



UNIVERSIDAD TÉCNICA LUIS VARGAS TORRES DE
ESMERALDAS

FORMATO INSTITUCIONAL DE GUION:

ASIGNATURA:
FUNDAMENTOS DE
PROGRAMACIÓN

Esmeraldas - Ecuador
2020

Índice

1. DATOS INFORMATIVOS DE LA ASIGNATURA	4
1.1. Datos Generales	4
1.2. JUSTIFICACIÓN:	4
1.3. PROBLEMA DE LA PROFESION:	5
1.4. OBJETO DE ESTUDIO	5
2. OBJETIVOS	6
2.1. Objetivo General	6
2.2. Resultados de aprendizaje:	6
3. CONTENIDOS	7
4. PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS	8
4.1. UNIDAD # 0: Presentación, motivación y diagnostico	9
4.1.1. Presentación del docente	9
4.1.2. Presentación de los estudiantes	9
4.1.3. Presentación de la asignatura	9
4.2. Unidad # 1: Introducción a las computadoras y los lenguajes de programación	10
4.2.1. Introduccion a las computadoras	10
4.2.2. Recursos y actividades de aprendizaje	11
4.2.3. Preguntas de autocontrol	11
Referencias	14
4.2.4. Arquitectura de la computadora	16
4.2.5. Sistema de numeración	17
4.2.6. Los lenguaje de programación	17
4.2.7. Preguntas de autocontrol	19
4.2.8. Introducción a Linux y termux	21
4.2.9. Comando/Paquetes de linux: Instalación, configuración y prácticas de uso	22
4.2.10. Taller: Utilización de termux en la programación	23
4.2.11. Preguntas de autocontrol	27

4.2.12. Introducción a Vim y sus comandos	29
5. Vim en modo normal	31
5.0.1. Taller uso de vim: Creación y edición de archivos	33
5.0.2. Preguntas para el autocontrol	38
5.0.3. Introducción a la programación	41
6. Ciclo de vida del Software	41
7. Creación del Software: Principios básicos.	41
7.1. El análisis y diseño	41
7.2. La etapa de la codificación	42
7.3. La etapa de compilación y ejecución	42
7.4. La etapa de verificación y depuración	43
7.5. La etapa de mantenimiento	43
7.6. Documentación	44
7.6.1. Taller de programación básica 1	45
7.6.2. Taller de programación 2: Elaboración de informe	45
7.6.3. Figuras para el diagrama de flujo	48
7.7. Inicio/fin	49
7.8. Símbolo de Proceso	49
7.9. Símbolo de decisión	49
7.10. Símbolo de entrada y salida	50
7.10.1. Taller de Diagrama de flujo	50
8. Resolviendo un problema muy simple	50
8.1. Análisis del problema	51
8.2. Mejorando nuestro algoritmo utilizando pseudo-código	51
8.3. El diagrama de flujo para la resta de dos números	52
9. El clásico programa del punto de venta	54
9.1. Operaciones matemáticas	54
9.2. Operaciones lógicas	54
9.3. Preguntas de autocontrol	56

9.4. Diagrama de Flujo (Descisiones)	58
9.5. Diagrama de Flujo (Estructura de repetición)	58
9.5.1. Taller de Diagrama de Flujo	59
10. Un programa que cuenta y suma los número pares e impares.	59
11. Problemas propuestos.	60
11.0.1. Preguntas para el autocontrol	62
11.0.2. Estructura básica de un programa en C++	65
12. Elementos básicos de un programa en c++	66
13. Identificadores	66
14. Bloques	67
15. Las bibliotecas de C++	70
16. La directiva	70
17. Espacio de nombres	71
17.1. Personalizando el espacio de nombre	72
18. Datos, tipos de datos y operaciones primitivas	73
18.0.1. Estructura de selección (if-else) en C++	83
18.1. Estructura de repetición (do-while)	87
18.1.1. Taller para estructura de repetición en la programación de C++	88
18.1.2. Proyecto Final Integrador	93
19. Funciones declaradas por el usuario	100
19.1. Clases: Estructura	105
19.2. Preguntas de autocontrol	108
19.3. Formas y tipos de evaluación	112
19.4. Rubricas para autoevaluación del silabo	113

1. DATOS INFORMATIVOS DE LA ASIGNATURA**1.1. Datos Generales**

Facultad	:	Ingenierías
Carrera	:	Ingeniería en Tecnología de la Información y Comunicación
Asignatura	:	Fundamentos de Programación
Código	:	ITICI1104
Nivel Académico	:	A-B
No. Créditos	:	6
Unidad de formación	:	Profesional
Prerrequisito (s)	:	Nivelación
Correquisitos	:	Física, Análisis matemático
Componentes de aprendizaje	Aprendizaje Asido por el profesor	64 horas
	Aprendizaje colaborativo	
	Prácticas de aplicación y colaboración	32 horas
	Aprendizaje autónomo	64 horas
Total horas de aprendizaje		160 horas
Periodo Académico	:	Septiembre 2021 – Enero 2022
Profesor	:	Ing. Stalin Francis M.sc
Título cuarto nivel	:	Magister en Ciencias de la Computación.
Email	:	stalin.francis@utelvt.edu.ec
Teléfono	:	0997919650

1.2. JUSTIFICACIÓN:

Dentro de la formación de un ingeniero la planificación lógica y secuencial de actividades es una tarea indispensable a la hora de querer realizar una tarea que resuelva un problema;

por eso la programación es una tarea que el ser humano ha realizado mucho antes que las computadoras existieran como actualmente la conocemos.

Las computadoras permiten mantener registradas actividades y muchas de ellas ejecutarlas de forma automática liberando al hombre de la carga de llevar al control con el riesgo de no cumplir con los tiempos planificados.

La asignatura de Fundamentos de programación que se dicta en el primer semestre de la carrera de Ingeniería en Tecnología de la Información, brinda al estudiante la habilidad de analizar problemas y luego diseñar su solución computacional utilizando los diagramas de flujo que sirvan para llevarlos a un programa de computador.

1.3. PROBLEMA DE LA PROFESION:

La asignatura de Fundamentos de Programación, trata sobre el diseño e implementación de problemas de computador, su importancia se debe a que desarrolla en el profesional y el estudiante, pensamientos lógicos, analíticos, creativos y el ingenio; características indispensables en el perfil del ingeniero de sistemas. De esta manera se logrará incluirlos en la sociedad como profesionales proactivos. Sentará de manera adecuada las bases para la resolución de problemas de diversas áreas del conocimiento, a través del desarrollo de la lógica de programación.

1.4. OBJETO DE ESTUDIO

Problemas de matemática y de lógica resolubles utilizando el lenguaje C++.

2. OBJETIVOS

2.1. Objetivo General

Desarrollar destrezas y habilidades en los estudiantes para analizar, diseñar y crear soluciones a problemas de matemática, física a través del diagrama de flujo y la programación en C++.

2.2. Resultados de aprendizaje:

1. El estudiante podrá identificar las diferentes partes constitutivas de un ordenador e imaginar cual es su funcionamiento en el momento que entre en ejecución un programa.
2. El estudiante tendrá los conocimientos, habilidades y destrezas para utilizar el sistema operativo Android de su Smartphone para programar en C++.
3. El estudiante podrá analizar un problema matemático para diseñarlo utilizando los diagramas de flujo.
4. El estudiante tendrá los conocimientos, habilidades y destrezas para crear un conjunto de instrucciones en C++ bien estructurada.
5. El estudiante podrá analizar un problema matemático y lógico para diseñarlo utilizando estructura de selección en los diagramas de flujo.
6. El estudiante podrá analizar un problema matemático y lógico para diseñarlo utilizando estructura de repetición en los diagramas de flujo.
7. El estudiante podrá crear un programa utilizando funciones almacenadas en librerías personales.

3. CONTENIDOS

No	Unidades	Componentes						Total Horas	
		Docencia				Práctica Experimental			
		C	L	S	E	CP	TA		
0	Presentación,motivación, diagnóstico (10 - 14 Agosto 2020)	4						4	
1	Introducción a las computadoras y los lenguajes de programación	4	4				8	16	
2	Nociones de linux, vim, clang	4				8	12	24	
3	Metodología de la programación y Diagra- ma de flujo	4	2			12	16	34	
	Semana de evaluación sumativa				6			6	
	Total parcial	16	6		6	20	36		84
4	Programación en C++: Introducción	2				4	4	10	
5	Flujo de control I: Estructura selecti- vas(cap. 4)	4				8	8	20	
6	Flujo de control I: estructura repetitiva(cap. 5)	4				8	8	20	
7	Funciones(cap. 6) y librerías personales	4			16	8	8	36	
	Ex amen 2do Parcial (23 -27 noviembre 2020)				6			6	
	Total parcial	14			6	28	28		76
TOTAL HORAS POR TIPO DE CLASE:		30	6		12	48	64		160

Cuadro 1: Tipo de clases: C:Conferencia L: Lecciones Oral S: Seminario, CP: Clases Prácticas, TA: Taller

4 PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS

4. PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
PRIMERA SEMANA	30-agosto-2021	09-septiembre-2021	2		C	<p>SESIÓN 1 - (01-septiembre-2021)</p> <p>Presentación del docente y de los estudiantes.</p>	<ul style="list-style-type: none"> ✓ Palabras de bienvenida del docente. ✓ Presentación de los estudiantes. ✓ Análisis del silabo y metodología de trabajo.
			2		C	<p>SESIÓN 2</p> <p>Presentación de la asignatura y uso de la plataforma Classroom.</p>	<ul style="list-style-type: none"> ✓ El docente explica como los estudiantes deben configurar y utilizar la plataforma Classroom.
			2	2	C	<p>SESIÓN 3</p> <p>INTRODUCCIÓN A LAS COMPUTADORAS. (Aguilar, 2008, cap.1) (Francis, 2020, cap. 1) (Youtube, 1999)</p>	<ul style="list-style-type: none"> ✓ Docente dicta una charla magistral del tema. ✓ Test de diagnóstico de conocimiento previos. ✓ se envía la Actividad B1
			6	2			

4.1. UNIDAD # 0: Presentación, motivación y diagnostico

En esta etapa el docente facilitará a los estudiantes toda la información general necesaria para comenzar el curso, esta información comprende datos de contacto así como una descripción de la asignatura (silabo), metodologías y recursos a utilizar para asegurar el éxito del proceso de aprendizaje.

4.1.1. Presentación del docente

El docente tendra que proveer datos de contacto como:

- ✓ Nombres y apellidos.
- ✓ Correo electrónico.
- ✓ Número de telefono.
- ✓ Dirección del sito web personal

4.1.2. Presentación de los estudiantes

Cada estudiantes en la videoconferencias, hara una pequeña intervención para que los demas compañeros los conozcan en la cual daran los siguientes datos:

- ✓ Nombres y apellidos.
- ✓ Colegio del cual proviene.
- ✓ Especialidad en la cual se graduo.
- ✓ ¿Por que decidio escoger la carrera de Ingeniería en Tecnología de la Información.?
- ✓ ¿Qué espera aprender en la carrera?

4.1.3. Presentación de la asignatura

En esta etapa se dara el nombre de la asignatura y una explicación introductoria de lo que se va a enseñar. Se analizará cada sección del silabo, haciendo énfasis en los contenidos mínimos, las actividades de aprendizaje y la herramienta Classroom y MOODLE que se va a utilizar durante el proceso

4.2. Unidad # 1: Introducción a las computadoras y los lenguajes de programación**4.2.1. Introduccion a las computadoras**

El computador es uno de los mayores inventos que ha realizado el hombre, varios fueron las personas a las que se le atribuye este maravilloso invento entre ellos Jhon Von Neumann.

Año	CIENTÍFICO	CONTRIBUCIÓN
1833	Charles Babbage	Diseña e intento construir la primera computadora (Máquina analítica)
1890	Herman Hollerith	Inventa la máquina tabuladora utilizada para el censo de Estados Unidos, Fundo la IBM.
1936	Alan Turing	Formalizo los conceptos de algoritmo y de máquina de Turing, la clave de la computadore moderna.
1944	En inglaterra	Se construye la computadora Colossus(Colossus Mark I y Colossus Mark 2), para descifrar comunicaciones de los alemanes en la Segunda Guerra Mundial.
1947	En la Universidad de Pensilvania	Se contruye ENIAC(Elcetronic Numerical Integrador And Calculator), funciona con válvulas y fue la primera computadora electrónica de propósito general.
1951	Comienza a operar EDVAC, concebida por John Von Neumann	ENIAC no era decimal, sino binaria, y tuvo el primer programa diseñado para ser almacenado.
1951	J. Presper Eckert y John William Mauchly	Crearon UNIVAC I que fue la primera computadora electrónica comercial en los estadosunidos.

4 PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS

1947	John Bardeen, Walter Brattain, William Shockley de los laboratorio Bell	Inventan el transistor que reemplaza al tubo del vacío
1957	Robert Noyce y Jack Kilby	Inventaron el circuito integrado o microchip, motor de la revolución de los ordenadores personales.
1971	Intel	Presenta el primer procesador comercial, el primer chip: el microprocesador intel 4004 .
1973	Xerox	Desarrollo el Xerox PARC que fue uno de los primeros ordenadores personales de la historia.
1975	Bill Gates y Paul Allen	Fundadores de la empresa Microsoft
1976	Steve Jobs, Steve Wozniak, Mike Markkula	Fundadores de la empresa Apple
1977	Apple	Presenta el primer computador personal que se vende a gran escala Apple II .
1981	IBM	Lanza al mercado la IBM PC que se convierte en un éxito comercial.
1983	Microsoft	presenta el sistema operativo MS-DOS, por encargo de IBM.

4.2.2. Recursos y actividades de aprendizaje

“<https://www.youtube.com/watch?v=xKka6kzTQgw&t=822s>”

4.2.3. Preguntas de autocontrol

× **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.

✓ char babage.

✓ Jhon Vom Neumann.

