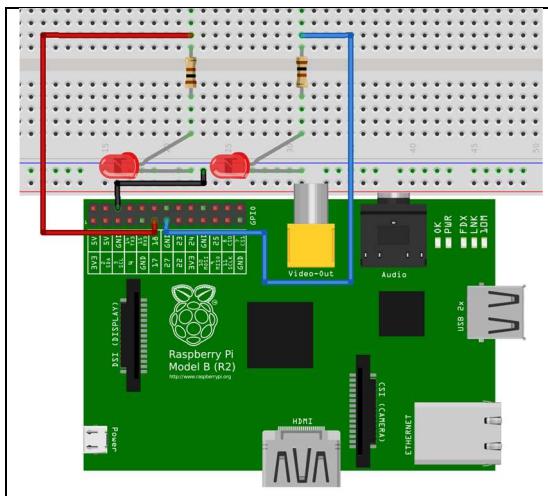


GPIO Numbers		
Raspberry Pi B Rev 1 P1 GPIO Header	Raspberry Pi A/B Rev 2 P1 GPIO Header	Raspberry Pi B+ B+ J8 GPIO Header
Pin No.	Pin No.	Pin No.
3.3V 1 2 5V	3.3V 1 2 5V	3.3V 1 2 5V
GPIO0 3 4 5V	GPIO2 3 4 5V	GPIO2 3 4 5V
GPIO1 5 6 GND	GPIO3 5 6 GND	GPIO3 5 6 GND
GPIO4 7 8 GPIO14	GPIO4 7 8 GPIO14	GPIO4 7 8 GPIO14
GND 9 10 GPIO15	GND 9 10 GPIO15	GND 9 10 GPIO15
GPIO17 11 12 GPIO18	GPIO17 11 12 GPIO18	GPIO17 11 12 GPIO18
GPIO21 13 14 GND	GPIO27 13 14 GND	GPIO27 13 14 GND
GPIO22 15 16 GPIO23	GPIO22 15 16 GPIO23	GPIO22 15 16 GPIO23
3.3V 17 18 GPIO24	3.3V 17 18 GPIO24	3.3V 17 18 GPIO24
GPIO10 19 20 GND	GPIO10 19 20 GND	GPIO10 19 20 GND
GPIO9 21 22 GPIO25	GPIO9 21 22 GPIO25	GPIO9 21 22 GPIO25
GPIO11 23 24 GPIO8	GPIO11 23 24 GPIO8	GPIO11 23 24 GPIO8
GND 25 26 GPIO7	GND 25 26 GPIO7	GND 25 26 GPIO7
Key		
Power +	UART	
GND	SPI	
I²C	GPIO	

Como puede apreciarse en la figura anterior algunos pines de éste puerto tienen utilidades especiales, aunque pueden usarse para otras finalidades siempre que se les reconfigure.

8.4.4. Control de LED's en Python

Realizaremos el arreglo mostrado en la imagen siguiente con el Raspberry Pi apagado y nuestro protoboard. Utilizaremos las terminales GPIO 17 y el 27, el arreglo es aplicable para los modelos de 28 y 40 pines. Las resistencias pueden ser de entre 150 y 220 Ohms.



A continuación, abrimos una terminal y comprobaremos si tenemos instalada la biblioteca que controla los GPIO mediante Python mediante los siguientes comandos:

```
#sudo python
>>import RPi.GPIO as GPIO
>>
```



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx





Si el sistema no marca ningún error entonces la biblioteca se encuentra instalada.

```
pi@raspiehg ~ $ sudo python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
>>> 
```

Salga de la consola de Python con: >>exit()

123

De no tener instalada la biblioteca, la descargamos de internet:

wget 'http://downloads.sourceforge.net/project/raspberry-gpio-python/RPi.GPIO-0.5.4.tar.gz'

Luego:

- Descomprimimos: tar zxvf RPi.GPIO-0.5.4.tar.gz
- Entramos a la biblioteca creada: cd RPi.GPIO-0.5.4/
- Para instalar requeriremos el paquete dev de Python: sudo apt-get install python-dev
- Instalamos la biblioteca: sudo python setup.py install

Para crear el programa usaremos el editor de texto básico llamado *nano*, ejecutando en la línea de comandos:

sudo nano blink.py

En este caso ‘blink’ es el nombre formal del archivo que contendrá el script, la extensión ‘.py’ es la que corresponde a cualquier script que pueda ser ejecutado por la máquina virtual Python. Agregue al archivo el siguiente texto:

```
import RPi.GPIO as GPIO ## Se importa la biblioteca Python de control GPIO
import time ##Se importa una función de retardo

GPIO.setmode(GPIO.BCM) ## Se indica la nomenclatura que se va a usar para los pines
GPIO.setup(17, GPIO.OUT) ## GPIO 17 como salida
GPIO.setup(27, GPIO.OUT) ## GPIO 27 como salida

def blink(): ## Definimos un módulo para el parpadeo de los leds
    print "Ejecucion iniciada..." ## Se imprime un mensaje en terminal para indicar el
    inicio
    iteracion = 0 ## Se crea e inicializa una variables de conteo
    while iteracion < 30: ## Ciclo para la duración de la función
        GPIO.output(17, True) ## Se enciende el LED en el pin 17
        GPIO.output(27, False) ## Se apaga el LED en el pin 27
        time.sleep(1) ## Se realiza un retardo de 1 segundo
        GPIO.output(17, False) ## Se apaga el LED en el pin 17
        GPIO.output(27, True) ## Se enciende el LED en el pin27
        time.sleep(1) ## Esperamos 1 segundo
```



“HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL”

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



```

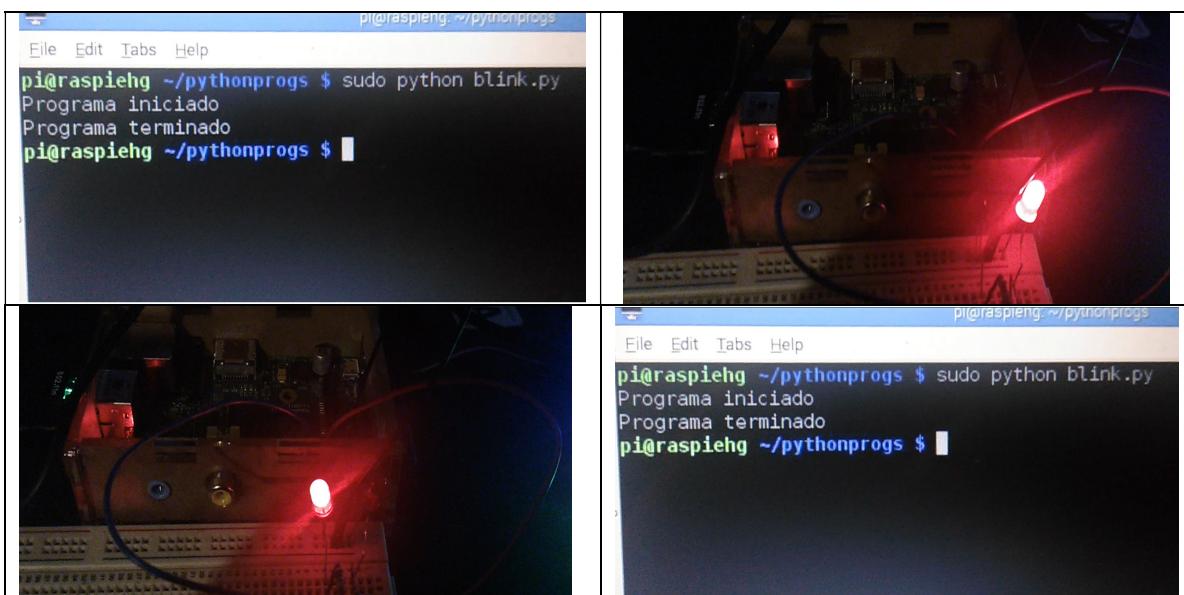
iteracion = iteracion + 2    ## Incrementos de iterador en pares
print "Ejecucion finalizada"  ## Se imprime en terminal mensaje de fin de
ejecución
GPIO.cleanup()                ## Se limpian los GPIO
blink()                       ## Se invoca a la función creada

```

124

Guardamos el archivo y salimos del editor.

Para ejecutar el archivo escribimos en la terminal de comandos: \$ sudo python blink.py
En las siguientes imágenes mostramos la ejecución de las instrucciones mencionadas.



Con la prueba del programa queda concluida ésta práctica y nuestro curso de Programación Básica.



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx



CACEI

CACEB A.C.

COMEXAA

Bibliografía

- G. Levine G., Introducción a las computadoras y programación estructurada, Ed. Mc Graw Hill.
- Peter Norton, Introducción a la computación, Ed. Mc. Graw Hill.
- Robert Sedgewick, Algorithms in C++, Ed. Addison Wesley.
- Wikipedia, la Enciclopedia Libre (2016). Consultado: 20/06/2016. Disponible: <https://es.wikipedia.org>
- Tema 1. Elementos de las gráficas por computadoras. (2010, Abril 12). Recuperado octubre 27, 2013, desde OpenCourseWare de la Universidad Anáhuac México Norte, Web site: http://educommons.anahuac.mx:8080/eduCommons/computacion-y-sistemas/programacion-de-graficas-computacionales/tema-1-1/aprendizaje_motivacion
- Programming Tutorials and Lecture Notes. (16 de 07 de 2009). Recuperado el 29 de 10 de 2013, de <http://chortle.ccsu.edu/Bloodshed/howToGL.html>
- *Programming Tutorials and Lecture Notes*. (16 de 07 de 2009). Recuperado el 29 de 10 de 2013, de <http://chortle.ccsu.edu/Bloodshed/howToGL.html>
- Sistemas Adaptativos y Bioinspirados en Inteligencia Artificial. (07/05/2016 de 05 de 2016). *Tutorial OpenGL*. Obtenido de <http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial/cap3.htm>

125



"HACIA LA EXCELENCIA, CON
CALIDEZ HUMANA Y CALIDAD INTEGRAL"

Carretera Acuaco-Zacapoaxtla, km 8, Colonia Totoltepec, C.P. 73680, Zacapoaxtla, Puebla.
Tel/Fax. 01 (233) 317 50 00, e-mail: tecnologico@itsz.edu.mx
www.itsz.edu.mx