4 PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS

- ✓ Rober Noise.
- × **Pregunta # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **Pregunta # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **Pregunta # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

4 PROGRAMACIÓN ANALÍTICA DE APRENDIZAJE Y ACTIVIDADES BÁSICAS

Actividad	Criterio	Nivel 1		
<div>Actividad B1:</div> <p>Leer críticamente el capítulo 1 del libro guía y elaborar un video donde utilizando la metodología de la exposición, explique lo que comprendio.</p>	Conocimiento y preparación del tema	Deficiente: Demuestra falta de conocimientos del tema. La información dada es irrelevante.	Bueno: Demuestra confianza en conocimientos, pero falla en algunos momentos tratar ofrecer la información más precisa.	Excelente: Demuestra solvencia, confianza al expresar conocimientos, presentando información precisa y pertinente para desarrollo del tema.
		0-10	20	30
	Expresión de un punto de vista personal	Deficiente: Expresa ideas incoherentes respecto del tema de la exposición.	Bueno: Argumenta ideas a partir de conocimientos válidos sobre tema elegido, aunque no logra sostenerse en idea central.	Excelente: Argumenta ideas a partir de conocimientos válidos sobre el tema elegido, así como el énfasis en las ideas centrales.
		0-10	20	30
	Estructura y orden	Deficiente: Ofrece una exposición carente de orden o cuidado por la organización del tema.	Bueno: La exposición es organizada de manera adecuada, aunque sin terminar en tiempo establecido y dejando ideas sueltas.	Excelente: Ofrece exposición muy organizada, respeta tiempos, facilita la captación de su discurso desde el inicio hasta el final.
		0-10	20	40

Referencias

- Aguilar, L. J. (2008). Fundamentos de programación: algoritmos, estructura de datos y objetos. McGraw-Hill.
- Bronson, G. (2007). C++ para ingeniería y ciencias. G. Borse, Lehigh University.
- Francis, S. (2020). Fundamentos de programación: Desarrollo en c++ utilizando dispositivos móviles. EDUCAYSOFT.
- Sierra, F. J. C. (1998). Programación orientada a objetos con c++. RA-MA Editor.
- Youtube. (1999). La historia de la computadora y computacion - documental completo.
url<https://www.youtube.com/watch?v=7eOKcLnm0Xo>.
- Youtube. (2014). Estructura de computadores - unidad 1: Introducción - José Luis Abellán.
url<https://www.youtube.com/watch?v=6EPsiLs8HPM>.
- Youtube. (2015). Estructura de computadores - unidad 2: Introducción - José Luis Abellán.
url<https://www.youtube.com/watch?v=6EPsiLs8HPM>.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
SEGUNDA SEMANA	06-septiembre-2021	10-septiembre-2021	2	2	C	ARQUITECTURA DEL COMPUTADOR. (Aguilar, 2008, cap.1) (Francis, 2020, cap. 1) (Youtube, 2014) (Youtube, 2015)	SESIÓN 4 ✓ El docente en una videoconferencia refuerza el tema. ✓ Evaluación sobre la Tarea B1.
			2	2	C-L	SISTEMA DE NUMERACIÓN. (Aguilar, 2008, cap.1) (Francis, 2020, cap. 1)	SESIÓN 5 ✓ El docente explicará los diferentes sistemas de numeración. ✓ Evaluación arquitectura del computador.
			2	2	C-L	LOS LENGUAJES DE PROGRAMACIÓN. (Aguilar, 2008, cap.1) (Francis, 2020, cap. 1)	SESIÓN 6 ✓ El docente en videoconferencia explicará sobre los diferentes lenguajes de programación. ✓ Evaluación de los sistemas de numeración.
	6	6					

4.2.4. Arquitectura de la computadora

Todos los sistemas digitales de procesamiento de información incluyendo a los smartphones, comparten la misma arquitectura del ordenador propuesta por John Von Newman en el documento "First Draft of Report on the EDVAC" escrito en 1945.

En esta propuesta Newman describe los elementos que participan en el procesamiento de la información desde el momento que los datos ingresan al ordenador hasta que salen como información.

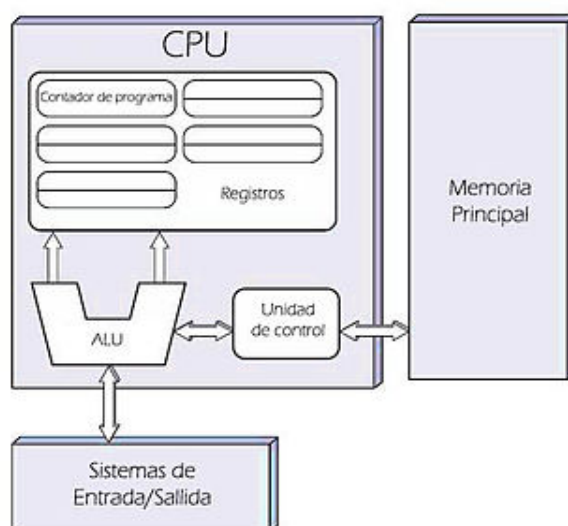


Figura 1: Arquitectura Neuman

Los ordenadores deben permitir recibir los datos a través de los dispositivos de entrada, y luego almacenarlos en la memoria principal para desde aquí comenzar un proceso de transformación utilizado en CPU; el CPU a la vez está compuesto por la Unidad de control, la Unidad aritmética lógica y los registros que se encargan de realizar las operaciones aritméticas y lógicas, para brindar los resultados que vuelven a ser almacenados en la memoria principal de donde son enviados a los dispositivos de salida según la necesidad.

Todos los elementos de la arquitectura son utilizados en el proceso de ejecución de un programa es por eso que es necesario que el programador los tenga presentes:

- × Dispositivos de entrada.

- × Memoria principal.
- × CPU.
 - ✓ Unidad de control.
 - ✓ Unidad lógica aritmética.
 - ✓ Registros.
- × Dispositivo de salida.

4.2.5. Sistema de numeración

Todos los ordenadores intenamente manejan la información en un formato binario, es así que la información analógica, ya sea esta en forma de texto, imagen, sonido, debe ser trasformado por el dispositivo de entrada en su representación binaria para ser posteriormente almacenado en la memoria (Francis, 2020).

La memoria es una estructura que contiene datos en forma de direcciones y valores:

Address	Value
0x00	01001010
0x01	10111010
0x02	01011111
0x03	00100100
0x04	01000100
0x05	10100000
0x06	01110100
0x07	01101111
0x08	10111011
...	...
0xFE	11011110
0xFF	10111011

Figura 2: Representación de la memoria

4.2.6. Los lenguaje de programación

Cuando ya hablamos del contenido de la memoria, estamos hablando del software; el software que trabajo junto con el hardware de un computadora se lo clasifica en Sistema Operativo y Aplicativo; los dos se diferecia por el tipo de función que realiza.

El Sistema Operativo, tiene como función el control del hardware, para el diseño de esta software lo que interesas es lo que hace y como lo hace por que no estan orientado al usuario, pero por el contrario el software aplicativo a mas de su funcionalidad es necesario su apariencia.



Figura 3: Representación de la memoria

4.2.7. Preguntas de autocontrol

- × **Pregunta # 1:** ¿Porqué C++ se utiliza para dar los fundamentos de la programación.
 - ✓ Es la base de todos los lenguajes.
 - ✓ Todo se puede hacer con c++.
 - ✓ Tiene un curva de aprendizaje menos pronunciada.
- × **Pregunta # 2:** ¿Por qué el ordenador internamente solo puede manejar números binarios?.
 - ✓ Es más facil de entender.
 - ✓ El hardware solo maneja niveles de voltajes.
 - ✓ Los calculos son más rapidos con números binarios.
 - ✓ Ninguna de las anteriores.
- × **Pregunta # 3:** ¿Por qué la memoria principal no puede sustituir al registro
 - ✓ Es muy grande.
 - ✓ Es muy lenta.
 - ✓ Es volatil.
- × **Pregunta # 4:** ¿A que número decimal equivale el siguiente numero binario 10100101?.
 - ✓ 200
 - ✓ 250.
 - ✓ 165_{10} .

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
TERCERA SEMANA	13-septiembre-2021	17-septiembre-2021	2		C	Introducción a Linux y termux.	SESIÓN 7 <ul style="list-style-type: none"> ✓ El docente en una videoconferencia explicará conceptos básicos del Software Libre y el programa Termux para android. ✓ Evaluación sobre lo aprendido la clase anterior.
			2		C	Paquetes de Lunux: ejercicios prácticos(lang,vim,tree).	SESIÓN 8 <ul style="list-style-type: none"> ✓ Los estudiantes instalarán y configurarán los paquetes necesarios para trabajar en Termux.
			2		C	Taller de uso de comando de termux.	SESIÓN 9 <ul style="list-style-type: none"> ✓ Los estudiantes empiezan a elaborar informe sobre uso de termux y sus comandos. ✓ se califica la Actividad B1

4.2.8. Introducción a Linux y termux

Linux es un software que se distribuye libremente cuyo código es abierto, gracias a esta características de distribución se han creados muchas versiones del Sistema Operativo entre la que tenemos Ubuntu, Fedliza Suselinux, etc.

Año	Personaje	Contribución
1983	Richard Stallman	Crea el proyecto de GNU para crear un sistema operativo libre
1989	Richard Stallmen	Escribe la primera versión de la licencia GNU GPL
1991	Linux Torvalds	La primera versión pública del núcleo de linux en un servidor ftp
1992	Linux Torvalds	Licencia el núcleo de Linux bajo GNU GPL
1993	Más de 100 desarrolladores	Trabajan el núcleo Linux, y se crea un gran espectro, tambien se inicia a desarrollar wine
1994	Linux Torvalds	Presenta la version 1.1 de Linux. Red Hat y SUSE tamien presenta su version 1.1
1995	DEC y SUN SPARC	Linux funciona en DEC y SUN SPARC
2000	StarOffice	La suite StarOffice es ofrecida segun los terminos de GNU GPL.
2002	OpenOffice.org	La comunidad OpenOffice.org libera la version 1,0; tambien el navegador web libre Mozilla.

4.2.9. Comando/Paquetes de linux: Instalación, configuración y prácticas de uso

El sistema operativo linux tiene la ventana de comando al cual se le da comandos, que actúan como ordenes que ejecuta el núcleo del sistema, algunos programas son llamados por su nombre, es decir que al colocar su nombre en la línea de comando esta actúa como un comando que llama al programa.

Comando/paquete	Descripción
pwd	Comando para ver en que directorio se encuentra
ls	Comando para listar directorio y archivos
cd	Comando para ingresar a salir de directorio
mkdir	Comando para crear un directorio nuevo
mv	Comando para mover un archivo o directorio a otra ubicación, también sirve para cambiar el nombre de archivos o directorios.
tree	Paquete y comando para ver el contenido en forma de árbol debe instalarse con pkg install tree
vim	Paquete y comando para crear y editar archivos debe instalarse con pkg install vim
clang	Compilador de c++

Actividad C1-1: Taller sobre el uso de termux

Tiempo estimado para realizar esta actividad: 5 horas

Grabar un video tutorial sobre el **Taller sobre termux**, en el cual se explique los pasos que están detallados en esta guía.

El entregable es un video subido al canal de youtube y luego compartido en la plataforma Classroom

Para ser evaluada esta actividad se utilizara la rubrica del anexo #1.

4.2.10. Taller: Utilización de termux en la programación

Objetivo: Utilizar con agilidad los comandos mkdir, rm, mv, cd; para crear, borrar, mover y movilizarse entre los directorio.

1. Verificar en que directorio se encuentra usted ubicado.

```
$ pwd
```

2. Asegurarse que esté en el directorio de trabajo del usuario ~

```
$ cd ~
```

Creando directorios

3. Crear un nuevo directorio con el nombre **Mis_musicas**

```
$ mkdir Mis_musicas
```

4. Crear un nuevo directorio con el nombre **Mis_peliculas**

5. Crear un nuevo directorio con el nombre **Mis_fotos**

6. Crear un nuevo directorio con el nombre **Mis_tareas**
7. Crear un nuevo directorio con el nombre **Mis_documentos**
8. Crear un nuevo directorio con el nombre **Otra_informacion**
9. Listar los directorios creados utilizando el comando para desplegar una vista simple.

```
$ ls
```

10. Listar los directorios creados para ver la fecha de creación.

```
$ ls -l
```

Creando subdirectorios: _____

11. Ingresar al directorio **Mis_musicas**

```
$ cd Mis_musicas
```

12. Verificar que estamos dentro del directorio.

```
$ pwd
```

13. Crear un nuevo directorio con el nombre **Salsa**

```
$ mkdir Salsa
```

14. Crear un nuevo directorio con el nombre **Romantica**

15. Crear un nuevo directorio con el nombre **Clasica**

16. Crear un nuevo directorio con el nombre **Vallenato**

17. Listar los directorios creados en una vista simple para verificar que los directorios han sido creados.

```
$ ls
```

18. Retorne al directorio de trabajo, puede utiliza cualquier de los dos comandos siguiente.

```
$ cd ..
```

```
$ cd ~
```

19. Utilizar el comando **tree** para ver todos los directorios y subdirectorio al mismo tiempo.

```
$ tree
```

20. De la misma manera ingrese a los demás directorios y cree como mínimo dos subdirectorio con el nombre que usted crea conveniente.

Mover directorio dentro de otros directorio

21. Asegurarse que esté en el directorio de trabajo del usuario ~

```
$ cd ~
```

22. Haciendo el análisis de los contenido de los directorios creados, se llega a la conclusión que el directorio **Mis_tareas** debe esta dentro de directorio **Mis_documentos**; mueva el directorio con el comando.

```
$ mv Mis_tareas Mis_documentos/
```

23. Verifique la acción realizada con el comando **tree** para ver todos los directorios.

```
$ tree
```

24. Se ha llegado a la conclusión que el directorio llamado **Otra_informacion**. no va a ser utilizado por eso hay que eliminarlo.

```
$ rm -r Otra_informacion
```

si revisas con el comando **ls** el directorio ya no existe.

Cambiar los nombres de directorios:

25. Asegurarse que esté en el directorio de trabajo del usuario ~

```
$ cd ~
```

26. Se decide cambiar el nombre de los directorio creados a nombre mas simple posible

Mis_musicas ahora simplemente se llamará **Musicas**

```
$ mv Mis_musicas Musica
```

Mis_fotos ahora simplemente se llamara **Fotos**

```
$ mv Mis_fotos Fotos
```

Mis_tareas se llamara **Tareas**

```
$ mv Mis_tares Tareas
```

Mis_documentos se llamará **Documentos**

```
$ mv Mis_documentos Documentos
```

4.2.11. Preguntas de autocontrol

- × **Pregunta # 1:** ¿Porqué C++ se utiliza para dar los fundamentos de la programación.
 - ✓ Es la base de todos los lenguajes.
 - ✓ Todo se puede hacer con c++.
 - ✓ Tiene un curva de aprendizaje menos pronunciada.
- × **fase # 2:** El comando mv permite.
 - ✓ Mover un directorio a otrade ubicación.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** El comando mkdir permite.
 - ✓ Cambiar el nombre de un directorio.
 - ✓ Crear un directorio nuevo.
 - ✓ Eliminar un directorio.
 - ✓ Ninguna de las anteriores.
- × **fase # 4:** Indique cuan de las siguiente sentencias es correcta para el comando cd .
 - ✓ Este comando es un ataja para ir directamente a la raiz del sistemas de archivo.
 - ✓ Este comando permite crear un directorio con nombre .
 - ✓ Este comando es un atajo para ir directamente a directorio home.
 - ✓ Ninguna de las anteriores es correcta.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
CUARTA SEMANA	28-febrero-2022	04-marzo-2022	2		C	Introducción a Vim y sus comandos.	SESIÓN 10 <ul style="list-style-type: none"> ✓ En videoconferencia explicará sobre entornos de desarrollo aterrizando en Vim. ✓ Evaluación sobre lo aprendido.
			2		C	Ejecicios prácticos con Vim (directorios y archivo)	SESIÓN 11 <ul style="list-style-type: none"> ✓ Los estudiantes crearan y navegaran directorios y subdirectorios. ✓ Los estudiantes crearan y editaran archivos con Vim.
			2		C	Taller sobre Vim.	SESIÓN 12 <ul style="list-style-type: none"> ✓ Los estudiantes comienzan a elaborar un informa sobre el uso de Vim. ✓ se envia la Actividad C1

4.2.12. Introducción a Vim y sus comandos

Seleccionar un entorno de desarrollo que se adecue a las condiciones y preferencias del programador es una de las primeras cosas que debe haberse realizado para empezar la emocionante tarea de la programación.

Este capítulo describe las funciones más importantes de uno de los primeros editores de texto creado para funcionar con el Sistema Operativo linux, el cual se ha mantenido y evolucionado para competir con editores de texto que trabajan en entorno gráfico. **VI** era el nombre como inicialmente se lo conoció, pero que actualmente ha sido renombrado con **VIM** para indicar que es una versión modificada, que aunque sigue siendo para trabajar en modo texto también funciona para interfaces gráficas y tanto en la de texto como en la gráfica su principal potencial es cuando se lo utiliza únicamente con el manejo del teclado.

VIM fue un editor de texto, creado al inicio para ser el editor predeterminado de la ventana de comando de UNIX, inicialmente sirvió para poder crear y visualizar archivos de texto plano y muy cortos, pero posteriormente su uso se fue expandiendo hasta convertirse en un editor de texto avanzado utilizado como entorno de desarrollo para crear código en varios de los lenguajes de programación más importantes.

Su ejecución es simple solo hay que llamar a la ventana de comandos y escribir su nombre.

```
$ vim
```

La principal característica de vim es que utiliza combinaciones de teclas y es un programa de tipo modal.

- × **Modo normal o de comando:** Se podría decir que el modo normal de Vim es el estado de reposo. Otros editores de texto pasan la mayor parte del tiempo en lo que a Vim equivale al modo insertar. Para alguien que acaba de llegar a este editor modal, puede parecer extraño que pasemos la mayor parte del tiempo en el modo normal.
- × **Modo de inserción:** En modo inserción cuando se pulsan las teclas se edita el texto

como en otros editores. Se puede cambiar del modo comandos al modo inserción pulsando la tecla `i`. Hay un gran abanico de comandos para pasar al modo inserción, que difieren sustancialmente, pues permiten por ejemplo editar al final de la línea, en un punto concreto del texto, editar borrando una palabra, entre muchas otras. Un usuario experto puede sacar un gran provecho de la existencia de esta variedad de órdenes.

- × **Modo visual:** Este modo es una mejora respecto a `vi`. Mediante unas ciertas combinaciones de teclas en combinación con las teclas de movimiento del cursor, se puede marcar un área de texto, ya sea un grupo de líneas o un bloque. Una vez se tiene el texto marcado se pueden usar órdenes del modo comandos para manipularlo. Las operaciones que se pueden realizar en este modo son más simples que las del modo comandos.
- × **Modo línea de órdenes:** A este modo se accede pulsando la tecla dos puntos `:`. Tras los dos puntos se pueden introducir órdenes complejas, como por ejemplo buscar y reemplazar con expresiones regulares. Pulsando la tecla `Esc` se puede volver al modo órdenes. Las búsquedas se pueden realizar con la orden `/` (hacia adelante) y `?` (hacia atrás). También se pueden filtrar líneas mediante.



Figura 4: Vim un editor de texto y entorno de desarrollo muy potente

Ventajas

Fondo de escritorio con el logotipo de Vim. La mayoría de los usuarios que usan Vim aseguran que este editor incrementa su productividad comparándolo con editores más simples

una vez se ha superado la curva de aprendizaje. Las combinaciones de teclas se pueden memorizar empleando métodos mnemotécnicos, pues guardan relación con palabras inglesas. La complejidad intrínseca de aprender las instrucciones se ve recompensada por la mejora en la eficiencia. Los usuarios expertos pueden, usando unas pocas combinaciones de teclas, copiar texto, formatearlo u ordenarlo de muchas formas diferentes, que sólo se pueden realizar en la mayoría de editores mediante operaciones considerablemente más complejas. Basta con un poco de experiencia para notar que las combinaciones de instrucciones que permiten ediciones de texto complejas se facilitan con Vim.

5. Vim en modo normal

Usar Vim es una experiencia completamente distinta a usar cualquier otro editor de código. Vamos a hacer una breve demostración. Se utiliza una sintaxis de verbo-modificador-objeto. Empezamos en el modo normal, pulsaremos `i` (entrar al modo insertar) para introducir unos cuantos párrafos de texto, pulsamos **Esc** para volver al modo normal y aquí empieza la magia. No os preocupis, iremos mirando cada uno de ellos en detalle en los próximos post, pero podeis echándole un vistazo. Aprende algunos verbos: `v`(visual), `c`(change/cambiar), `d`(delete/borrar), `y`(yank/copiar).

Aprende algunos modificadores: `i`(inside/dentro de), `a` (around/alrededor), `t` (till/ hasta que encuentra el carácter). `f` (find /hasta que encuentra el carácter incluyendolo), `/` (buscar). Aprender algunos objetos: `w` (word/palabra), `s` (sentencia/frase), `p` (paragraphs/párrafo), `t` (tag/ para html/xml).

Los principales comandos utilizados en modo de comando son:

Comando	Descripción
ESC	Se asegura que VIM este en modo de comandos
i	cambia del modo de comando a modo de inserción
a	cambia del modo de comando a modo de inserción
h	mover una fila hacia arriba
l	mover una fila hacia abajo
j	mover un espacio a la izquierda
k	mover un espacio a la derecha
dd	Eliminar la linea actual de texto.
yy	copiar una linea
p	pegar la linea copiada
G	Mover el cursor al final del archivo.
gg	Mover el cursor al inicio del archivo
G	Mover el cursos al final del archivo
/	preparar a VIM para buscar una palabra
:s/hola/cola/g	remplar la palabra hola conla palabra
:	se prepara a VIM para recibir un comando
:set number	mostrar la numeración de cada linea
0	mover el cursor al inicio de la linea
\$	Mover el cursor al final de la linea

Cuadro 8: Comandos frecuentemente utilizados en la edición con VIM

VIM en modo insertar A este modo se puede ingresar presionando cualquiera de las siguientes letras (i,l,o,O,c,C) estando en modo se puede insertar o modificar el texto.

Actividad C1-2:Taller sobre el uso de vim

Tiempo estimado para realizar esta actividad: 5 horas

Grabar un video tutorial sobre el **Taller sobre vim**, en el cual se explique los pasos que están detallados en esta guía.

El entregable es un video subido al canal de youtube y luego compartido en la plataforma Classroom

Para ser evaluada esta actividad se utilizara la rubrica del anexo #1.

5.0.1. Taller uso de vim: Creación y edición de archivos

1. Ingrese al directorio **Tareas**, utilizando el siguiente comando.

```
$ cd ~/Tareas
```

2. Cree un directorio llamado **Practica1**.

```
$ mkdir Practica1
```

3. Ingrese al directorio **Practica1**.

```
$ cd Practica1
```

4. Crear un nuevo archivo llamado **suma.cpp** con el editor **vim**, utilice la siguiente qqinstrucción.

```
$ vim suma.cpp
```

5. Pulsa la tecla **i** (para entrar en el modo edición que permite escribir)
6. Escribir el siguiente texto sin dejar líneas en blanco:

```
#include<iostream>
using namespace std;
int main()
{
    float A,B,C;
    cin>>A>>B;
    C=A+B;
    cout<<C;
    return 0;
}
```

7. Hemos acabado de escribir, salimos del modo edición presionando la tecla **ESC**.
8. Ingresamos al modo comando presionando la tecla que contiene los dos puntos **:**
9. Grabar escribiendo **w** minúscula.
10. Vuelve al modo comando con dos puntos **:** y salga de vim escribiendo **q** minúscula.
11. Lista los archivos que hay en el directorio actual (use `ls -l`), y veraz el archivo **suma.cpp**.
12. Genere el archivo ejecutable con el siguiente comando.

```
$ g++ suma.cpp -o suma
```

13. Si no le presento ninguna error; lista los archivos que hay en el directorio actual (use `ls -l`).
14. Ejecute el programa de la siguiente manera.

```
$ ./suma
```

15. Abra otra vez el archivo creado con el editor vim.
16. Modifique su contenido para que quede de la siguiente manera:

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

```
int main()
{
float A,B,C;
cout<<"Ingrese 2 numero A B :";
cin>>A>>B;
C=A+B;
cout<<"El resultado es :";
cout<<C;
return 0;
}
```

17. Para modificarlo siga las siguientes instrucciones.

18. Muestre el número de las líneas escribiendo el comando

```
: set number
```

19. Asegure que esta al inicio del archivo con **ESC** y luego escribe **gg** en minúscula.

20. Valla a la novena línea escribiendo **:** **9**.

21. Cree una nueva línea en la parte superior con la tecla **O** mayúscula.

22. Escriba la siguiente línea y vuelva al modo de comando con ESC.

```
cout<<"El resultado es :";
```

23. Asegurese de estar en el modo comando presionando la tecla **ESC**.

24. Valla a la quinta línea escribiendo **:** **5**.

25. Abra una nueva línea en la parte inferior con la tecla **o** minúscula.

26. Escriba la siguiente línea y vuelva al modo de comando presionando **ESC**.

```
cout<<"Ingrese 2 numero separados de espacio A B :";
```

27. Modifique la antepenultima línea.

```
cout<<C<<endl;
```

28. Ahora vamos a cambiar la letra **A** por **x** escribiendo el siguiente comando.

```
:1,$s/A/x/g
```

29. De la misma manera cambien la letra **B** por la letra **y** escribiendo el siguiente comando.

Masculino.

```
:1,$s/B/y/g
```

30. De la misma manera cambie la letra **C** por la letra **z** con el siguiente comando. **Masculino.**

```
:1,$s/C/z/g
```

31. Finalmente obtenemos el siguiente contenido modificado.

```
#include<iostream>
using namespace std;
int main()
{
float x,y,z;
cout<<"Ingrese 2 número separados de espacio x y :";
cin>>x>>y;
z=x+y;
cout<<"El resultado es :";
cout<<z<<endl;
return 0;
}
```

32. Asegurese de que esta en el modo de comando presionando la tecla **ESC**:

33. Entra en el modo comando apretando la tecla dos puntos:

34. Grabar **W**.

35. Vuelve al modo comando y con **:** **q**.

36. Lista los archivos que hay en el directorio actual (use `ls -l`).

37. Genere el archivo ejecutable.

```
$ g++ suma.cpp -o suma
```

38. Si no le presento ninguna erro lista los archivos que hay en el directorio actual (use `ls -l`).

39. Ejecute el programa para ver como se ve con los cambios realizados.

```
$ ./suma
```

5.0.2. Preguntas para el autocontrol

× **Pregunta # 1:** ¿Cuál es la modalidad que permite darle ordenes a vim?.

- ✓ edición
- ✓ commado
- ✓ vista

× **fase # 2:** ¿Cuales son las letras que me permite mover verticalmente?.

- ✓ a,b
- ✓ j,k
- ✓ h,l.
- ✓ Ninguna de las anteriores.

× **Pregunta # 3:** ¿Cuál es la o las letra que permite ir al inicio del archivo?.

- ✓ G
- ✓ gg.
- ✓ s.

× **Pregunta # 4:** ¿Cuál es la tecla que permite y al final de la linea?.

- ✓ 0.
- ✓ l.
- ✓ \$.

× **Pregunta # 4:** ¿Cuál es la tecla que permite remplazar una letra?.

- ✓ 0.
- ✓ r.
- ✓ \$.

Actividad	Criterio	Nivel 1		
Nota máxima		100		
<div>Actividad C1:</div> <p>Analisis y diseño de problemas básicos de matemática, física o estadística, utilizando el diagrama de flujo transcribiendolo a lenguaje de código C++.</p>	Documentación y estética en el diagrama de flujo	Deficiente: El problema no esta bien definido, no se documenta las variables y el diagrama no esta esteticamente bien formado.	Bueno: El problema esta bien definido pero el diagrama no esta esteticamente bien formado o no hay documentación suficiente..	Excelente: Buena definición del problema, buena documentación y buena estética del diagrama.
		0-10	20	40
	Uso de standares en nombre de variables	Deficiente: La variables utilizadas no cumplen para nada con standares enseñados y utilizados en clase.	Bueno: Algunas variables utilizadas no cumplen con standares utilizas en clase ni en textos..	Excelente: Los nombres de las variables se siñen perfectamente a los standares indicados en la tería.
		0-10	20	40
	Correcto uso del modelo matemático	Deficiente: El modelo matemático utilizado no es el correcto y brinda una salida con errores lógicos.	Bueno: El modelo matemático es correcto pero no se aplica de toda su dimensión..	Excelente: El modelo matemático es el correcto y se aplica en todas sus dimensiones.
		0-10	20	40

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
QUINTA SEMANA	27-septiembre-2020	01-octubre-2021	2		C	Introducción a la programación.	SESIÓN 13 <ul style="list-style-type: none"> ✓ El docente en una videoconferencia explicará sobre el ciclo de vida del software. ✓ Se creara el programa de hola mundo(edición y copilacion) .
			2		C	Taller de introducción a la programación .	SESIÓN 14 <ul style="list-style-type: none"> ✓ Los estudiantes editaran y compilarar el programa de +,-,*,/.
			2		C	Taller sobre introducción a la programación.	SESIÓN 15 <ul style="list-style-type: none"> ✓ Los estudiantes comienzan a elaborar un informe sobre la práctica. ✓ se califica la Actividad C1 ✓ se envia la Actividad A1

5.0.3. Introducción a la programación

6. Ciclo de vida del Software

Es importante entender que el desarrollo de un software nunca termina pues siempre habrán oportunidades de mejoras, es por eso que el proceso se convierte en un ciclo.



Figura 5: Ciclo de vida del Software

7. Creación del Software: Principios básicos.

La creación de software es un proceso interactivo que se realiza en varias etapas.

- | | |
|-----------------------------|-------------------|
| 1. Análisis del problema. | 5. Verificación. |
| 2. Diseño del Algoritmo. | 6. Depuración. |
| 3. Codificación. | 7. Mantenimiento. |
| 4. Compilación y ejecución. | 8. Documentación. |

7.1. El análisis y diseño

El análisis del problema por lo general se lo realiza a la par que el Diseño del Algoritmo, este se puede dar gracias a que los problemas pequeños como los que se resuelven en este libro por lo general ya han sido analizados por los estudiantes durante los procesos de estudios que les precedió a este nivel académico (colegiatura), es por eso que directamente se puede pasar al diseño de algoritmo utilizando herramientas de diagrama de flujo para generar la secuencia de instrucciones que servirán para llevarlas a un archivo de texto con

7 CREACIÓN DEL SOFTWARE: PRINCIPIOS BÁSICOS.

una sintaxis definida por un lenguaje de programación en una siguiente etapa que se llama codificación.

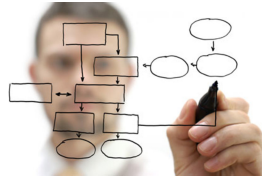


Figura 6: Análisis y diseño

7.2. La etapa de la codificación

En esta etapa, es donde el profesional informático o programador debe hacer gala de su habilidad para escribir el conjunto de instrucciones utilizando un conjunto de palabras reservadas aprendidas y memorizadas en su proceso de capacitación, el lenguaje de programación como se llama a este conjunto de palabras reservadas sera escogido en función del tipo de problema a resolver, en este libro se escogió el C++ para resolver problemas matemáticos; la codificación involucra crear un archivo entendible en primera instancia por el programador.



Figura 7: Fase de codificación

7.3. La etapa de compilación y ejecución

Es necesario que el computador interprete cada linea de código que se encuentra en el archivo creado en la etapa de codificación; después de contar con el archivo, se recurre a un programa del sistema operativo llamado compilador el cual en entorno linux es g++ este comando convierte el archivo fuente de C++ en un archivo que contiene un lenguaje entendible por la máquina, el cual posteriormente puede ser ejecutado.



Figura 8: Fase de compilación y ejecución

7.4. La etapa de verificación y depuración

Es muy difícil que un programa en su primera ejecución brinde los resultados deseados por el usuario o programador más aun si se trata de la resolución de un problema antes no resuelto, es por eso que después de una primera ejecución es necesario verificar que los resultados sean los correctos. y si no es así comenzar la etapa de DEPURACIÓN, la cual tiene como objetivo asegurar que el programa obtenga la salida deseada.

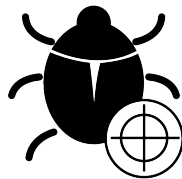


Figura 9: Fase de verificación y depuración

7.5. La etapa de mantenimiento

El software nunca termina de elaborarse y es que por lo general el programa a pesar de obtener las salidas deseadas, necesita actualización en función de adaptarse a los cambios que se dan en su entorno.



Figura 10: Fase de mantenimiento

7.6. Documentación

Aunque un programa elegantemente codificado utilizando las normas de programación recomendadas en los standares, no necesita de mucha documentación, si es verdad que la etapa de documentación es necesaria para la evolución y transportabilidad de un programa.

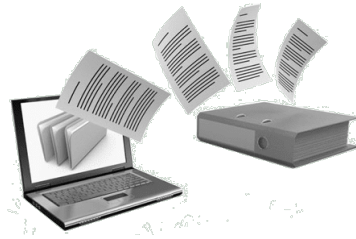


Figura 11: Fase de documentación del Software

7.6.1. Taller de programación básica 1

7.6.2. Taller de programación 2: Elaboración de informe

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

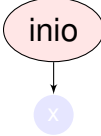
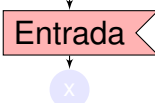
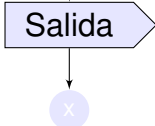
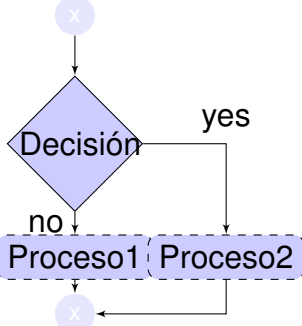
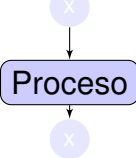
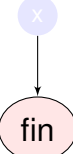
7 CREACIÓN DEL SOFTWARE: PRINCIPIOS BÁSICOS.

Actividad	Criterio	Nivel 1		
Nota máxima		100		
<div>Actividad A1:</div> <p>Avances de proyecto: Diagrama de flujo y código C++ de un problema de matemática, estadística o física que comprenda la toma de decisiones sobre los resultados.</p>	Documentación y estética en el código	Deficiente: Al código le falta comentarios importantes y no tiene estética en su presentación.	Bueno: El código tiene estética en su presentación pero faltan los comentarios importantes..	Excelente: El código tiene todos los comentarios y una buena estética.
		0-10	20	40
	Aplicaciones correctas de estándares en variables	Deficiente: Las variables utilizadas no cumplen para nada con estándares enseñados y utilizados en clase.	Bueno: Algunas variables utilizadas no cumplen con estándares utilizados en clase ni en textos..	Excelente: Los nombres de las variables se señalan perfectamente a los estándares indicados en la teoría.
		0-10	20	40
	Aplicaciones correctas de modelos matemáticos	Deficiente: El modelo matemático utilizado no es el correcto y brinda una salida con errores lógicos.	Bueno: El modelo matemático es correcto pero no se aplica de toda su dimensión..	Excelente: El modelo matemático es el correcto y se aplica en todas sus dimensiones.
		0-10	20	40

7 CREACIÓN DEL SOFTWARE: PRINCIPIOS BÁSICOS.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
SEXTA SEMANA	04-octubre-2021	08-octubre-2021	2	2	C	Ciclo de vida del software y Diagrama de Flujo.	SESIÓN 16 ✓ El docente explicará los elementos para el diagrama de flujo, con los programas de suma, resta, multiplicación y división.
			2		C	Ciclo de vida del software y Diagrama de Flujo .	SESIÓN 17 ✓ El docente explicará los elementos de decisión con el programa de 'El número mayor', 'La resta con resultado positivo'.
			2		C	Taller de Diagrama de Flujo.	SESIÓN 18 ✓ Los estudiantes comienzan a elaborar un informe sobre las prácticas Diagrama de Flujo.

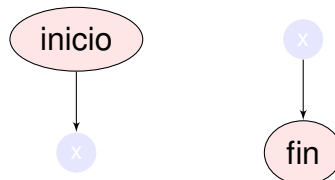
7.6.3. Figuras para el diagrama de flujo

Símbolo	Propósito	Descripción
	inicio	Indica el inicio de un programa.
	Entrada	Habilita el teclado(o dispositivo de entrada) para ingresar datos
	Salida	Habilita la pantalla(o dispositivo de salida) para presentar información al usuario)
	Decisión	Permite ejecutar de forma alternativa dos procesos distintos.
	Proceso	Indica específicamente una tarea que realiza el CPU ya sea para realizar una operación matemática, lógica de asignatura u otra
	fin	Indica la finalización del programa

Cuadro 13: Símbolos básicos utilizados para la creación de los diagrama de flujo

7.7. Inicio/fin

Este símbolo indica el proceso que se realiza antes de empezar a resolver el problema, el computador debe prepararse para la utilización de dispositivos de entrada y salida, y es en esta etapa donde se verifica que existen estos dispositivos y además están disponibles.



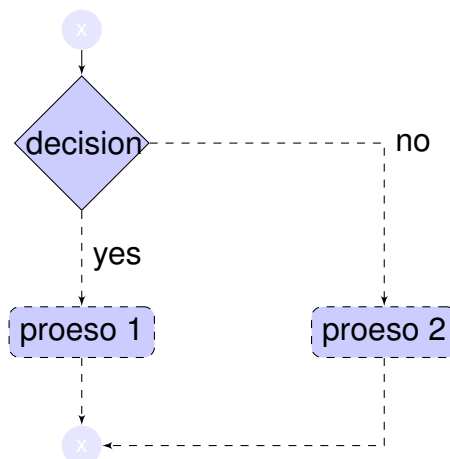
7.8. Símbolo de Proceso

Informa que el computador esta ocupado realizando algún proceso que por lo general tiene que ver con operaciones matemáticas.

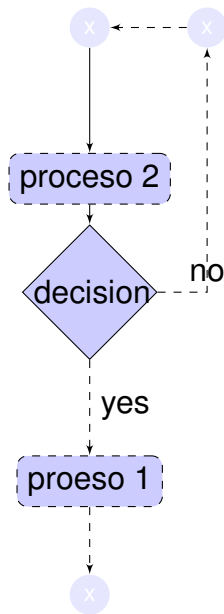


7.9. Símbolo de decisión

Este símbolo es utilizado en un punto en la secuencia de instrucciones donde se necesita mostrar más de una acción a seguir para llegar a la solución.

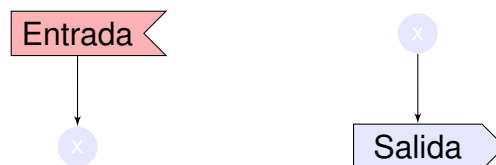


Estructura de repetición utilizando el simbolo de decisión.



7.10. Símbolo de entrada y salida

Estos dos símbolos son utilizados frecuentemente, tanto al inicio (entrada) como al final (salida) de la secuencia de instrucciones para obtener los datos a procesar y para presentar los resultados; por lo general los datos de entrada son provistos por una fuente externa a la computadora (como es el teclado) y los resultados se los muestra también generalmente por el monitor o pantalla.



7.10.1. Taller de Diagrama de flujo

8. Resolviendo un problema muy simple

Un problema muy simple que se presenta frecuentemente en la vida de las personas, es la resta o suma de dos números; aunque restar o sumar dos números (por ejemplo: $4-2=2$) puede parecer un problema que no necesitaría la ayuda de un computador, cuando la resta o suma se realiza entre números que pasan de los dos dígitos (ejemplo: $994-930=64$), al humano le toma un poco más de tiempo y trabajo hacerlo mentalmente, ya que el cerebro no ha sido diseñado para mantener en memoria la información por mucho tiempo, y es ahí

donde los computadores gracias a la arquitectura de John Vonn Newman se vuelve en la mejor aliada.

8.1. Análisis del problema

En el momento que tus oídos escuchan un problema, tu mente de forma automática va intentar encontrar la solución, comenzando a elaborar un conjunto de instrucciones posibles para obtener esta solución; en el problema planteado las siguientes son instrucciones que aunque de forma no muy precisa nos da una solución que podemos mejorar con el conocimiento de las técnicas enseñadas en la asignatura de fundamentos de programación.

"Algoritmo para restar dos números"

1. Conseguir el primer número.
2. Conseguir el segundo número.
3. Restar los dos números.
4. Presentar el resultado.

8.2. Mejorando nuestro algoritmo utilizando pseudo-código

Luego de tener una idea de las actividad que debemos realizar para solucionar el problema, se procede utilizar un lenguaje un poco más formal como es el pseudo-código, que como se indicóa anteriormente permite comunicar la propuesta de solución a otros colegas con los cuales hemos trabajado anteriormente.

Algoritmo para restar de dos números

```

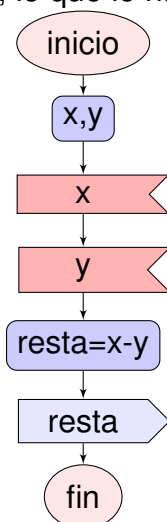
variable
entero a,b,suma
inicio
    escribir("Introduzca primer número entero")
    leer(a)
    escribir("Introduzca segundo número entero")
    leer(b)
    suma<-a+b
    escribir(suma)
fin

```

8.3. El diagrama de flujo para la resta de dos números

Aunque el pseudo-código desarrollado anteriormente las palabras reservadas son bastante expresivas y fácil de comprender por cualquiera que entienda el idioma español, no resultaría quizás entendible para algunas otras personas, por ejemplo que no maneje el español, es así que utilizar diagramas de flujo para resolver esta problema haría que esta solución sea más fácil de encontrar por personas que puedan llevar esta solución a un programa.

En el siguiente diagrama de flujo los símbolos podemos decir que rempazan a las palabras reservadas, lo que lo hace comprensible universalmente.



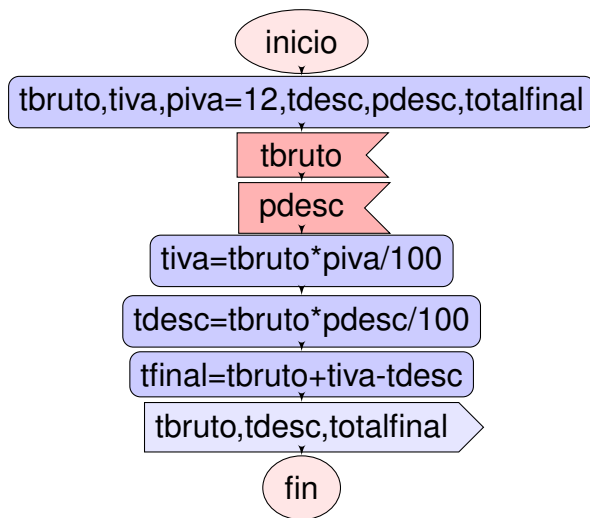
Variables	Descripción
x	Guarda el minuendo.
y	Guarda el sustraendo.
resta	Guarda el resultado de la resta o diferencia.

8 RESOLVIENDO UN PROBLEMA MUY SIMPLE

El símbolo inicio presenta la carga de las condiciones suficientes para comenzar el proceso, esto puede incluir la reserva de memoria, el símbolo input representa el ingreso de datos por teclado esto también incluye los mensajes para dar retroalimentación al usuario; el símbolo process incluye el trabajo en conjunto del procesados y la memoria para sumar los número y devolver los resultados a memoria; otra vez el símbolo output incluye la salida por pantalla; y finalmente el símbolo fin incluye la limpieza de la memoria.

9. El clásico programa del punto de venta

Enunciado: Un programa que permita ingresar el valor del total de las compras, porcentaje del iva y descuento luego calcular el total final a pagar.



Variable	Descripción
tbruto	valor total después de sumar los artículos
piva	porcentaje del iva
tiva	valor total de iva
pdesc	porcentaje a descontar
tdesc	Valor total del descuento
totalfinal	Valor final a pagar

9.1. Operaciones matemáticas

Las operaciones matemática básicas utilizadas en los diagramas de flujo y programas en c++ mostrado en este libro, son aquellas que utilizan los cinco operadores que se muestran a continuación:.

× + Suma o Adición.

× − Resta o Sustracción.

× * Multiplicación.

× / División.

× % División residual o módulo.

9.2. Operaciones lógicas

Las operaciones lógicas que utilizan los siguientes operadores:

× < Menor que.

× > Mayor que.

× == Iguala a.

× != No es igual a.

× <= Menor o igual.

× >= Mayor o igual.

9 EL CLÁSICO PROGRAMA DEL PUNTO DE VENTA

La característica de este tipo de operaciones, es que su resultado esta en el rango de (0,1) o (F,V) o (NO, SI); en cambio en las operaciones matemáticas, el resultado pueden ser cualquier número en el rango de los números naturales.

Dos ejemplo de operaciones lógicas son dadas como:

Operación	Interpretación	Operación	Interpretación
$2 < 3 = 1$	2 SI es menor que 3	$2 == 2 = 1$	2 SI es igual a 2

Se puede evaluar más de una operación lógica utilizando conectores.

		and	or	not
A	B	&&		not A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Cuadro 14: Tabla de verdad

Actividad A1:Entrega de la propuesta del proyecto final

Tiempo estimado para realizar esta actividad: 5 horas

Realizar diagrama de flujo de los ejercicios que se van a realizar.

Para ser evaluada esta actividad se utilizara la rubrica del anexo #1.

9.3. Preguntas de autocontrol

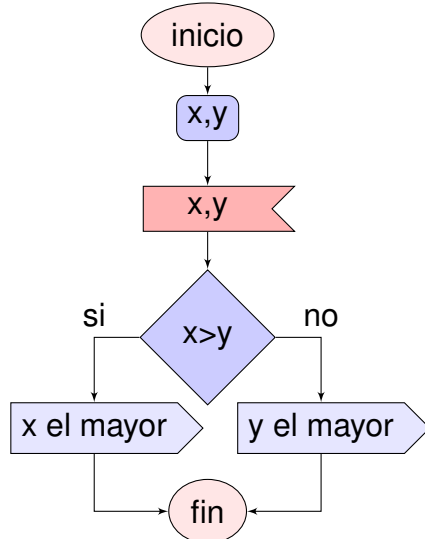
- × **Pregunta # 1:** Cual de los siguientes es el conjunto de figura que vamos a utilizar para el digrama de flujo.
 - ✓ inicio, variable, entrada, decisión, salida, proceso, fin.
 - ✓ salida, inicio, decision, proceso, entrada, fin.
 - ✓ inicio, salida, procesador, fin, entrada.
- × **Pregunta # 2:** Para que sirve la figura de proceso.
 - ✓ Para decidir.
 - ✓ Para terminar
 - ✓ Para calcular y declarar variables
 - ✓ Para calcular o calcular variables..
- × **Pregunta # 3:** Para qué sirve el simbolo de decision.
 - ✓ Para calculos matemáticos.
 - ✓ Para evaluar una operaciones lógica.
 - ✓ Para indicar terminación
- × **fase # 4:** Para que sirve la figura entrada.
 - ✓ Para que ingrese el sonivo
 - ✓ Para que ingrese las variables
 - ✓ Para que ingraes el valor de las variables.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
SEPTIMA SEMANA	11-octubre-2021	15-octubre-2021	2		C	Diagrama de Flujo(Desciones).	SESIÓN 19 ✓ El docente explicará las figura de descisión con los diagrama de flujo de las operaciones “El mayor de 3 número”, “El calculo de la edad”.
			2		C	Diagrama de Flujo (Estructura repetiva) .	SESIÓN 20 - (15-julio-2020) ✓ Suma de varios número, Facturación de varios artículos, Cuenta monegas, El más alto del curso.
			2		C	Taller de Diagrama de Flujo.	SESIÓN 21 - (17-julio-2020) ✓ Los estudiantes comienzan a elaborar un informa sobre las prácticas Digrama de Flujo. ✓ se califica la Actividad A1

9.4. Diagrama de Flujo (Descisiones)

El clásico programa del número mayor

Enunciado : Se desea crear un programa que permita ingresar dos números por teclado y evalúe estos dos números para saber cual de ellos es el mayor.

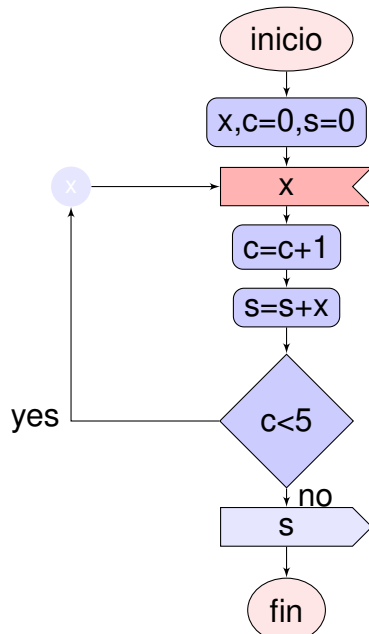


Variables	Descripción
x	Primer número ingresado por teclado.
y	Segundo número ingresado por teclado.

9.5. Diagrama de Flujo (Estructura de repetición)

El clásico programa de sumar varios números

Enunciado: Se necesita ingresar por teclado 5 números, estos números serán sumados y el resultado de esa suma debe ser presentada por pantalla.

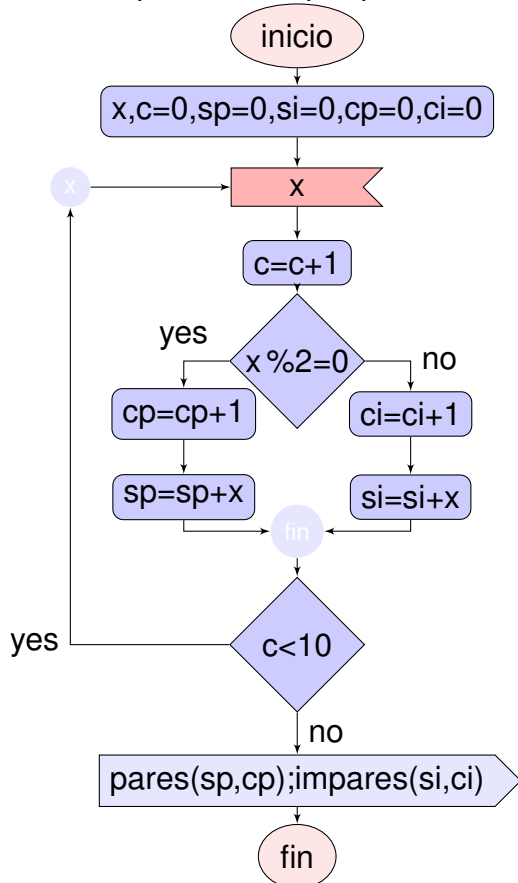


Variables	Descripción
x	Variable para almacenar los números ingresados por teclado.
c	Contador general
s	Acumulador

9.5.1. Taller de Diagrama de Flujo

10. Un programa que cuenta y suma los número pares e impares.

Enunciado: Se necesita un programa que permita ingresar varios número por teclado y de forma separada cuente y sume los números pares e impares, el resultado del conteo y de la suma debe presentarlo por pantalla.



Variables	Descripción
x	Variable para almacenar los números ingresados por teclado.
c	Contador general
cp	Contador de números pares
ci	Contador de números impares
sp	Acumulador de números pares
si	Acumulador de números impares

11. Problemas propuestos.

1. Utilizando la técnica del diagrama de flujo diseñar un programa que permite ingresar monedas de \$1 (un dolar), \$0.5 (cincuenta centavos) y \$0.25 (veinti cinco centavos); el programa calculara la cantidad total de monedas ingresadas, así como también la cantidad de dinero ingresado, además deberá calcular el total de monedas y el total en dinero pero por cada denominación de moneda, esta información deberá presentarla por pantalla.
2. Crear un diagrama de flujo que permita realizar 5 transacciones bancarias de tipo depósitos y retiro ; al final el diagrama muestra el saldo por pantalla.
3. Crear un diagrama de flujo que calcule la media, varianza y moda del promedio de notas de los estudiantes de un curso.
4. Crear un diagrama de flujo que calcule la desviación standar del promedio de las notas de los estudiantes de un curso.
5. Crear un diagrama de flujo que calcule la probabilidad de un evento.
6. Matemática
7. Crear un diagrama de flujo que muestra los 100 primeros número de la serie de fibonacci.
8. Crear un diagrama de flujo que calcule la suma de varias fracciones, la cantidad de fracciones la ingresa el usuario.
9. Crear un diagrama de flujo que calcule la velocidad de un vehículo dado el tiempo y la distancia; este diagrama debe indicar al usuario si la velocidad es muy baja ,normal, o va a exceso de velocidad (<40 baja velocidad, ≥ 40 y <60 velocidad normal , ≥ 60 exceso de velocidad).
10. Un algoritmo que indique el tiempo de impacto entre dos vehiculo que se dirigen en sentido contrario, dada la distancia entre ellos, la velocidad y la aceleración de cada uno de ellos.

11. Crear un diagrama de flujo que calcule la altura máxima alcanzada en un tiempo t , por una bala de cañón, dado el ángulo de disparo y la velocidad con que se dispara.
12. Crar un diagrama de flujo que permite declarar un vector permita ingresar valores dentro de la matriz y la presenta por pantalla.
13. Un algoritmo que calcula el vector resultante de la suma de dos vectores.
14. Un algoritmo que calcule el ángulo entre dos vectores
15. Un algoritmo que ordene los elemento de una matriz.
16. Ingreso de valores en una matriz y los presente por pantalla.
17. Un algoritmo que permita ingresar dos matrices y presenta la suma de sus elementos.
18. Crear un diagrama de flujo para caldular la edad de una persona, el diagrama debe permitir ingresar la fecha actual y la fecha de nacimiento.
19. Calcular el índice de masa corporal de una persona, el diagrama debe indicarle al usuario el peligro que corre si el índice no encuentra en el rango normal.
20. Crear un diagrama de flujo que calcule el total a pagar de un grupo de precios de productos ingresados por el usuario el diagrama debe mostrar también la suma de todos los productos, el iva a cobrar el valor de descuento y el total a pagar, el diagrama debe permitir ingresar la cantidad y los valores de cada artículo, el porcentaje del iva y el porcentaje de descuento.
21. Diseñar un programa utilizando la técnica del diagrama de flujo; este programa le permitirá al usuario ingresar un número el cual validará que este en el rango del 1 al 10, si es así presentará su equivalente en letras, en caso contrario mostrará un mensaje indicando que el número no se encuentra en el rango permitido y que lo intente otra vez.

11.0.1. Preguntas para el autocontrol

- × **Pregunta # 1:** Si i es la variable contadora y n contiene la cantidad de veces que se desea repetir un proceso, cual es la operación lógica que al dar verdadero permite ejecutar la repetición. .

- ✓ $i < n$

- ✓ $i > n.$

- ✓ $i == n.$

- × **fase # 2:** ¿Cuál es el resultado de la operación $x \% 2$ si x es un número par:?

- ✓ un número > 0

- ✓ un número $= 0$

- ✓ un número < 0

- × **fase # 3:** ¿Cuántas variables fueron necesarias para el algoritmo que suma varios números donde el usuario decida cuantos número sumar?.

- ✓ 2.

- ✓ 3.

- ✓ 5.

- ✓ Ninguna de las anteriores.

- × **fase # 4:** Cuantos tipos de bifurcaciones permite el simbolo de descisiones .

- ✓ 2.

- ✓ 3.

- ✓ 5.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
OCTAVA SEMANA	18-octubre-2021	22-octubre-2021	2		C	SESIÓN 22 - (20-julio-2020) Examen Primer Hemisemestre.	SESIÓN 22 ✓ El docente elabora y envia el formulario para tomar el examen del primer hemesemestre.
			2		C	SESIÓN 23 - (22-julio-2020) Examen del primer Hemisemestre.	SESIÓN 23 ✓ El docente elabora y envia el formulario para tomar el examen del primer hemesemestre.
			2		C	SESIÓN 24 - (24-julio-2020) Examen del primer Hemisemestre.	✓ El docente elabora y envia el formulario para tomar el examen del primer hemesemestre.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
NOVENA SEMANA	25-octubre-2021	29-octubre-2021	2	2	C	<p>SESIÓN 25 - (31-agosto-2020)</p> <p>Estructura básica del un programa en C++.</p>	<p>✓ El docente en videoconferencia explicará la estructura básica de un programa en c++ y su función principal con el programa Hola Mundo y Suma de Dos números.</p>
			2		C	<p>SESIÓN 26 - (02-septiembre-2020)</p> <p>Tipo de datos y declaración de variables.</p>	<p>✓ En videoconferencia el tema con el programa Operaciones matemáticas.</p>
			2		D	<p>SESIÓN 27 - (04-septiembre-2020)</p> <p>Taller en C++.</p>	<p>✓ Los estudiantes elaborarar u informe con sobre el programas de Factuación simple.</p> <p>✓ se envía la Actividad B2</p> <p>✓ se califica la Actividad B2</p>

11.0.2. Estructura básica de un programa en C++

Cuando se habla de programa en el contexto informático se entiende un conjunto de instrucciones que son proporcionados al computador para que este realice una tarea determinada (Bronson, 2007), por lo general esta tarea tiene que ver con la transformación de información utilizando procesos lógicos y matemáticos: este conjunto de instrucciones se escribe en un lenguaje entendible por el ordenador y cada instrucción genera trabajo para los diferentes componentes físicos del computador.

El programa o conjunto de instrucciones más simple que se puede crear utilizando el lenguaje C++ es el siguiente:

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout<< "Hola mundo" ;
6      return 0;
7  }
```

Código 1: Hola mundo

En las líneas de código anteriormente implementadas, se observa que todo programa en C++ debe implementar una función principal llamada **main**. esta función desde las últimas versiones debe retornar un valor que indica el resultado de la ejecución del mismo cuando se ejecuta normalmente debe devolver 0, pero si hubo alguna falla debe devolver un valor distinto de 0, la función main es el punto de entrada al programa y puede tener los siguiente 3 formatos.

```
int main(){ cuerpo}
```

```
int main( int argc, char * argv[]){ cuerdo}
```

argv : valor no negativo que indica el número de argumento enviados

argc : Putero al primer elemento de una matriz de punteros a cadena de texto terminado en

nulo.

12. Elementos básicos de un programa en c++

Dentro de las líneas de código que conforman un programa en C++ se podrán encontrar varios de estos siguientes elementos.

- × Palabras reservadas (main, return, if , while, do, .. etc).
- × Identificadores (nombre de variable ,funciones, nombre de programas, etc)
- × Caracteres especiales (como , punto y coma, llaves,etc)
- × Constantes.
- × Variables.
- × Expresiones.
- × Instrucciones.

Además de estos componentes básicos existe otros que son derivados como .

- × Bcle.
- × Contadores.
- × Acumuladores.
- × Interruptores
- × Estructuras(secuenciales, selectivas, repetitivas)

13. Identificadores

Un conjunto de elementos que se pueden observar en el pequeño programa de ejemplo escrito anteriormente (código 1) son las palabras reservadas que se han utilizado tales como: **main**, **include**, **iostream**, **in** , **cout**, **return**, las cuales se las llama identificadores y para su utilización hay que seguir las siguientes normas (Aguilar, 2008) .

- × El primer carácter puede sólo ser una letra o guión bajo.
- × Solo letras (A-Z, a-z) dígitos (0-9) o el guión bajo (_) pueden seguir al primer símbolo.
- × No se permite comenzar con doble guión bajo consecutivo.
- × Los identificadores son sensibles a las mayúsculas, así que si dos identificadores son iguales con la única diferencia de una o más letras mayúscula, el compilador de C++ lo considera diferentes : ejemplo nombre y noMbRe.
- × También hay que considerar que un identificador no puede coincidir con una palabra clave o con el de ninguna función de biblioteca.

Es importante indicar que las palabras entre comilla "Hola Mundo" no son identificadores por lo tanto no necesitan seguir las reglas indicadas.

14. Bloques

Otra última característica en este pequeño ejemplo de programa es el bloque, que son las líneas contenidas entre las llaves {...} y corresponde al conjunto de instrucciones creadas por el usuario para ser ejecutadas.

Los bloques en C++ en su forma general siguen el siguiente patrón:

"Bloques en C++"

```
{  
    <sentencia_1>;  
    <sentencia_2>;  
    <sentencia_3>;  
}
```

Los bloques son utilizados para agrupar un conjunto de sentencias que están relacionadas entre sí, con el fin de obtener un resultado en común, también los bloques pueden estar anidados por el mismo fin.

"Bloques anidado C++"

```
{  
    {  
        <sentencia\_1>;  
        <sentencia\_2>;  
        <sentencia\_3>;  
    }  
}
```

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

15. Las bibliotecas de C++

Continuando con la descripción del mismo simple ejemplo, se observa en la primera línea dos identificadores **#include<iostream>**, estos dos identificadores lo que hacen es importar un conjunto de bloques de instrucciones que serán utilizadas en las líneas de código creadas por el usuario-programador, el primer identificador **include** anteponiendo el símbolo **#**, se lo llama la directiva y es quien permite que los bloques de código dentro de **iostream** sean incluidos para ser llamados dentro del código, implementando la funcionalidad del identificador **cout**, el cual lo que hace es presentar el texto que está entre comillas dobles, por pantalla.

Así como la biblioteca **iostream**, C++ cuenta con mucho más que deben ser incorporada en función de lo que el usuario-programador quiera que su programa realice, alguna de estas bibliotecas son mostradas en el siguiente cuadro:

Biblioteca	Declaración	Descripción
vector	<code>#include<vector></code>	Permite trabajar con vectores como tipo de datos.
iostream	<code>#include<iostream></code>	Contiene los prototipos de las funciones que permiten el ingreso y salida de datos ya sea por pantalla o teclado.
math	<code>#include<math.h></code>	Contiene los prototipos de las funciones que permiten realizar operaciones matemáticas.
string	<code>#include<string.h></code>	Contiene los prototipos de las funciones que permiten manipular cadenas de caracteres.

Cuadro 18: Bibliotecas más utilizadas

16. La directiva

La directiva funciona en el pre-procesamiento de un programa en C++ y utiliza como argumento el nombre del archivo de la librería o biblioteca que va a ser importada, este

archivo puede estar en diferentes ubicaciones y hallados de diferentes formas, es por eso la llamada a estas bibliotecas se las realiza de diferentes maneras mostradas en la siguiente tabla:

<code>#include "archivo"</code>	Si se coloca el nombre de archivo dentro de comillas doble, el archivo es buscado en el mismo directorio donde se halla el archivo fuente.
<code>#include <archivo></code>	Si el nombre del archivo se coloca dentro de paréntesis angulares.

17. Espacio de nombres

Otra característica importante que se puede notar en este pequeño ejemplo es la utilización de **espacio de nombre** o **namespace**, esta estrategia permite agrupar el código en unidades lógicas, para poder hacer uso de identificadores con el mismo nombre, esta unidad lógica pueden contener tipos, funciones y objetos agrupados bajo un nombre común.

Para poder utilizar el espacio de nombre se le indicó al compilador que se va a usar el espacio de nombre, mediante la línea de instrucción **using namespace std**, donde **std** contiene la función **cout**.

Otra forma de utilizar el espacio de nombre es mediante el operador de ámbito "::", el cual se coloca antes de la función utilizarla y después del espacio de nombre.

```

1  #include <iostream>
2
3  int main ()
4  {
5      std::cout << "Hola mundo" ;
6      return 0;
7  }
```


17.1. Personalizando el espacio de nombre

Uno puede utilizar nombres de espacios para definir variables con el mismo identificador en el mismo programa sin que exista conflictos.

La estructura es la siguiente:

```
namespace <identificador >{  
    <tipo de datos> < variable>  
    <tipo de datos> < identificador de variable>  
    ....  
}
```

Donde <identificador> es un nombre colocado a criterio del programador, un ejemplo que muestra el uso del “espacio de nombre” es el siguiente.

```
1  #include<iostream>  
2  namespace jorge{  
3      int edad;  
4  }  
5  namespace pepe{  
6      int edad;  
7  }  
8  
9  int main()  
10 {  
11     jorge::edad=21;  
12     pepe::edad=15;  
13  
14     std::cout<< jorge::edad+ pepe::edad  ;  
15     return 0;  
16 }
```

18. Datos, tipos de datos y operaciones primitivas

La función principal del computador es transformar datos en información, y aunque el computador internamente maneja un solo tipo de dato que es el bit(1,0), para representar los datos utilizados por el ser humano en la vida real, de ha definido una unidad de información más grande llamada byte que consiste en 8 bit, con los cuales en su inicio sirvió para representar cualquier letra, número o símbolo (AaBbCc..1234...[]-/.); varios bytes son utilizados para almacenar unidades más grandes de información llamados tipos de datos 12 que se utilizan al momento de crear un programa en C++:

Los tipos de datos frecuentemente utilizados en la programación en C++ se muestran en la siguiente tabla 12:

Tipo	Tamaño (en bytes)	Rango
bit	1 bit	0 ó 1
char	1	0...255
signed char	1	-128...127
int	2	-32768...32767
unsigned	2	0...65535
long	4	-2147483648...2147483647
unsigned long	4	0...4294967295
float	4	$-1.5 * 10^{45} \dots +3.4 * 10^{38}$

Figura 12: Tipos de datos básicos utilizados por el lenguaje c

De estos tipos de datos, los fundamentales son:

- × Entero (int)
- × número de coma flotante (float).
- × caracteres (char).

char, int, float y double son palabras reservadas. Los tipos char, int y double tienen variaciones o modificadores de tipo de datos, tales como short, long, signed y unsigned, para permitir uso más eficiente de los tipos de datos.

Un ejemplo de código en C++ que nos ayuda a comprender el uso del tipo de datos **float** es el que permite calcular las cuatro operaciones básicas (suma, resta, multiplicación y división).

Código 2: **OperBasi1.cpp**: Operaciones Básicas

```

1  //=====
2  Operaciones basicas (suma, resta , multiplica y divide
3  //=====
4  #include<iostream>
5  using namespace std;
6  int main()
7  {
8      //para la suma de dos numeros.
9      float x1=3,x2=5,x3=8,x4=11,x5=10,x6=9,x7=1,x8=2,s,r,m,d;
10
11
12     s=x1+x2;    //Suma
13     r=x3-x4;    //Resta
14     m=x5*x6;    //Multiplicacion
15     d=x7/x8;    //Division
16
17     cout<<"El resultado de la suma fue :_"<<s<<endl;
18     cout<<"El resultado de la resta fue :_"<<r<<endl;
19     cout<<"El resultado de la multiplicacion fue :_"<<m<<endl;
20     cout<<"El resultado de la division fue :_"<<d<<endl;
21
22     return (0);
23 }

```

En esta porción de código además de aprender como declarar una variable de tipo float, también aprendemos que a una variable podemos asignarle un valor utilizando una expresión de asignación.

"Declaración de variable y Expresión de asignación"

```
float X1 = 3; X2 = 5;
```

También podemos observar la utilización de cuatro expresiones matemáticas que utilizan los operadores matemáticos básicos como son la suma (+), la resta (-), la división (/) y la multiplicación (*).

"Expresiones matemáticas"

```
s = x1 + x2; //suma  
r = x3 − x4; //resta  
m = x5 * x6; //multiplicación  
d = x7/x8; //división
```

Este código aunque no presenta errores al momento de copiarlo, no tiene una aplicación real puesto que la asignación de los valores de las variables deben ser a criterio del usuario que va a utilizar el programa no del programador, y si asignamos los valores directamente en el código el unico que puede cambiarlo seria el programador a una persona que tenga conocimiento de programación y esto no estaria bien, es así en el siguiente código se soluciona este inconveniente utilizando la función **cin** y **cout** para permitirle al usuario ingresar los valores que crea conveniente por teclado.

Código 3: **operbasi2.cpp**: Operaciones Básicas

```
1  //=====
2  Operaciones basicas (suma, resta , multiplica y divide)
3  //=====
4  #include<iostream>
5  using namespace std;
6  int main()
7  {
8      //para la suma
9      float x1,x2,s;
10     // para la resta
11     float x3,x4,r;
12     // para la division
13     float x5,x6,d;
14     // para la multiplicacion
15     float x7,x9,d;
16
17     cout<<" Ingrese los valores a sumar: ";
18     cout<<" Ingrese x1: "; cin>>x1;
19     cout<<" Ingrese x2: "; cin>>x2;
20
21     cout<<" Ingrese los valores a sumar: ";
22     cout<<" Ingrese x1: "; cin>>x1;
23     cout<<" Ingrese x2: "; cin>>x2;
24
25     cout<<" Ingrese los valores a sumar: ";
26     cout<<" Ingrese x1: "; cin>>x1;
27     cout<<" Ingrese x2: "; cin>>x2;
28
29     cout<<" Ingrese los valores a sumar: ";
30     cout<<" Ingrese x1: "; cin>>x1;
31     cout<<" Ingrese x2: "; cin>>x2;
32
33     s=x1+x2;
34     r=x3-x4;
35     m=x5*x6;
```

Código 3 (Cont.):

```

36     d=x7/x8;
37
38     cout<<"El resultado de la suma fue:_"<<s<<endl;
39     cout<<"El resultado de la resta fue:_"<<r<<endl;
40     cout<<"El resultado de la multiplicacion fue:_"<<m<<endl;
41     cout<<"El resultado de la division fue:_"<<d<<endl;
42
43     return (0);
44 }
```

Aunque estas dos versiones del programas **OperBasi.cpp** funcionan, estan muy lejos de satisfacer las necesidades de un usuario real, pues es dificil que alguien quiera rrealizar las cuatro operaciones matemáticas al mismo tiempo, esto no tiene sentido; lo que se si se aproxima a los requerimientos reales, es querer realizar una operacion a la vez y para eso debemos darle al usuario la opción de seleccionar que desea hacer:

Código 4: **operbasi3.cpp**: Operaciones Básicas

```

1  //=====
2  Operaciones basicas (suma,resta , multiplica y divide)
3  //=====
4  #include<iostream>
5  using namespace std;
6  int main()
7  {
8      int op;
9      float x1,x2,x3,x4,x5,x6,x7,x8,s,r,m,d;
10
11     cout<<"Ingrese el numero de la operacion que quiere realizar:_";
12     cout<<"1._Suma:_"<<endl;
13     cout<<"2._Resta:_ "<<endl;
14     cout<<"3._Producto:_ "<<endl;
15     cout<<"4._Division:_ ";
16
17     cout<<"Ingre la opcion:_"; cin>>op;
```

Código 4 (Cont.): **operbasi3.cpp**: Operaciones Básicas

```

18
19  if (op==1){
20      cout<<" Ingrese los valores a sumar: ";
21      cout<<" Ingrese x1: "; cin>>x1;
22      cout<<" Ingrese x2: "; cin>>x2;
23      s=x1+x2;
24      cout<<" El resultado de la suma fue: " <<s<<endl;
25  }
26  if (op==2){
27      cout<<" Ingrese los valores a sumar: ";
28      cout<<" Ingrese x1: "; cin>>x1;
29      cout<<" Ingrese x2: "; cin>>x2;
30      r=x3-x4;
31      cout<<" El resultado de la resta fue: " <<r<<endl;
32  }
33  if (op==3){
34      cout<<" Ingrese los valores a sumar: ";
35      cout<<" Ingrese x1: "; cin>>x1;
36      cout<<" Ingrese x2: "; cin>>x2;
37      m=x5*x6;
38      cout<<" El resultado de la multiplicacion fue: " <<m<<endl;
39  }
40  if (op==4){
41      cout<<" Ingrese los valores a sumar: ";
42      cout<<" Ingrese x1: "; cin>>x1;
43      cout<<" Ingrese x2: "; cin>>x2;
44      d=x7/x8;
45      cout<<" El resultado de la division fue: " <<d<<endl;
46  }
47  return (0);

```

Observamos que a medida que vamos mejorando el programa para que se adapte a las necesidades de un usuario real, el código se hace más extenso y más difícil de comprender, por ejemplo imagine que el usuario quiere que al seleccionar la opción de suma este le permita sumar varios números, o tiene la necesidad de sumar varios números positivos y

varios número negativo y luego restar el resultado de los número positivos con el resultado de los número negativos; en la sección sobre diagrama de flujo ya se reviso este tipo de problema por lo que podemos deducir que el código para estos nuevos requerimientos seria difícil de leer; es por eso que se hace importante el uso de funciones.

Código 5: **operbasi4.cpp**: Operaciones Básicas

```

1 //=====
2 Operaciones basicas(suma,resta , multiplica , divide)
3 //=====
4 #include<iostream>
5 using namespace std;
6 int main()
7 {
8     int op;
9     float x1,x2,x3,x4,x5,x6,x7,x8,s,r,m,d;
10
11 do{
12     system("clear");
13     cout<<"Ingrese el numero de la operacion que quiere realizar: ";
14     cout<<"1.- Suma: "<<endl;
15     cout<<"2.- Resta: "<<endl;
16     cout<<"3.- Producto: "<<endl;
17     cout<<"4.- Division: ";
18     cout<<"5.- Salir: ";
19
20     cout<<"Ingre la opcion: "; cin>>op;
21
22     if (op==1){
23         cout<<"Ingrese los valores a sumar: ";
24         cout<<"Ingrese x1: "; cin>>x1;
25         cout<<"Ingrese x2: "; cin>>x2;
26         s=x1+x2;
27         cout<<"El resultado de la suma fue: "<<s<<endl;
28     }
29     if (op==2){
30         cout<<"Ingrese los valores a sumar: ";
31         cout<<"Ingrese x1: "; cin>>x1;

```


Código 5 (Cont.): **operbasi4.cpp**: Operaciones Básicas

```
32     cout<<" Ingrese_x2_:_" ; cin>>x2;
33     r=x3-x4;
34     cout<<" El_resultado_de_la_resta_fue:_"<<r<<endl;
35 }
36
37 if (op==3){
38     cout<<" Ingrese los valores a sumar:_" ;
39     cout<<" Ingrese_x1_:_" ; cin>>x1;
40     cout<<" Ingrese_x2_:_" ; cin>>x2;
41     m=x5*x6;
42     cout<<" El_resultado_de_la_multiplicacion_fue:_"<<m<<endl;
43 }
44 if (op==4){
45     cout<<" Ingrese los valores a sumar:_" ;
46     cout<<" Ingrese_x1_:_" ; cin>>x1;
47     cout<<" Ingrese_x2_:_" ; cin>>x2;
48     d=x7/x8;
49     cout<<" El_resultado_de_la_division_fue:_"<<d<<endl;
50 }
51
52 } while (op!=5);
53     return (0);
54 }
```

Actividad	Criterio	Nivel 1		
Nota máxima		100		
<div>Actividad B2:</div> <p>Uno de los 5 estudiantes de cada grupo, creará un repositorio llamado B2, cada integrante creará un subdirectorío con el nombre “ApellidoNombre”, dentro de este se cargará la solución a 7 problemas propuestos en la unidad 9(diagrama de flujo y programa en C++).</p>	Documentación y estética en el código	Deficiente: Al código le falta comentarios importantes y no tiene estética en su presentación.	Bueno: El código tiene estética en su presentación pero faltan los comentarios importantes..	Excelente: El código tiene todos los comentarios y una buena estética.
		0-10	20	40
	Uso de estándares con variables en el código	Deficiente: Las variables utilizadas no cumplen para nada con estándares enseñados y utilizados en clase.	Bueno: Algunas variables utilizadas no cumplen con estándares utilizados en clase ni en textos..	Excelente: Los nombres de las variables se señalan perfectamente a los estándares indicados en la teoría.
		0-10	20	40
	Uso correcto de modelos matemáticos	Deficiente: El modelo matemático utilizado no es el correcto y brinda una salida con errores lógicos.	Bueno: El modelo matemático es correcto pero no se aplica de toda su dimensión..	Excelente: El modelo matemático es el correcto y se aplica en todas sus dimensiones.
		0-10	20	40

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
DECIMA SEMANA	01-noviembre-2021	05-noviembre-2021	2	2	C	SESIÓN 28 - (07-septiembre-2020) Estructura de selección (if-else) en C++. CODE-SESSION: FUNDPROG-10-01-PA/PB	✓ En videoconferencia se explicará el tema con los programa El mayor de dos números, Resta de con saldos negativos y El vehiculo más veloz.
			2		C	SESIÓN 29 - (09-septiembre-2020) Estructura de selección (if-else) en C++. CODE-SESSION: FUNDPROG-10-02-PA/PB	✓ En videoconferencia se explica los programas El mayor de 3 números, Calculo de la edad y Contador de monedas.
			2		C	SESIÓN 30 - (11-septiembre-2020) Taller en C++. CODESESSION: FUNDPROG-10-03-PA/PB	✓ Los estudiantes elaborarán un informe con los programas de Clasificación de monedas. ✓ se envía la Actividad C2

18.0.1. Estructura de selección (if-else) en C++

"Sintaxis estructura if"

```
if(<operación_lógica>
{
instruccion1;
instruccion2;
.....
instruccion_n;
}
else
{
instruccion1;
instruccion2;
.....
instruccion_n;
}
```

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

Actividad	Criterio	Nivel 1		
Nota máxima		100		
<div>Actividad C2:</div> <p>Creando un repositorio C2 con el mismo procedimiento de la actividad B2, cada estudiante transcribirá el programa en C++ a mano y los subirá al repositorio con las mejoras respectivas.</p>	Documentación y estética en el código	Deficiente: Al código le falta comentarios importantes y no tiene estética en su presentación.	Bueno: El código tiene estética en su presentación pero faltan los comentarios importantes..	Excelente: El código tiene todos los comentarios y una buena estética.
		0-10	20	40
	Uso de estándares con variables en el código	Deficiente: Las variables utilizadas no cumplen para nada con estándares enseñados y utilizados en clase.	Bueno: Algunas variables utilizadas no cumplen con estándares utilizados en clase ni en textos..	Excelente: Los nombres de las variables se señalan perfectamente a los estándares indicados en la teoría.
		0-10	20	40
	Uso correcto de modelos matemáticos	Deficiente: El modelo matemático utilizado no es el correcto y brinda una salida con errores lógicos.	Bueno: El modelo matemático es correcto pero no se aplica de toda su dimensión..	Excelente: El modelo matemático es el correcto y se aplica en todas sus dimensiones.
		0-10	20	40

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
ONCEVA SEMANA	08-noviembre-2021	12-noviembre-2021	2	0	C	Estructura de Repetición (do-while) en C++.	SESIÓN 31 ✓ El docente en videoconferencia explicará la estructura de selección en c++ y su función principal con el programa Suma de varios números, El estudiante más altos.
			2	0	C	Estructura de Repetición (do-while) en C++.	SESIÓN 32 ✓ En videoconferencia se explica los programas Contador de monedas, Tabla de multiplicar, Llenado, ordenamiento y presentación de matrices.
			2	0	C	Taller en C++.	SESIÓN 33 ✓ Los estudiantes comenzaran a elaborar un informe con los programas de Facturación de varios articulos ingresado por teclado.

18.1. Estructura de repetición (do-while)

En esta porción de se utiliza hay que aplicar el bucle o lazo de repetición (do.....while), cuya sintaxis es la siguiente:

"Sintaxis de un do...while(..);"

```
do{  
instruccion1;  
instruccion2;  
.....  
instruccion_n;  
}while(<operacion lógica>;
```


18.1.1. Taller para la práctica de estructura de repetición en la programación de C++**1. DATOS GENERALES.**

Nombre del estudiante: _____
Grupo : _____
Fecha de realización: _____
Fecha de entrega: _____

2. OBJETIVO.

Desarrollar destrezas y habilidades en el uso de estructuras repetitivas en la programación de C++.

3. INSTRUCCIONES.

En el repositorio de GitHub llamado PRACTICAS, creada por una de los integrantes del grupo, cada estudiante subirá el programa en C++ que suma varias fracciones. El usuario debe poder ingresar la cantidad de fracciones que quiere sumar y luego el programa le irá pidiendo el numerador y denominador de cada fracción, para finalmente presentar en un formato natural las fracciones ingresados y el resultado en número decimal.

4. MATERIALES A UTILIZAR.

- × Conexión a internet.
- × Una Cuenta en GitHub.
- × Teléfono inteligente.
- × Software, Termux, vim, git, clang

5. ACTIVIDADES POR DESARROLLAR.

- a) Crear un repositorio en Github llamado PRACTICA.
- b) Cada estudiante clonará el repositorio de su grupo.
- c) Cada estudiante creará un directorio dentro del repositorio clonado, el nombre de este directorio tendrá el formato "ApellidoNombre".

- d) Utilizando su smartphone y con la ayuda de un video tutorial los estudiantes editarán el programa suma de fracciones.
- e) El estudiante compilará y corregirá los errores que aparezcan hasta obtener el archivo ejecutable.
- f) Cada estudiante actualizará (git push) el repositorio.

6. RESULTADOS OBTENIDOS

7. CONCLUSIONES

8. RECOMENDACIONES.

Otro ejemplo de que las funciones evolucionan y se hacen más grandes y complejas es la función de la división que se lo puede interpretar como el resultado de una fracción y se puede extender la función para que devuelva el resultado de la suma de varias fracciones.

Código 6: **sumafracciones()**

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i=0,s=0,cposi=0,cnega=0,sposi=0,snega=0;
6      float x;
7      cout<< "Ingresa la cantidad de fracciones:";
8      cin>>cf;
9      do{
10         cout<< "Ingresa el numerador:";    cin>>n;
11         cout<< "Ingresa el denominador:";    cin>>d;
12         i=i+1;
13         cociente=cociente+n/d;
14         cout<<n<<" / "<<d<<" + ";
15     }while(i<cf);
16     cout<<" = " <<s<< endl;
17     return 0;
18 }
```

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
DOCEAVA SEMANA	15-noviembre-2021	19-noviembre-2021	2	0	C	Proyecto Final integrador.	SESIÓN 34 - (21-septiembre-2020) ✓ El docente en videoconferencia explicará de forma detallada los lineamiento a seguir para hacer y presentar el proyecto final.
			2	0	C	Plantilla de menu integrador.	SESIÓN 35 - (23-septiembre-2020) ✓ En videoconferencia se explica una plantilla para crer el menu integrador del proyecto final.
			2	0	C	Taller en tema proyecto final C++.	SESIÓN 36 - (25-septiembre-2020) ✓ Los docente en videoconferencia explicara un problema de Matemática, Física y Estadística. ✓ se envía la Actividad A2

18.1.2. Proyecto Final Integrador**"Sentencia switch"**

```
switch(<identificador de variable>) {  
    case <valor de la variable>;  
        sentencias;  
        .....;  
    case <valor de la variable>;  
        sentencias;  
        .....;  
    case <valor de la variable>;  
        .....  
    default;  
}
```

Como vimos en el programa **OperBasi5.cpp** se necesito utilizar varias estructuras **if** para seleccionar que operacion el usuario desea ejecutar, esa implementación puede resultar poco eficiente puesto que si el usuario desea ejecuta una solo opción, el programa evaluará todas las opciones utilizando el CPU innecesariamente, es por eso que una mejor opción para implementar un menú es mediante la estructura **switch-case** presentada en el recuadro anterior.

Implementando el programa **OperBasi5.cpp** con la estructura **case** el nuevo programa quedaría de la siguiente manera:

Código 7: **OperBasi6.cpp**

```
19  #include <iostream>
20  using namespace std;
21  int main()
22  {
23      int i=0,s=0;
24      float x;
25      do{
26
27          switch{op}
28          {
29              case 1:
30                  suma();
31                  break;
32              case 2:
33                  resta();
34                  break();
35              case 3:
36                  divide();
37                  break;
38              case 4:
39                  producto();
40                  break;
41              case 5:
42                  break;
43          }
44          i=i+1;
45          s=s+x;
46      }while(op != 5);
47      cout<<"La_suma_fue:_"<<s<< endl;
48      return 0;
49  }
```

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

Actividad	Criterio	Nivel 1		
Nota máxima		100		
<div>Actividad A2:</div> <p>Proyecto final: Un programa en C++ con varias funcionalidades integradas a través de un menú.</p>	Documentación y estética en el código	Deficiente: Al código le falta comentarios importantes y no tiene estética en su presentación.	Bueno: El código tiene estética en su presentación pero faltan los comenarios importantes..	Excelente: El código tiene todos los comentarios y una buena estética.
		0-10	20	40
	Uso de standares con varialbes en el código	Deficiente: La variables utilizadas no cumplen para nada con standares enseñados y utilizados en clase.	Bueno: Algunas variables utilizadas no cumplen con standares utilizas en clase ni en textos..	Excelente: Los nombres de las variables se siñen perfectamente a los standares indicados en la tería.
		0-10	20	40
	Uso correcto de modelos matemáticos	Deficiente: El modelo matemático utilizado no es el correcto y brinda una salida con errores lógicos.	Bueno: El modelo matemático es correcto pero no se aplica de toda su dimensión..	Excelente: El modelo matemático es el correcto y se aplica en todas sus dimensiones.
		0-10	20	40
Nota máxima		100		

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
TRECEAVA SEMANA	22-noviembre-2021	26-noviembre-2021	2		C	<p>SESIÓN 37 - (28-septiembre-2020)</p> <p>Revisión de Avances Grupo A,B,C.</p>	<p>✓ El docente en videoconferencia revisará los avances de algunos grupos de estudiantes.</p> <p>✓ se califica la Actividad C2</p>
			2		C	<p>SESIÓN 38 - (30-septiembre-2020)</p> <p>Revisión de Avance grupo D,E,F.</p>	<p>✓ En docente videoconferencia revisará los avances de otro grupo de estudiante.</p> <p>✓ se califica la Actividad C2</p>
			2		C	<p>SESIÓN 39 - (02-octubre-2020)</p> <p>Revisión de Avances G,H, I.</p>	<p>✓ Los docente en videoconferencia revisa avances de otro grupos de estudiantes.</p> <p>✓ se califica la Actividad C2</p>

Preguntas para el autocontrol

Están agrupadas por fases.

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
CATORCEAVA SEMANA	29-noviembre-2021	03-diciembre-2021	2		C	SESIÓN 40 - (05-octubre-2020) Funciones, Librerías y Clases	✓ El docente en videoconferencia explicará que son y como funcionan las Funciones, Librerías y la Clases.
			2		C	SESIÓN 41 - (07-octubre-2020) Programa de Funciones y Librería Operaciones básicas	✓ En docente videoconferencia explicara el programa de las operaciones básicas utilizando funciones y librerías.
			2		C	SESIÓN 42 - (09-octubre-2020) Programa con clases llamada persona	✓ Los docente en videoconferencia explicará el programa que maneja una clase llamada persona y su aplicación.

19. Funciones declaradas por el usuario

Como pudimos observar en la evolución del código anterior, a medida que el programa se hace más real, en el sentido de poder ser usable en un ambiente real, su complejidad funcional aumenta, y mantener todas las líneas de código en un mismo archivo no resulta conveniente para poder seguir actualizándolo, pues demasiadas líneas de código lo convierte en un documento difícil de leer. Es por eso que se hace necesario crear unidades funcionales más pequeñas, es decir su funcionalidad dividir las en archivos que contengan solo las líneas de código relacionadas a través de la función que estas realicen.

El siguiente recuadro presenta la sintaxis que debe ser usada para crear una función

"Sintaxis de un función"

```
<tipo datos><identificador_de_funcion>(<para_1>...<para_n>)  
{  
  instruccion_1;  
  instruccion_2;  
  .....  
  .....  
  instruccion_n;  
  return <identificador_de_variable>;  
}
```

Podemos resaltar que una función declarada por el usuario, tiene la misma estructura que la función principal **main**, es decir devuelve un tipo de datos y recibe parametros declarados de cualquier tipo; pero estas funciones creados por el usuario/programador, deben ser declaradas antes de la función principal **main** que es de donde por lo general se las llama.

En los siguientes ejemplos se muestran cuatro funciones (`suma()`, `resta()`, `producto()` y `division()`), y luego se implementa una función principal **main** de donde se las llama, se espera que estos ejemplos permitan su mejor comprensión.

Función suma : Suma dos números pasados como argumento.

Código 8: **suma.cpp**: función suma

```
1 //=====
2 // Funcion suma
3 //=====
4 #include<iostream>
5 using namespace std;
6 float suma(float x1, float x2)
7 {
8     return (x+y);
9 }
```

Podemos colocar la función resta en otro archivo

Código 9: **resta.cpp**: función resta

```
1 //=====
2 // Funcion resta
3 //=====
4 #include<iostream>
5 using namespace std;
6 float resta(float x1, float x2)
7 {
8     return (x-y);
9 }
```

El archivo para la función producto.

Código 10: **producto.cpp**: función producto

```
0 //=====
1 // Funcion producto
2 //=====
3 #include<iostream>
4 using namespace std;
5 float producto(float x1,float x2)
6 {
7     return(x*y);
8 }
```

Creamos un archivo llamado `division.cpp` para escribir la función `divide` que divide dos números.

Código 11: **division.cpp**: Operaciones Básicas

```

0 //=====
1 // Funcion division
2 //=====
3 #include<iostream>
4 using namespace std;
5 float divide(float x1, float x2)
6 {
7     return(x1/x2);
8 }

```

Para poder utilizar estos archivos que contienen las funciones de `suma()`, `resta()`, `producto()` y `division()` es necesario utilizar las directivas que cargan los archivos junto con el programa principal de la siguiente manera:

Código 12: **OperBasi5.cpp**

```

1 //=====
2 // Operaciones Basica Recargada
3 //=====
4 #include<iostream>
5 #include "suma.cpp"
6 #include "resta.cpp"
7 #include "producto.cpp"
8 #include "divide.cpp"
9
10 using namespace std;
11 int main()
12 {
13     float x,y;
14
15     cout<<"ingre_el_numero_de_la_opracion_que_desa_realizar";
16     cout << "1=suma"<<endl;

```


Código 12 (Cont.): **OperBasi5.cpp**: función suma

```
17 cout << "2=resta "<<endl;
18 cout << "3=producto "<<endl;
19 cout << "4=division "<<endl;
20 cin>>opc;
21
22 if (opc==1){
23     cout<<"Ingrese_el_primer_sumando:_"; cin<<x;
24     cout<<"Ingrese_el_segundo_sumando:_"; cin<<y;
25     cout<<"El_suma_dio : "<<suma(x,y)<<endl;
26 }
27
28 if (opc==2){
29     cout<<"Ingrese_el_minuendo:_"; cin<<x;
30     cout<<"Ingrese_el_sustraendo:"; cin<<y;
31     cout<<"El_resta_dio :_"<<resta(x,y)<<endl;
32 }
33 if (opc==3){
34     cout<<"Ingrese_el_primer_factor:_"; cin<<x;
35     cout<<"Ingrese_el_segundo_factor:"; cin<<y;
36     cout<<"El_multiplicacion_dio : "<<producto(x,y)<<endl;
37 }
38 if (opc==4){
39     cout<<"Ingrese_el_dividendo:_"; cin<<x;
40     cout<<"Ingrese_el_divisor:"; cin<<y;
41     cout<<"El_division_dio :_"<<divide(x,y)<<endl;}
42 }
```

Utilizar funciones es importante por que esto permite concentrar el esfuerzo que se realiza en la etapa de la actualización, en una parte del código, por ejemplo, si deseamos que la suma sea de varios número solo tenemos que modificar el archivo de suma.cpp y reemplazar esta porción de código.

Código 13: **sumanúmeros()**

```
50  #include<iostream>
51  using namespace std;
52  int main()
53  {
54      int i=0,s=0,cposi=0,cnega=0,sposi=0,snega=0;
55      float x;
56      do{
57          cout<< "Ingresa un numero entero: "; cin>>x;
58          i=i+1;
59          s=s+x;
60          if (x>0){
61              cposi=cposi+1;    sposi=sposi+x;
62          } else {
63              cnega=cnega+1;    snega=snega+x;
64          }
65      }while(i<10);
66      cout<<"La_suma_fue:_ "<<s<< endl;
67      return 0;
68  }
```

19.1. Clases: Estructura

Una característica que identifica al lenguaje C++ es la programación orientada a objetos, esto significa que se pueden utilizar estructuras llamadas clases, las cuales permiten agrupar atributos y eventos o funciones de un objeto para ser utilizados como una sola unidad lógica.

En C++ los atributos reciben el nombre de datos miembros y los métodos el nombre de funciones miembros de la clases incluyendo los miembros constructores que permiten inicializar un objeto y los destructores que permiten destruir un objeto (Sierra, 1998).

De forma general la estructura clases es:

"estructura de datos CLASS"

```
class <Identificador de la clase> {  
    public;  
    <tipo de datos> <identificador de variable>;  
    .....;  
    protected:  
    <tipo de datos> <identificador de variable>;  
  
    private:  
    <tipo de datos> <identificador de variable>;  
  
    .....;  
    case <valor de la variable>;  
  
    .....;  
    case <valor de la variable>;  
    .....  
    .....  
    ;  
}
```

Un ejemplo de clase bastante utilizado, es la clase persona que en C++ se implementa de la siguiente manera:

Código 14: **Persona.cpp**

```
1
2 // Clase base Persona:
3 class Persona {
4     public:
5         Persona(char *n, int e);
6         const char *LeerNombre(char *n) const;
7         int LeerEdad() const;
8         void CambiarNombre(const char *n);
9         void CambiarEdad(int e);
10
11     protected:
12         char nombre[40];
13         int edad;
14 };
15
16 // Clase derivada Empleado:
17 int main()
18 {
19     Persona person;
20 };
```

19.2. Preguntas de autocontrol

- × **Pregunta # 1:** A quien se le atribuye la arquitectura del computador.
 - ✓ char babage.
 - ✓ Jhon Vom Neumann.
 - ✓ Rober Noise.
- × **fase # 2:** Dar seguimiento al proceso académico de las asignaturas.
 - ✓ Llevar el control de asistencia tanto de maestrantes como de docentes.
 - ✓ Manejar toda la información que genera el proceso académico (silabos, módulos, tareas, etc.)
 - ✓ Control de cumplimiento de actividades de acuerdo a un cronograma.
 - ✓ Ingreso y difusión de calificaciones.
- × **fase # 3:** Gestionar la información académica.
 - ✓ Repositorio digital.
 - ✓ Control de carga y descarga de información al Repositorio.
 - ✓ Búsqueda de información.
- × **fase # 4:** Manejar la relación con los maestrantes y docente.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.
 - ✓ Integración con Microsoft Team y Google Classroom.

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
QUINCEAVA SEMANA	06-diciembre-2021	10-diciembre-2021	2		C	<p>SESIÓN 43 - (12-octubre-2020)</p> <p>Revisión de Avances Grupo A,B,C.</p>	<p>✓ El docente en videoconferencia revisará los avances de algunos grupos de estudiantes.</p> <p>✓ Actividad A2</p>
			2		C	<p>SESIÓN 44 - (14-octubre-2020)</p> <p>Revisión de Avance grupo D,E,F.</p>	<p>✓ EL docente en videoconferencia revisará el proyecto final de los grupos D,E,F y asignara la calificación.</p> <p>✓ Actividad A2</p>
			2		C	<p>SESIÓN 45 - (16-octubre-2020)</p> <p>Revisión de Avances G,H, I.</p>	<p>✓ En videoconferencia se revisa proyecto final al grupo G,H,I y asigna la calificación.</p> <p>✓ Actividad A2</p>

Semana	Desde	Hasta	Ho. Doce	H. Auton.	Tipo	TEMA TRATADO	ACTIVIDADES REALIZADAS
DIECISEISAVA SEMANA	13-diciembre-2021	17-diciembre-2021	2		C	SESIÓN 46 - (19-octubre-2020) Examen 2do Hemisemestre.	SESIÓN 46 ✓ El docente elabora y envia el formulario para tomar el examen del 2do hemesemestre.
			2		C	SESIÓN 47 - (21-octubre-2020) Examen del 2dor Hemisemestre.	SESIÓN 47 ✓ El docente elabora y envia el formulario para tomar el examen del 2do hemesemestre.
			2		C	SESIÓN 48 - (23-octubre-2020) Examen del 2do Hemisemestre.	SESIÓN 48 ✓ El docente elabora y envia el formulario para tomar el examen del 2do hemesemestre.

Semana	Desde	hasta	Actividad
	23-agosto-2021	27-agosto-2021	Entrega de guía de estudio a dirección de carrera
	30-agosto-2021	03-septiembre-2021	Entrega y subida de silabo
INICIO PRIMER PARCIAL			
Semana #1	30-agosto-2021	03-noviembre-2021	Clases - Presentación del Docente
Semana #2	06-septiembre-2021	10-noviembre-2021	Clases - Envio de primera actividad(B1)
Semana #3	13-septiembre-2021	17-noviembre-2021	Clases - Entrega de calificación B1
Semana #4	20-septiembre-2021	24-noviembre-2021	Clases - Envía de Actividad (C1)
Semana #5	27-septiembre-2021	27-noviembre-2021	Clases - Entrega de calificaciones C1
Semana #6	04-octubre-2021	08-octubre-2021	Clases - Envía de Actividad (A1 Anteproyecto)
Semana #7	11-octubre-2021	15-octubre-2021	Clases - Entrega de calificación A1
Semana #8	18-octubre-2021	22-octubre-2021	EVALUACION SUMATIVA PRIMER PARCIAL
INICIO SEGUNDO PARCIAL			
Semana #9	25-octubre-2021	29-octubre-2020	Clases - Envio de actividad B2
Semana #10	01-noviembre-2021	05-noviembre-2021	Clases Entrega de calificación B2
Semana #11	08-noviembre-2021	12-noviembre-2021	Clases - Envio de actividad C2
Semana #12	15-noviembre-2021	19-noviembre-2021	Clases - Entrega de calificacion C2
Semana #13	22-noviembre-2021	26-noviembre-2021	Clases - Envio de Actividad (A2 -Proyecto final)
Semana #14	29-noviembre-2021	03-diciembre-2021	Clases - Entraga calificación A2
Semana #15	06-diciembre-2021	10-diciembre-2021	EVALUACION SUMATIVA FINAL
Semana #16	13-diciembre-2021	17-diciembre-2021	Evaluación de habilitación

19.3. Formas y tipos de evaluación

EVALUACION	TIPOS	OPCIONES	PTOS.	Σ
Medio Ciclo	Acumulativa 70 %	Actividad A1	3.5	
		Actividad B1	1.75	
		Actividad C1	1.75	
	Examen medio ciclo 30 %	Evaluación sumativa	3	
SUBTOTAL:				10
Fin de Ciclo	Acumulativa 70 %	Actividad A1	3.5	
		Actividad B1	1.75	
		Actividad C1	1.75	
	Examen final 30 %	Evaluación sumativa	3	
SUBTOTAL:				10
PROMEDIO:				10

Ing. Stalin Francis Ms.c

DOCENTE

Ing. Stalin Francis MSc.

**COORDINADOR DE ÁREA ACADÉMICA
DE PROGRAMACIÓN**

Ing. Baster Estupiñan Ortiz, MSc.

**DIRECTOR DE CARRERA DE INGENIERÍA
EN TECNOLOGÍA DE LA INFORMACIÓN**

Ing. Fabiola Espantoso

SECRETARIA

19.4. Rubricas para autoevaluación del silabo

CRITERIO	SI	NO
DESCRIPCIÓN DE LA ASIGNATURA		
Los datos informativos esta completos:	<input type="checkbox"/>	<input type="checkbox"/>
La descripción de la asignatura es clara:	<input type="checkbox"/>	<input type="checkbox"/>
El Objetivo General es claro:	<input type="checkbox"/>	<input type="checkbox"/>
Los resultados de aprendizaje son claros(1 por cada unidad)	<input type="checkbox"/>	<input type="checkbox"/>
Se indica la metodología de aprendizaje(Aula invertida, otros)	<input type="checkbox"/>	<input type="checkbox"/>
Se indica los contenidos (Unidades y tema)	<input type="checkbox"/>	<input type="checkbox"/>
Estan definidas las 6 actividades (A1,B1,C1,A2,B2,C2)	<input type="checkbox"/>	<input type="checkbox"/>
Estan definidas las 3 evaluaciones(E1, E1,R(Recuperacion))	<input type="checkbox"/>	<input type="checkbox"/>
GESTIÓN DURACIÓN DE ESTUDIO		
Estan definidas las 16 semanas de clases	<input type="checkbox"/>	<input type="checkbox"/>
Estan definidos los días y horas de cada clases “ virtual”	<input type="checkbox"/>	<input type="checkbox"/>
Estan definidas las actividades autónomas y su duración	<input type="checkbox"/>	<input type="checkbox"/>
Estan definidas las fechas y horas de tutorias	<input type="checkbox"/>	<input type="checkbox"/>
GETIÓN INTERACCIÓN DOCENTE-ESTUDIANTE		
Estan definidos los temas para cada clase “virtual”	<input type="checkbox"/>	<input type="checkbox"/>
Esta definido el orden del día para las clases virtuales	<input type="checkbox"/>	<input type="checkbox"/>
Esta definido el tema para cada tutoria	<input type="checkbox"/>	<input type="checkbox"/>
BIBLIOGRAFÍA		
Esta indicada la bibliografía básica	<input type="checkbox"/>	<input type="checkbox"/>
Esta indicada la bibliografía complementaria	<input type="checkbox"/>	<input type="checkbox"/>
Esta indicada la bibliografía recomendada	<input type="checkbox"/>	<input type="checkbox"/>
Esta indicada la bibliografía audiovisual	<input type="checkbox"/>	<input type="checkbox"/>